2011

ρήση τεχνολογιών Διαδικτύου(Web Services, EST, WCF) για την ανάκτηση και ανάλυση εδομένων μέσω του Παγκόσμιου Ιστού

Ονοματεπώνυμο:
Λιανού Σπυριδούλα
Αριθμός Μητρώου:
2007147
Εισηγητής καθηγητής:
Παναγιώτης Τριτάκης

Περίληψη

Αυτή η πτυχιακή εργασία παρέχει μία επισκόπηση για τις υπηρεσίες διαδικτύου, περιλαμβάνοντας τις βασικές αρχές τους και επεξήγηση του WCF framework. Τα θέματα που καλύπτονται περιέχουν και τις απαραίτητες τεχνολογίες που χρησιμοποιούνται για την υλοποίηση των υπηρεσιών διαδικτύου. Ένα δείγμα των τεχνολογιών αυτών είναι η XML, η υπηρεσιοστρεφής αρχιτεκτονική και διάφορα πρωτόκολλα που είναι ικανά να αντικαταστήσουν τη χρήση του HTTP με ποιο αποτελεσματικό τρόπο όπως το SOAP. Οι υπηρεσιοστρεφής εφαρμογές που παράγονται από το WCF framework καλύπτονται σε βάθος για να γίνει κατανοητό τι ικανότητες, γνώσεις και αποφάσεις απαιτούνται για να υλοποιηθούν. Σε μία προσπάθεια περεταίρω εμβάθυνσης γίνεται και η παρουσίαση μιας εφαρμογής αυτού του είδους που υλοποιήθηκε για να υπηρετήσει αυτό το σκοπό.

Ευχαριστίες

Η παρούσα πτυχιακή εργασία εκπονήθηκε στο τμήμα Τεχνολογίας Πληροφορικής και Τηλεπικοινωνιών του ΑΤΕΙ Καλαμάτας στο παράρτημα Σπάρτης στα πλαίσια των δραστηριοτήτων της κατεύθυνσης της πληροφορικής. Ωστόσο η πραγματοποίησή της δεν θα ήταν εφικτή χωρίς τη βοήθεια συγκεκριμένων ανθρώπων.

Θα ήθελα να ευχαριστήσω θερμά τον καθηγητή και επιβλέποντα της πτυχιακής μου εργασίας κ. Παναγιώτη Τριτάκη για την ευκαιρία που μου έδωσε να εργαστώ σε ένα τόσο σύγχρονο και ταυτόχρονα ενδιαφέρον αντικείμενο, αλλά και για την αμέριστη βοήθεια που μου προσέφερε.

Επίσης θα ήθελα να ευχαριστήσω ιδιαίτερα τον προϊστάμενο του τμήματος κ. Γρηγόρη Καραγιώργο και το εκπαιδευτικό προσωπικό της σχολής για την επιπλέον βοήθεια που μου προσέφεραν κατά τη διεξαγωγή της πτυχιακής εργασίας.

Τέλος θα επιθυμούσα να εκφράσω τις ευχαριστίες μου προς την οικογένεια μου για την κατανόηση και τη συμπαράσταση με κάθε δυνατό τρόπο καθ' όλη τη διάρκεια των σπουδών μου.

Περιεχόμενα

Πίνακας Εικόνων

Εισαγωγή

Όταν ανακαλύφθηκε ο σιδηρόδρομος η επιρροή του στο εμπόριο ήταν τεράστια και κατ' επέκταση στην παγκόσμια οικονομία . Με τον ίδιο τρόπο οι υπηρεσίες ιστού άλλαξαν τα διεθνή δεδομένα του διαδικτυακού εμπορίου. Αυτό συμβαίνει γιατί συνδέουν προγράμματα σε απομακρυσμένα σημεία σε όλη την υδρόγειο μεταφέροντας μεγάλο όγκο δεδομένων αποτελεσματικότερα και φθηνότερα από ποτέ άλλοτε. Για τις εταιρείες και τους καταναλωτές σημαίνει γρηγορότερη καλύτερη και πιο παραγωγική επικοινωνία. ***Οι υπηρεσίες ιστού έχουν αλλάξει τα πάντα.***

Το διαδίκτυο ξεκίνησε υποστηρίζοντας την ανθρώπινη αλληλεπίδραση με δεδομένα και γραφικά σε μορφή κειμένου. Οι άνθρωποι χρησιμοποιούν το διαδίκτυο καθημερινά κυρίως για να αγοράσουν καταναλωτικά αγαθά και να πληροφορηθούν τις οικουμενικές ειδήσεις. Το επίπεδο αλληλεπίδρασης για πολλούς χρήστες είναι ικανοποιητικό. Από την άλλη πλευρά όμως υπάρχουν και αυτοί που θεωρούν ότι θα μπορούσε να βελτιωθεί σε πολλά σημεία.

Αυτή η βελτίωση θα προκύψει αντιμετωπίζοντας το εξής: ο βασισμένος σε κείμενο ιστός δεν υποστηρίζει την αλληλεπίδραση λογισμικού με ικανοποιητικό τρόπο ιδιαιτέρως όταν μεταφέρεται μεγάλος όγκος πληροφορίας. Είναι επιτακτική μια πιο αποτελεσματική μέθοδος η οποία επιτρέπει στις εφαρμογές να αλληλεπιδρούν απευθείας η μία με την άλλη εκτελώντας αυτόματα εντολές που διαφορετικά θα πρέπει να περαστούν χειροκίνητα χρησιμοποιώντας έναν browser.

Άτομα ή εταιρείες που κάνουν buss iness μέσα στο δίκτυο και χρειάζονται ένα τρόπο να δημοσιεύουν συνδέσμους για εφαρμογές και δεδομένα παρόμοιο με αυτόν που δημοσιεύουν συνδέσμους για την ιστοσελίδα τους. Οι εφαρμογές που είναι βασισμένες στο διαδίκτυο πρέπει να είναι ικανές να μπορούν να βρουν, να έχουν πρόσβαση και να αλληλεπιδρούν αυτόματα με άλλες εφαρμογές που βασίζονται και αυτές στον ιστό.

Οι υπηρεσίες διαδικτύου βελτίωσαν το διαδίκτυο επιτρέποντας την επικοινωνία μεταξύ προγραμμάτων. Μέσω της μεγάλης διάδοσης και αποδοχής των υπηρεσιών διαδικτύου, εφαρμογές με ποικίλες τοποθεσίες διαδικτύου μπορούν απευθείας να επικοινωνήσουν σαν ένα μεγάλο σύστημα τεχνολογίας πληροφοριών.

Σκοπός της συγκεκριμένης εργασίας είναι να γίνουν κατανοητές σε βάθος οι υπηρεσίες ιστού και τα πλεονεκτήματά τους. Η πτυχιακή είναι χωρισμένη σε δύο μέρη το θεωρητικό και το πρακτικό. Το θεωρητικό μέρος απαρτίζεται από τρία κεφάλαια. Στο πρώτο κεφάλαιο ορίζονται και αναλύονται οι βασικές έννοιες των υπηρεσιών ιστού. Στο επόμενο κεφάλαιο γίνεται εκτενής αναφορά στα τα συστατικά τους στοιχεία καθώς και παραδείγματα για τη περαιτέρω κατανόησή τους. Στο τελευταίο κεφάλαιο υπάρχει παρουσίαση του WCF framework. Ένα framework κατάλληλο για τη δημιουργία υπηρεσιοστρεφών εφαρμογών.

Για το πρακτικό μέρος έχει δημιουργηθεί μία εφαρμογή που υπόκειται στο σενάριο μίας εταιρίας πώλησης προϊόντων. Η εταιρία με βάση αυτό το πρόγραμμα έχει τη δυνατότητα να αναζητεί προϊόντα πελάτες και παραγγελίες με διάφορα κριτήρια. Το ενδιαφέρον που παρουσιάζεται στη συγκεκριμένη εφαρμογή είναι ότι όντας μία υπηρεσία ιστού που χρησιμοποιεί το πρωτόκολλο SOAP μπορεί να επικοινωνήσει με τον client χωρίς να εμποδιστεί από τα firewalls που είναι εγκατεστημένα.

Κεφάλαιο 1: Υπηρεσίες Ιστού

1.1Εισαγωγή

Η τεχνολογία των υπηρεσιών Ιστού αλλάζει το Διαδίκτυο, αυξάνοντας τον *eyeball web* με τη δυνατότητα να παραχθεί ο *transactional web*. Ο *eyeball web* εξουσιάζεται από τις αλληλεπιδράσεις του προγράμματος με το χρήστη(*program-to-user*) και της επιχείρησης με τους καταναλωτές(business-to-consumer(B2C)). Ο *transactional web* θα εξουσιαστεί από τις program-to-program και business-to-business(B2B) αλληλεπιδράσεις. Αυτός ο μετασχηματισμός τροφοδοτείται από το program-to-program πρότυπο επικοινωνίας των υπηρεσιών Ιστού που στηρίζονται στα υπάρχοντα και αναδυόμενα πρότυπα όπως το πρωτόκολλο μεταφοράς υπερκειμένων (HTTP), την Extensible Markup Language (XML), το
Simple Object Access Protocol (SOAP), την Web Services Description Language (WSDL) και τέλος το Universal Description, Discovery, and Integration (UDDI).

Η τεχνολογία των υπηρεσιών Ιστού παρέχει έναν πρότυπο προγραμματισμού με ουδέτερη γλώσσα και ουδέτερο περιβάλλον που επιταχύνει την ολοκλήρωση μιας εφαρμογής μέσα και έξω από την επιχείρηση. Η ολοκλήρωση της εφαρμογής μέσω των υπηρεσιών Ιστού παράγει τα εύκαμπτα χαλαρά συνδεμένα επιχειρησιακά συστήματα. Επειδή οι υπηρεσίες Ιστού εφαρμόζονται εύκολα ως wrappering τεχνολογία γύρω από τις υπάρχουσες εφαρμογές παρέχονται και τα προτερήματα της τεχνολογίας πληροφοριών. Δεδομένου ότι η υιοθέτηση της νοοτροπίας των υπηρεσιών Ιστού επεκτείνεται, ο αριθμός τους θα αυξηθεί, ενθαρρύνοντας την ανάπτυξη δυναμικότερων προτύπων για just in time εφαρμογές και της επιχειρησιακής ολοκλήρωσης μέσω του Διαδικτύου.

Σε αυτό το πρώτο κεφάλαιο θα γίνει μία προσπάθεια να κατανοήσουμε τις υπηρεσίες διαδικτύου. Αρχικά θα παρατεθούν διάφοροι ορισμοί των υπηρεσιών ιστού από μερικές σημαντικές επιχειρήσεις που αναπτύσσουν τέτοιες εφαρμογές. Θα υπάρξει μία ιστορική αναδρομή και θα αναπτυχθεί το μοντέλο μιας υπηρεσίας ιστού για να γίνει κατανοητό τι είναι μια τέτοια υπηρεσία. Στη συνέχεια θα αναλυθούν τα χαρακτηριστικά της αλλά και τι είναι η σωρός των υπηρεσιών ιστού καθώς θα πραγματοποιηθεί και μία σύντομη αναφορά στα συστατικά τους τα οποία θα σχολιαστούν εκτενέστερα στο κεφάλαιο 2. Στο τέλος θα εξηγήσουμε τους όρους Choreography και Orchestration και θα κλείσουμε με την ασφάλεια των υπηρεσιών ιστού.

1.2 Xml υπηρεσίες Ιστού

Οι XML υπηρεσίες Ιστού είναι οι θεμελιώδεις δομικές μονάδες στην κίνηση στο διανεμημένο υπολογισμό στο διαδίκτυο. Τα ανοιχτά πρότυπα και η εστίαση στην επικοινωνία και τη συνεργασία μεταξύ των ανθρώπων και των εφαρμογών έχουν δημιουργήσει ένα περιβάλλον όπου οι XML υπηρεσίες Ιστού γίνονται η πλατφόρμα για την ολοκλήρωση της εφαρμογής. Οι εφαρμογές κατασκευάζονται χρησιμοποιώντας τις πολλαπλές XML υπηρεσίες Ιστού από τις διάφορες πηγές που λειτουργούν μαζί ανεξάρτητα από που κατοικούν ή το πώς εφαρμόστηκαν.

Υπάρχουν πολλοί ορισμοί για τις υπηρεσίες ιστού. Κάθε επιχείρηση που δημιουργεί τέτοιες υπηρεσίες έχει δώσει και έναν ορισμό. Πρέπει να σημειωθεί ότι όλοι οι ορισμοί έχουν κοινά χαρακτηριστικά:

- Οι XML υπηρεσίες Ιστού εκθέτουν χρήσιμη λειτουργία στους χρήστες Ιστού μέσω ενός τυποποιημένου πρωτοκόλλου Ιστού. Στις περισσότερες περιπτώσεις, το πρωτόκολλο που χρησιμοποιείται είναι το SOAP.
- Οι υπηρεσίες Ιστού XML παρέχουν έναν τρόπο να περιγράψουν τις διεπαφές τους με αρκετές λεπτομέρειες για να επιτρέψουν σε έναν χρήστη να χτίσει μια εφαρμογή πελατών που να μπορεί να επικοινωνήσει μαζί τους. Αυτή η περιγραφή παρέχεται συνήθως σε ένα έγγραφο XML αποκαλούμενο γλωσσικό έγγραφο περιγραφής υπηρεσιών Ιστού(WSDL).
- Οι XML υπηρεσίες Ιστού εγγράφονται έτσι ώστε οι δυνητικοί χρήστες να μπορούν να τις βρουν εύκολα. Αυτό γίνεται με την καθολική περιγραφή και την ολοκλήρωση ανακαλύψεων (UDDI).

Θα μπορούσαν όμως να δοθούν παραδείγματα από διάφορες εταιρίες για να γίνουν κατανοητά αυτά που λέγονται:

1. Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

   Ορισμός από w3c.
2. Web services are Web based applications that use open, XML-based standards and transport protocols to exchange data with clients.

   Ορισμός από oracle.
3. Web services extend the World Wide Web infrastructure to provide the means for software to connect to other software applications. Applications access Web services via ubiquitous Web protocols and data formats such as HTTP, XML, and SOAP, with no need to worry about how each Web service is implemented.

   Ορισμός από Microsoft.

Μεταφράζοντας τους ορισμούς:

1 Η υπηρεσία Ιστού είναι ένα σύστημα λογισμικού με σκοπό να υποστηρίξει τη διαλειτουργικότητα μηχανής προς μηχανή μέσα σε ένα δίκτυο. Είναι μια διεπαφή που περιγράφεται με ένα επεξεργάσιμο από μηχανές σχήμα (συγκεκριμένα WSDL). Άλλα συστήματα αλληλεπιδρούν με την υπηρεσία Ιστού με έναν τρόπο που ορίζεται από την περιγραφή του χρησιμοποιώντας τα SOAP-μηνύματα, μεταφέρονται χρησιμοποιώντας το HTTP μαζί με XML serialization και σε συνδυασμό με άλλα πρότυπα που σχετίζονται με τον ιστό.

2 Οι υπηρεσίες Ιστού είναι βασισμένες στον Ιστό εφαρμογές που χρησιμοποιούν τα ανοικτά πρότυπα βασισμένα σε xML και τα πρωτόκολλα μεταφορών για να ανταλλάξουν στοιχεία με τους πελάτες.

3 Οι υπηρεσίες Ιστού επεκτείνουν την υποδομή του World Wide Web για να παρέχουν τα μέσα στο λογισμικό για να το συνδέσουν με άλλες εφαρμογές λογισμικού. Οι εφαρμογές έχουν πρόσβαση στις υπηρεσίες Ιστού μέσω των πανταχού παρόντων πρωτοκόλλων του ιστού και μορφών δεδομένων όπως το HTTP, η XML, και το SOAP, χωρίς την ανάγκη να ανησυχήσουν για το πώς κάθε υπηρεσία Ιστού υλοποιείται.

Τι καθιστά όμως τις υπηρεσίες ιστού τόσο σημαντικές; Αδιαμφισβήτητα η πληθώρα των πλεονεκτημάτων τους. Ένα από τα αρχικά πλεονεκτήματα της αρχιτεκτονικής των XML υπηρεσιών Ιστού είναι ότι επιτρέπει στα προγράμματα που γράφονται σε διαφορετικές γλώσσες για τις διαφορετικές πλατφόρμες να επικοινωνήσει το ένα με το άλλο με έναν βασισμένο σε πρότυπα τρόπο. Υπάρχει η πιθανότητα κάποιος να διερωτηθεί γιατί να είναι αποτελεσματικότερο από άλλες τεχνολογίες όπως η CORBA και η DCE που σχεδιαστήκαν για παρόμοια χρήση. Η απάντηση είναι το SOAP το οποίο είναι σημαντικά λιγότερο σύνθετο από τις προηγούμενες προσεγγίσεις, έτσι το εμπόδιο στην είσοδο για μια εφαρμογή SOAP είναι σημαντικά χαμηλότερο.

Επίσης ένα άλλο σημαντικό πλεονέκτημα που οι XML υπηρεσίες Ιστού σε σχέση με τις προηγούμενες τεχνολογίες είναι ότι λειτουργούν με τα τυποποιημένα πρωτόκολλα ιστού όπως η XML, το HTTP και το TCP/IP. Ένας σημαντικός αριθμός επιχειρήσεων έχει ήδη μια υποδομή Ιστού, και ανθρώπους με τη γνώση και την εμπειρία να τη διατηρήσουν , ωστόσο πάλι, το κόστος της εισόδου για τις XML υπηρεσίες Ιστού είναι σημαντικά λιγότερο από τις προηγούμενες τεχνολογίες.

Επιπρόσθετα οι πρώτες XML υπηρεσίες Ιστού έτειναν να είναι πηγές πληροφοριών που θα μπορούσαν εύκολα να ενσωματωθούν σε εφαρμογές όπως καιρικές προβλέψεις, αθλητικά αποτελέσματα κ.λπ. Είναι εύκολο να φανταστούμε μια καινούρια κατηγορία εφαρμογών που θα μπορούσε να δημιουργηθεί για να αναλύσει τις πληροφορίες που παρουσιάζονται με ποικίλους τρόπους , για παράδειγμα, ένας υπολογισμός με λογιστικό φύλλο (spreadsheet) Microsoft® Excel που να μπορεί να συνοψίζει ολόκληρη την εικόνα για τα οικονομικά αποθέματα, τα 401K, τους τραπεζικούς λογαριασμούς, τα δάνεια , κ.λπ. . Εάν αυτές οι πληροφορίες είναι διαθέσιμες μέσω των XML υπηρεσιών Ιστού, το Excel μπορεί να το ενημερώσει συνεχώς. Μερικές από αυτές τις πληροφορίες θα είναι ελεύθερες και μερικές θα απαιτούν μια συνδρομή στην υπηρεσία. Το μεγαλύτερο μέρος αυτών των πληροφοριών είναι διαθέσιμο τώρα στον Ιστό, αλλά οι XML υπηρεσίες Ιστού θα καταστήσουν την προγραμματική πρόσβαση σε αυτό ευκολότερα και πιο αξιόπιστα.

Ακόμα η προβολή των υπαρχουσών εφαρμογών ως XML υπηρεσίες Ιστού θα επιτρέψει στους χρήστες για να χτίσουν νέες, ισχυρότερες εφαρμογές που χρησιμοποιούν τις XML υπηρεσίες Ιστού ως δομικές μονάδες. Παραδείγματος χάριν, ένας χρήστης για να αναπτύξει μια εφαρμογή αγοράς πρέπει να λάβει αυτόματα τις πληροφορίες για τις τιμές από ποικίλους προμηθευτές, να επιτρέψει στο χρήστη για να επιλέξει έναν προμηθευτή, να υποβάλει τη διαταγή και να ακολουθήσει έπειτα την αποστολή έως ότου παραλαμβάνεται. Η εφαρμογή προμηθευτών, εκτός από την έκθεση των υπηρεσιών της στον Ιστό, στη συνέχεια θα μπορεί να χρησιμοποιήσει τις XML υπηρεσίες Ιστού για να ελέγξει την πίστωση του πελάτη, να χρεώσει στον λογαριασμό του πελάτη και οργανώσει την αποστολή με μια εταιρία που ασχολείται με αποστολές.

Συνοψίζοντας μπορούμε να αναφέρουμε μερικά πλεονεκτήματα ακόμη για τις XML υπηρεσίες ιστού :

- Έκθεση της υπάρχουσας λειτουργίας προς το δίκτυο.
- Διαλειτουργικότητα μεταξύ διαφορετικών εφαρμογών.
- Τυποποιημένο πρωτόκολλο.
- Χαμηλότερο κόστος της επικοινωνίας
- Ευκολότερη να επαναχρησιμοποίηση των υπαρχουσών υπηρεσιών.
- Ευκολότερη να διανομή πληροφοριών σε περισσότερους καταναλωτές

- Γρήγορη ανάπτυξη

Παρόλο τα πολλά πλεονεκτήματα που μας παρέχει η συγκεκριμένη τεχνολογία έχει και σημαντικά μειονεκτήματα.

- Η χρήση XML ένα σχήμα μεταφοράς δεδομένων σημαίνει ότι τα μηνύματά μας είναι μεγάλα: Οι ετικέττες XML λαμβάνουν πολύ διάστημα, και αυτό τοποθετεί ένα φορτίο σε μας για τη δημιουργία και την ερμηνεία, καθώς επίσης και τη μεταφορά των μηνυμάτων μας.
- Η χρήση απομακρυσμένων υπολογιστών για να κάνουν την επεξεργασία μας βάζει στο έλεος του Διαδικτύου, και δημιουργεί πολλά πιθανά σημεία αποτυχίας μεταξύ του κεντρικού υπολογιστή και της υπηρεσίας του δικτύου μας.
- Λίγες επιχειρήσεις παρέχουν αυτήν την περίοδο υπηρεσίες Ιστού, και λίγες επιχειρήσεις τις χρησιμοποιούν. Θα πάρει το χρόνο για τις υπηρεσίες Ιστού να πολλαπλασιαστούν και να εξεταστούν κατάλληλα από την κοινότητα υπεύθυνων για την ανάπτυξη.
- Τα πρότυπα χορήγησης αδειών και χρέωσης για τις υπηρεσίες Ιστού πρέπει να είναι αποδεκτά από τους υπεύθυνους για την ανάπτυξη. Έτσι με λίγες υπηρεσίες Ιστού εκεί έξω, και τις περισσότερες επιχειρήσεις που προσπαθούν σκληρά να κάνουν καλή εντύπωση με την κράτηση του κόστους χαμηλά και των όρων λογικών, είναι μια στιγμή προτού το κόστος των υπηρεσιών να γίνει σαφές.
- Έχουν χαμηλή απόδοση σε σχέση με άλλες τεχνολογίες όπως οι RPC.
- Δεν υποστηρίζουν προηγμένα χαρακτηριστικά η τουλάχιστον δεν είναι ακόμα προτυποποιημένη η υποστήριξη.

1.3 Ιστορική αναδρομή

1999-2001:

Τον Ιούνιο του 2001 η ομάδα Gartner τεκμηρίωσε μια υπόδειξη ως προς το χρόνο για υιοθέτηση των υπηρεσιών Ιστού από το 2001 έως το 2005. Υπέθεσαν ότι το 2001 θα έβλεπαν πολλά εργαλεία ανάπτυξης για τις υπηρεσίες Ιστού παραδοθέντα. Με τα Beta και Final Release tools από τη Microsoft, τη IBM,τη Sun, τη Software AG, τη Oracle και πολλές άλλες, αυτός ο πολλαπλασιασμός των εργαλείων υπηρεσιών Ιστού ήταν σε εξέλιξη.

2002-2004:

Το Gartner θεώρησε ότι το 2002 θα δει τις επιχειρησιακές υπηρεσίες Ιστού να αρχίζουν να εμφανίζονται σε μεγάλους αριθμούς, επίσης με την Business-to-Consumer (B2C) υπήρχε πρόσβαση προσανατολισμένες ως προς τον καταναλωτή υπηρεσίες Ιστού. Ένα παράδειγμα τέτοιων B2C υπηρεσιών Ιστού είναι το My Services από τη Microsoft (με κωδικό Hailstorm). Οι My Services σχεδιάζεται να δοθούν στο εμπόριο το 2002 σε συνδυασμό με τη Microsoft .NET.

Το 2003 έχουμε την υιοθέτηση των ληξιαρχείων UDDI (καθολικές Universal Description, Discovery and Integration). Αυτό προβάλλεται για να υποστηρίξει μαζί με τα ιδιωτικά ληξιαρχεία τις ιδιωτικές ανταλλαγές. Τα δημόσια ληξιαρχεία θα προκύψουν για να υποστηρίξουν τις δημόσιες ανταλλαγές, με την κυβερνητική χρήση των υπηρεσιών Ιστού που επιταχύνουν επίσης αισθητά.

Μερικά από αυτά τα ληξιαρχεία μπορούν να προσφέρουν τη ελεύθερη πρόσβαση στις υπηρεσίες Ιστού, αλλά πιό πολύ αναμένονται για να διατεθούν με βάση την καταβολή χρημάτων.

Το 2004, σύμφωνα με την ομάδα Gartner, θα δει την πρώιμημορφή των επιχειρησιακών προτύπων που είναι βασισμένα σε υπηρεσίες ιστού, με τα ιδιωτικά ληξιαρχεία να κυριαρχούν ακόμα. Τα νέα πρότυπα εισόδημα-παραγωγής και οι ευκαιρίες καναλιών θα γίνουν κοινά. Το Gartner προβλέπει ότι μέχρι το 2004, το 40% των συναλλαγών των οικονομικών υπηρεσιών θα ελέγχουν τα πρότυπα υπηρεσιών Ιστού, με ποσοστό 35% των σε απευθείας σύνδεση κυβερνητικών υπηρεσιών που παραδίδονται ως υπηρεσίες Ιστού.

2005 και μετά

Το 2005 θα δούμε τα δημόσια ληξιαρχεία UDDI να κερδίζουν την προσοχή δεδομένου ότι οι δημόσιες B2B ανταλλαγές αρχίζουν να επανεμφανίζονται μετά από μια σχετική παύση από το 2001 έως το 2002 κατά τη διάρκεια της μείωσης dot-COM. Οι δυναμικές υπηρεσίες Ιστού θα κερδίσουν επίσης περισσότερη προσοχή. Θα αναφερθούμε σε αυτήν την περίοδο (2005 και μετά) ως φάση 3 της εξέλιξης των υπηρεσιών Ιστού. Για τις υπηρεσίες Ιστού που παραδίδουν γρήγορα, η συνεχής ολοκλήρωση των συνεργατικών επιχειρήσεων σε μια επιχειρηματική κλίμακα λαμβάνει χώρο κατά τη διάρκεια της φάσης 2, κατά την οποία διάφορα ζητήματα θα πρέπει να εξεταστούν.

Αυτά περιλαμβάνουν: Ποιότητα εξυπηρέτησης , αξιοπιστία δικτύων ,αποκατάσταση συναλλαγής , πραγματικό χρόνο, ασφάλεια και μηχανισμοί τιμολόγησης. Τα δίκτυα υπηρεσιών Ιστού μεταξύ των φορέων παροχής υπηρεσιών και των αιτούντων των υπηρεσιών πρέπει να χειριστούν την επικύρωση του τελικού σημείου μεταξύ των συνεργατών, και πρέπει να παρέχουν την ασφάλεια, στοιχεία κρυπτογράφηση και non-repudiation των συναλλαγών. Οι προμηθευτές δικτύων υπηρεσιών Ιστού θα πρέπει επίσης να προσφέρουν και το σύγχρονο και ασύγχρονο μήνυμα. Το τελευταίο επιτρέπει σε έναν αιτούντα υπηρεσιών πελατών για να εκτελέσει άλλους στόχους περιμένοντας μια απάντηση από έναν φορέα παροχής υπηρεσιών. Ο δίκτυο-ποιοτικός έλεγχος, η διαχείριση λάθους και τα σχέδια στοιχείο-συμπίεσης θα βοηθήσουν να βελτιώσουν την εξελιξιμότητα και την αξιοπιστία δικτύων.

Οι προμηθευτές δικτύων των υπηρεσιών Ιστού προκύπτουν για να αντιμετωπίσουν αυτά τα ζητήματα, όπως οι Grand Central, Flamenco Networks, και Kenamea. Το Grand Central χρησιμοποιεί μια hub τοπολογία , ενώ το Flamenco Networks χρησιμοποιεί ένα πληρεξούσιο κεντρικών υπολογιστών για μια πολυσημειακή προσέγγιση δικτύων. Και οι δύο εστιάζουν στη σταθερότητα συναλλαγών, και σ τον έλεγχο δικτύων με βάση το πρότυπο server-to-server. Το Kenamea ειδικεύεται στην τελευταία τάση των δικτύων σε μια ευρεία σειρά τύπων συσκευών, όπως για τις αλυσίδες εφοδιασμού.

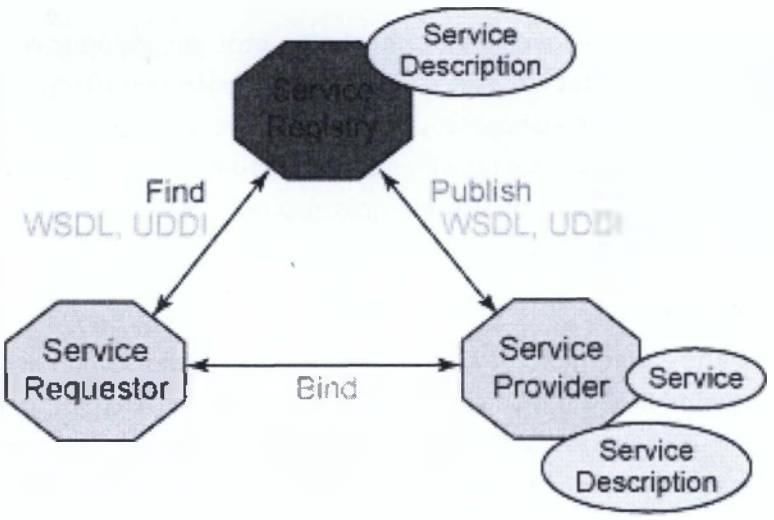
Στη φάση 3 της εξέλιξης των υπηρεσιών Ιστού, οι οργανώσεις θα αλλάξουν όχι μόνο τις επιχειρησιακές διαδικασίες τους, αλλά και τα επιχειρησιακά πρότυπά τους καθώς κινούν προς τη συνεργασία σε πραγματικό χρόνο και την ολοκλήρωση των διαδικασιών ανάμεσα στις επιχειρήσεις. Ενώ η φάση 1 και η φάση 2 εξετάζουν την εμφάνιση των υπηρεσιών Ιστού που κλειδώνονται προηγουμένως μακριά στα συστήματα ρευμάτων και κληρονομιών, η φάση 3 θα δει την εμφάνιση των νέων προϊόντων λογισμικού και των συστημάτων που σχεδιάζονται και αναπτύσσονται εξαρχής για να παραδοθούν ως υπηρεσίες ιστού.

Αυτά θα χρησιμοποιηθούν από τις οργανώσεις για να βρούν τους συνέταιρους δυναμικά, ή για να χρησιμοποιήσουν τους μακρινούς πόρους, και θα επιτρέψουν σε εκείνες τις οργανώσεις για να προσαρμοστούν γρήγορα στην αλλαγή. Σαν προηγούμενο ζήτημα των TEN που συζητείται, η υιοθέτηση των υπηρεσιών Ιστού από μια τεχνική προοπτική δεν είναι σύνθετη. Οι XML ορισμοί

κάθε υπηρεσίας Ιστού θα παραχθούν αυτόματα: για το SOAP, το WSDL και το UDDI. Παραδείγματος χάριν, το Microsoft Visual Studio.NET παρέχει την αυτόματη παραγωγή για Visual Basic.NET, C++.NET and C#.NET που αφορούν τις λειτουργικές κλήσεις. Το WebSphere Studio Application Developer της IBM παρέχει την αυτόματη παραγωγή των μεθόδων για J2EE (Java 2 Enterprise Edition).

1.4 Το μοντέλο μίας υπηρεσίας ιστού

Η αρχιτεκτονική υπηρεσιών Ιστού είναι βασισμένη στις αλληλεπιδράσεις μεταξύ τριών ρόλων: φορέας παροχής υπηρεσιών(service Provider), ληξιαρχείο υπηρεσιών(service registry) και πελάτης υπηρεσιών(service requestor). Οι αλληλεπιδράσεις περιλαμβάνουν δημοσιεύουν, βρίσκουν και δεσμεύουν τις διαδικασίες. Μαζί, αυτοί οι ρόλοι ενεργούν επάνω στα αντικείμενα των υπηρεσιών Ιστού: στην ενότητα του λογισμικού υπηρεσιών και στην περιγραφή της. Σε ένα χαρακτηριστικό σενάριο είναι ένας φορέας παροχής υπηρεσιών φιλοξενεί μια δίκτυο-προσιτή ενότητα λογισμικού (μια εφαρμογή υπηρεσίας Ιστού). Ο φορέας παροχής υπηρεσιών καθορίζει μια περιγραφή υπηρεσιών για την υπηρεσία Ιστού και την δημοσιεύει σε έναν αιτούντα υπηρεσιών ή ένα ληξιαρχείο υπηρεσιών. Ο πελάτης υπηρεσιών χρησιμοποιεί μια λειτουργία για να ανακτήσει την περιγραφή τοπικά ή από το ληξιαρχείο και χρησιμοποιεί την περιγραφή υπηρεσιών για να δεσμεύσει το φορέα παροχής υπηρεσιών ή να επικαλεσθεί ή να αλληλεπιδράσει με την εφαρμογή των υπηρεσιών Ιστού. Ο φορέας παροχής υπηρεσιών και οι ρόλοι των αιτούντων υπηρεσιών είναι λογικά κατασκευάσματα και μια υπηρεσία μπορεί να εκθέσει τα χαρακτηριστικά και από τα δύο. Το σχήμα 1 επεξηγεί αυτές τις διαδικασίες, τα συστατικά που παρέχουν τις και τις αλληλεπιδράσεις τους.



Σχήμα1. 1 Ρόλοι υπηρεσιών ιστού

Ρόλοι σε μια αρχιτεκτονική υπηρεσιών Ιστού

- Φορέας παροχής υπηρεσιών. Από μια επιχειρησιακή προοπτική, αυτό είναι ο ιδιοκτήτης της υπηρεσίας. Από μια αρχιτεκτονική προοπτική, αυτό είναι η πλατφόρμα όπου οι hosts έχουν πρόσβαση στην υπηρεσία.

- Αιτών της υπηρεσίας. Από μια επιχειρησιακή προοπτική, αυτό είναι η επιχείρηση που απαιτεί ορισμένες λειτουργίες για να εκτελεστούν. Από μια αρχιτεκτονική προοπτική, αυτό είναι η εφαρμογή που ψάχνει και διαδοχικά καλεί ή αρχίζει να αλληλεπιδρά με μια υπηρεσία. Ο ρόλος του αιτούντος μιας υπηρεσίας μπορεί να διαδραματιστεί από έναν browser που οδηγείται από ένα πρόσωπο ή ένα πρόγραμμα χωρίς διεπαφή χρήστη, παραδείγματος χάριν μια άλλη υπηρεσία Ιστού.

- Ληξιαρχείο υπηρεσιών. Αυτό είναι ένα εξερευνήσιμο ληξιαρχείο περιγραφών υπηρεσιών όπου οι φορείς παροχής υπηρεσιών δημοσιεύουν τις περιγραφές των υπηρεσιών τους. Οι αιτούντες υπηρεσιών βρίσκουν τις υπηρεσίες και λαμβάνουν τις δεσμευτικές πληροφορίες (που βρίσκονται στις περιγραφές υπηρεσιών) για τις υπηρεσίες κατά τη διάρκεια της ανάπτυξης για τη στατική σύνδεση ή κατά τη διάρκεια της εκτέλεσης για τη δυναμική σύνδεση. Για τη στατικά συνδεδεμένη υπηρεσία οι αιτούντες και το ληξιαρχείο υπηρεσιών είναι ένας προαιρετικός ρόλος στην αρχιτεκτονική, επειδή ένας φορέας παροχής υπηρεσιών μπορεί να στείλει την περιγραφή άμεσα στους αιτούντες υπηρεσιών. Επιπλέον, οι αιτούντες μιας υπηρεσίας μπορούν να λάβουν μια περιγραφή υπηρεσιών από άλλες πηγές εκτός από ένα ληξιαρχείο υπηρεσιών, όπως ένα τοπικό αρχείο, μια περιοχή FTP, μια ιστοσελίδα, το Advertisement and Discovery of Services (ADS) ή από Discovery of Web Services (DISCO).

1.5 Χαρακτηριστικά συμπεριφοράς υπηρεσιών ιστού

Έχοντας είδη αναλύσει το μοντέλο μιας υπηρεσίας ιστού καταφέραμε να κατανοήσουμε τους ρόλους που μπορεί να έχει μια υπηρεσία. Ανάλογα με το ρόλο που κατέχει συμπεριφέρεται και αντίστοιχα. Όμως πρέπει να επικεντρωθούμε στο γεγονός ότι ανεξάρτητα με το αν μια υπηρεσία είναι service Provider, service registry ή service requestor έχει κοινά χαρακτηριστικά συμπεριφοράς με όλες τις άλλες. Τα κοινά χαρακτηρίστηκα συμπεριφοράς που παρουσιάζουν όλες οι υπηρεσίες ιστού είναι τα ακόλουθα:

- Βασισμένες σε XML
  Οι υπηρεσίες Ιστού χρησιμοποιούν XML στα στρώματα αντιπροσώπευσης και μεταφοράς στοιχείων. Η χρήση XML αποβάλλει οποιοδήποτε σύστημα δικτύωσης, ή σύνδεσης πλατφορμών. Έτσι οι βασισμένες στις υπηρεσίες, εφαρμογές Ιστού είναι ιδιαίτερα διαλειτουργικές εφαρμογές στο επίπεδο των πυρήνων τους.

- Loosely coupled (Χαλαρά συνδεμένος)
  Ένας καταναλωτής μιας υπηρεσίας Ιστού δεν είναι δεμένος σε εκείνη την υπηρεσία Ιστού άμεσα. Η διεπαφή υπηρεσιών Ιστού μπορεί να αλλάξει με την πάροδο του χρόνου χωρίς συμβιβασμό της δυνατότητας του πελάτη να αλληλεπιδρά με την υπηρεσία. Ένα στενά συνδεμένο σύστημα υπονοεί ότι η λογική πελατών και η λογική κεντρικών υπολογιστών είναι στενά συνδεδεμένες, που υπονοεί ότι εάν υπάρχουν αλλαγές σε μια διεπαφή, πρέπει να ενημερωθεί και η άλλη. Η υιοθέτηση μιας αόριστα συνδεμένης αρχιτεκτονικής τείνει να καταστήσει τα συστήματα λογισμικού πιό εύχρηστα και επιτρέπει την απλούστερη επικοινωνία μεταξύ διαφορετικών συστημάτων.

- Coarse-grained

  Οιτεχνολογίες που είναι αντικειμενοστραφής όπως για παράδειγμα η java αποκαλύπτουν τις υπηρεσίες τους μέσω μεμονωμένων μεθόδων. Μία μεμονωμένη μέθοδος είναι πολύ απλά μία λειτουργία που παρέχει οποιαδήποτε χρήσιμη ικανότητα σε ένα εταιρικό επίπεδο. Η δημιουργία ενός προγράμματος σε java απαιτεί τη δημιουργία διαφόρων fine-grained μεθόδων που συνδυάζονται σε μία coarse-grainedυπηρεσία που χρησιμοποιείται είτε από κάποιο πελάτη είτε από κάποια άλλη υπηρεσία. Οι επιχειρήσεις και τα interface που αποκαλύπτουν πρέπει να είναι coarse-grained. Η τεχνολογία των υπηρεσιών διαδικτύου παρέχει ένα τρόπο για να ορίζει τις coarse-grained υπηρεσίες που έχουν πρόσβαση με το σωστό τρόπο στην επιχειρησιακή λογική.

- Δυνατότητα να είναι σύγχρονος ή ασύγχρονος

  Το Synchronicity αναφέρεται στη σύνδεση του πελάτη με την εκτέλεση της υπηρεσίας. Στις σύγχρονες διαδοχικές κλήσεις, ο πελάτης εμποδίζεται και περιμένει την υπηρεσία να ολοκληρώσει τη λειτουργία του πριν συνεχίσει. Οι ασύγχρονες διαδικασίες επιτρέπουν σε έναν πελάτη να κάνει κλήση μιας υπηρεσία και να εκτελέσει έπειτα άλλες λειτουργίες. Οι ασύγχρονοι πελάτες ανακτούν το αποτέλεσμά τους σε μια πιό πρόσφατη συγκεκριμένη στιγμή, ενώ οι σύγχρονοι πελάτες λαμβάνουν το αποτέλεσμά τους όταν ολοκληρώσει η υπηρεσία. Η ασύγχρονη ικανότητα είναι ένας παράγοντας κλειδί στη διευκόλυνση των αόριστα συνδεμένων συστημάτων.

- Υποστήριξη απομακρυσμένων κλήσεων διαδικασιών (RPCs)

  Οι υπηρεσίες Ιστού επιτρέπουν στους πελάτες για να επικαλεσθούν τις διαδικασίες, τις λειτουργίες, και τις μεθόδους στα μακρινά αντικείμενα χρησιμοποιώντας ένα βασισμένο σε XML πρωτόκολλο. Οι απομακρυσμένες διαδικασίες εκθέτουν τις παραμέτρους input και output που μια υπηρεσία Ιστού πρέπει να υποστηρίξει. Η ανάπτυξη συστατικών μέσω της επιχείρησης JavaBeans (EJBs) και .NET γίνεται όλο και περισσότερο ένα μέρος των αρχιτεκτονικών και των επιχειρηματικών επεκτάσεων τα τελευταία χρόνια. Και οι δύο τεχνολογίες είναι προσιτές και διανέμονται και μέσω ποικίλων μηχανισμών RPC. Μια υπηρεσία Ιστού υποστηρίζει το RPC με την παροχή των υπηρεσιών της, που είναι ισοδύναμες με εκείνους ενός παραδοσιακού συστατικού, ή με τη μετάφραση των εισερχόμενων διαδοχικών κλήσεων σε μια κληση ενός EJB ή ενός .NET συστατικού.

- Υποστήριξη Ανταλλαγής εγγράφων

  Ένα από τα βασικά πλεονεκτήματα της XML είναι ο γενικός τρόπος που αναπαριστά όχι μόνο τα στοιχεία, αλλά και σύνθετα έγγραφα. Αυτά τα έγγραφα μπορούν να είναι απλά, όπως η αντιπροσώπευση μιας τρέχουσας διεύθυνσης, ή μπορούν να είναι σύνθετα, όπως η αντιπροσώπευση ενός ολόκληρου βιβλίου. Οι υπηρεσίες Ιστού υποστηρίζουν τη διαφανή ανταλλαγή των εγγράφων για να διευκολύνουν την επιχειρησιακή ολοκλήρωση.

1.6 Σωρός πρωτοκόλλων

Σε αυτό το σημείο θα αναφέρουμε τα μέρη που απαρτίζουν μια στοίβα μιας υπηρεσίας ιστού τα στοιχεία που την αποτελούν αλλά και τα στρώματα που τη συνθέτουν. Θα ξεκινήσουμε με το κατώτερο στρώμα αυτό της μεταφοράς και καταλήξουμε στο ανώτερο το οποίο είναι το στρώμα ροής υπηρεσιών.

Στρώμα μεταφορών (Transport layer)

Όπως μπορείτε να δείτε στο σχήμα 1.2, το στρώμα μεταφορών είναι το θεμέλιο του σωρού υπηρεσιών Ιστού. Οι υπηρεσίες Ιστού πρέπει να κληθούν από έναν πελάτη υπηρεσιών έτσι ώστε να μπορούν να είναι προσιτές σε ένα δίκτυο. Οι υπηρεσίες Ιστού που είναι δημόσια διαθέσιμες τρέχουν μέσω του Διαδικτύου. Μόνο οι εξουσιοδοτημένοι χρήστες μέσα σε μια εσωτερική οργάνωση μπορούν να δουν τις ενδοδίκτυο-διαθέσιμες υπηρεσίες Ιστού, ενώ οι αναρμόδιοι χρήστες στον εξωτερικό κόσμο δεν μπορούν.

Tool	Layer	Business Issues		
WSFL	Service Flow	Security	Management	Quality of Service
Static→UDDI	Service Discovery			
Direct→UDDI	Service Publication			
WSDL	Service Description -Service Implementation -Service Interface Secure Messaging			
SOAP	XML-Based Messaging			
HTTP, FTP, SMTP, MQ RMI over IIOP	Transport			

Σχήμα1. 2 Σωρός υπηρεσιών ιστού.

Το πρωτόκολλο Διαδικτύου που μπορεί να υποστηριχθεί από αυτόν τον σωρό είναι το HTTP. Οι περιοχές του ενδοδικτύου χρησιμοποιούν τις υποδομές κλήσης υλικολογισμικού, όπως MQSeries της IBM, και CORBA (Common Object Request Broker Architecture). Το τελευταίο στηρίζεται σε ένα πρωτόκολλο αποκαλούμενο Inter-ORB Protocol (IIOP) για τα απομακρυσμένα αντικείμενα.

Στρώμα μηνύματος βασισμένο σε XML (XML-based messaging layer)

Σε αυτό το στρώμα, το SOAP είναι το πρωτόκολλο μηνύματος για την XML. Χτίζεται επάνω στο χαμηλότερο στρώμα (μεταφοράς) και αυτό σημαίνει ότι το SOAP χρησιμοποιείται μεμονωμένα ή σε σχέση με οποιαδήποτε πρωτόκολλο μεταφοράς. Όλα τα μηνύματα SOAP υποστηρίζουν το publish, bind, και find(εντολές) των διαδικασιών της αρχιτεκτονικής υπηρεσιών Ιστού. Το SOAP περιλαμβάνει τρία μέρη: ένας φάκελος για να περιγράψει το περιεχόμενο ενός μηνύματος, ένα

σύνολο κανόνων κωδικοποίησης, και έναν μηχανισμό για τις μακρινές κλήσεις διαδικασίας (RPCs) και τις απαντήσεις. Θα γίνει εκτενέστερη αναφορά στη συνέχεια.

Η IBM, η Microsoft, και άλλες υπέβαλαν το SOAP στο W3C ως βάση της ομάδας εργασίας πρωτοκόλλου XML. Όταν το W3C απελευθερώσει πρότυπα σχεδίων για το πρωτόκολλο XML, ο σωρός αρχιτεκτονικής υπηρεσιών Ιστού θα μεταναστεύσει από το SOAP στο πρωτόκολλο XML.

Στρώμα περιγραφής υπηρεσιών (Service description layer)

Η περιγραφή υπηρεσιών παρέχει τα μέσα για να κληθούν οι υπηρεσίες Ιστού. Το WSDL είναι το βασικό πρότυπο για τον καθορισμό, μαζί με το σχήμα XML, των εφαρμογών και των διεπαφών των υπηρεσιών. Αυτό σημαίνει ότι το WSDL διαιρεί μια περιγραφή υπηρεσιών σε δύο μέρη: την εφαρμογή και τη διεπαφή υπηρεσιών. Πρέπει να δημιουργηθεί μια διεπαφή υπηρεσιών προτού να είναι δυνατό να εφαρμοστεί WSDL.

Στρώμα δημοσιεύσεων υπηρεσιών (Service publication layer)

Χρειάζονται και άλλα έγγραφα περιγραφής υπηρεσιών για να συμπληρωθεί το WSDL. Μπορούν να χρησιμοποιηθούν οι δομές δεδομένων UDDI για να περιγράψουν το επιχειρησιακό πλαίσιο, παραδείγματος χάριν. Μπορούν να καταστήσουν ένα έγγραφο WSDL διαθέσιμο σε έναν πελάτη υπηρεσιών σε οποιοδήποτε στάδιο του κύκλου ζωής του πελάτη υπηρεσιών. Όταν αυτό συμβαίνει, ο σωρός κινείται επάνω προς το επόμενο στρώμα, τη δημοσίευση υπηρεσιών. Σε αυτό το στρώμα, για παράδειγμα, ο φορέας παροχής υπηρεσιών μπορεί να στείλει ένα έγγραφο WSDL άμεσα σε έναν πελάτη υπηρεσιών μέσω του ηλεκτρονικού ταχυδρομείου. Αυτή η δράση είναι γνωστή ως άμεση δημοσίευση. Ο φορέας παροχής υπηρεσιών μπορεί επίσης να επιλέξει να δημοσιεύσει το έγγραφο WSDL σε ένα οικοδεσπότης ο οποίος μπορεί να είναι ένα τοπικό ληξιαρχείο WSDL, ή σε ένα δημόσιο ή ιδιωτικό ληξιαρχείο UDDI. Το WSCA της IBM διευκρινίζει πώς οι ορισμοί για δύο συστατικά WSDL μπορούν να προέλθουν από τις καταχωρήσεις UDDI.

Στρώμα ανακαλύψεων υπηρεσιών (Service discovery layer)

Η ανακάλυψη υπηρεσιών στηρίζεται στη δημοσίευση υπηρεσιών . Εάν μια υπηρεσία Ιστού δεν είναι δημοσιευμένη ή δεν μπορεί να δημοσιευθεί τότε δεν μπορεί να βρεθεί ή να ανακαλυφθεί. Κατά το χρόνο εκτέλεσης ο πελάτης υπηρεσιών μπορεί να καταστήσει την περιγραφή υπηρεσιών διαθέσιμη σε μια εφαρμογή . Ο πελάτης υπηρεσιών, παραδείγματος χάριν, μπορεί να ανακτήσει ένα έγγραφο WSDL από ένα τοπικό αρχείο που λαμβάνεται μέσω μίας άμεσης δημοσίευσης. Αυτή η δράση είναι γνωστή ως στατική ανακάλυψη. Η υπηρεσία μπορεί επίσης να ανακαλυφθεί κατά το σχέδιο ή το χρόνο εκτέλεσης χρησιμοποιώντας είτε ένα τοπικό ληξιαρχείο WSDL είτε ένα δημόσιο η ιδιωτικό ληξιαρχείο UDDI.

Στρώμα ροής υπηρεσιών (Service flow layer)

Η Web Services Flow Language (WSFL) είναι το πρότυπο για το στρώμα ροής υπηρεσιών στην κορυφή του σωρού. Διαφέρει από άλλα πρότυπα στο σωρό δεδομένου ότι εξετάζει τη διαμόρφωση και τις workflows της επιχειρησιακής διαδικασίας. Η WSFL χρησιμοποιείται για να περιγράψει πώς οι υπηρεσίες Ιστού πρόκειται να αλληλεπιδράσουν στις workflows και πώς

πρόκειται να αποδώσουν στις επικοινωνίες ή τις συνεργασίες στο πρότυπο υπηρεσία-υπηρεσία. Αυτό σημαίνει ότι οι υπηρεσίες Ιστού είναι συστατικά, ή μπορεί να ενορχηστρωθούν δυναμικά μέσα σε ροή εργασίας (workflows).

Είναι αναγκαίο ένα εργαλείο, όπως το IBM MQSeries Workflow (τώρα αποκαλούμενο WebSphere Process Manager), για να καθοριστούν οι επιχειρησιακές διαδικασίες ως σειρά δραστηριοτήτων, και για να ποικίλει η ακολουθία αυτών των δραστηριοτήτων καθώς οι απαιτήσεις για τις επιχειρησιακές διαδικασίες αλλάζουν.

1.7 Τα συστατικά των υπηρεσιών Ιστού

Τα συστατικά των υπηρεσιών ιστού είναι η XML, το SOAP, το WSDL και το UDDI. Σε περίπτωση που μία υπηρεσία ιστού υποστηρίζει το αρχιτεκτονικό ύφος REST αντί για το SOAP θα βασίζεται και στο WADL αντί για το WSDL. Παρακάτω γίνεται μια μη εκτενής αναφορά σε αυτά τα συστατικά (θα αναλυθούν στο κεφάλαιο 2).

XML: Η XML παρέχει μια βασισμένη σε πρότυπα μέθοδο για να περιγράφει τα στοιχεία. Χρησιμοποιείται εκτενώς στην δημιουργία και την κατανάλωση των υπηρεσιών Ιστού. Έχει τη δυνατότητα να περιγράψει την πληροφορία που είναι ιδιαίτερα διαλειτουργικό μεταξύ πολλών διαφορετικών συστημάτων στο διαδίκτυο. Χρησιμοποιώντας τα βασικά στοιχεία της XML μπορούμε να καθορίσουμε τους απλούς και σύνθετους τύπους και τις σχέσεις των στοιχείων. Η XML προωθεί τη δυνατότητα των υπηρεσιών Ιστού να επικοινωνήσουν για τα δεδομένα τους αποδοτικά και αποτελεσματικά. Εξασφαλίζει μια συνεπή και ακριβή ερμηνεία των στοιχείων όταν η υπηρεσία και ο καταναλωτής χρησιμοποιούν διαφορετικές πλατφόρμες.

SOAP: Αν και δημιουργήθηκε αρχικά ως τεχνολογία για να γεφυρώσει το χάσμα μεταξύ ανόμοιων πλατφόρμων επικοινωνίας βασισμένες σε RPC, το SOAP εξελίχθηκε ευρύτατα σε υποστηριγμένο σχήμα και πρωτόκολλο μηνύματος για τη χρήση στις XML υπηρεσίες Ιστού. Ως εκ τούτου το αρκτικόλεξο SOAP συγχέεται συχνά ως προσανατολισμένη προς τις υπηρεσίες αρχιτεκτονική (ή εφαρμογή)(Service-Oriented Architecture (or Application) δηλαδή SOA) αντί του απλού πρωτοκόλλου πρόσβασης αντικειμένου(Simple Object Access Protocol). Η προδιαγραφή SOAP καθιερώνει ένα σχήμα πρότυπων μηνυμάτων που αποτελείται από ένα έγγραφο XML ικανό να υποστηρίξει RPC και των έγγραφοκεντρικά στοιχεία.

Αυτό διευκολύνει το σύγχρονο (αίτημα και απάντηση) καθώς επίσης και το ασύγχρονο (process-driven) πρότυπο ανταλλαγής στοιχείων. Με το WSDL που καθιερώνει ένα τυποποιημένο σχήμα περιγραφής endpoint για τις εφαρμογές, το έγγραφο-κεντρικό σχήμα μηνυμάτων είναι πολύ περισσότερο κοινό.

WSDL: Το WSDL 2.0 περιγράφει μια υπηρεσία Ιστού σε δύο θεμελιώδη στάδια: μια περίληψη και ένα concrete. Μέσα σε κάθε στάδιο, η περιγραφή χρησιμοποιεί διάφορα κατασκευάσματα για να προωθήσει την ικανότητα επαναχρησιμοποίησης της περιγραφής και για να χωρίσει τις ανεξάρτητες ανησυχίες σχεδίου.

Σε αφηρημένο επίπεδο, το WSDL 2.0 περιγράφει μια υπηρεσία Ιστού από την άποψη των μηνυμάτων που στέλνει και λαμβάνει. Τα μηνύματα είναι περιγεγραμμένα ανεξάρτητα από ένα συγκεκριμένο σχήμα που χρησιμοποιεί ένα σύστημα τύπων, χαρακτηριστικά το σχήμα XML.

Μια λειτουργία συνδέει ένα σχέδιο ανταλλαγής μηνυμάτων με ένα ή περισσότερα μηνύματα. Ένα σχέδιο ανταλλαγής μηνυμάτων προσδιορίζει την ακολουθία και τον αριθμό στοιχείων του συνόλου των μηνυμάτων που στέλνονται ή λαμβάνονται καθώς επίσης και ποιοι

τους στέλνουν λογικά ή παραλαμβανουν. Μια διεπαφή συγκεντρώνει τις διαδικασίες χωρίς οποιαδήποτε δέσμευση για τη μεταφορά ή το wire σχήμα.

Στο concrete επίπεδο, ένα binding διευκρινίζει τις λεπτομέρειες του σχήματος μεταφορών και wire format για μια ή περισσότερες διεπαφές. Ένα endpoint συνδέει μια διεύθυνση δικτύων με ένα binding. Και τελικά, μια υπηρεσία συγκεντρώνει τα endpoints που υλοποιούν μια κοινή διεπαφή.

UDDI: Ένα από τα θεμελιώδη συστατικά μιας προσανατολισμένης προς τις υπηρεσίες αρχιτεκτονικής είναι ένας μηχανισμός για τις περιγραφές υπηρεσιών Ιστού που ανακαλύπτονται από τους πιθανούς πελάτες. Για να θεσπιστεί αυτό το μέρος του πλαισίου υπηρεσιών Ιστού, ένας κεντρικός κατάλογος απαιτείται για τις περιγραφές υπηρεσίας. Ένας τέτοιος κατάλογος μπορεί να γίνει ένα αναπόσπαστο τμήμα μιας οργάνωσης ή μιας κοινότητας Διαδικτύου, έτσι, αυτό θεωρείται επέκταση στην υποδομή.

Για αυτό το Universal Description, Discovery και Integration γίνεται όλο και περισσότερο σημαντικό. Ένα βασικό μέρος του UDDI είναι η τυποποίηση των αρχείων των σχεδιαγραμμάτων που αποθηκεύονται μέσα σε έναν τέτοιο κατάλογο, γνωστό ως ληξιαρχείο. Ανάλογα με το για ποιους το ληξιαρχείο προορίζεται , μπορούν να δημιουργηθούν διαφορετικές εφαρμογές.

REST: Το Representational State Transfer (REST) είναι ένα αρχιτεκτονικό ύφος μεγάλης κλίμακας δικτυωμένου λογισμικού που εκμεταλλεύεται τις τεχνολογίες και τα πρωτόκολλα του World Wide Web. Το REST περιγράφει πώς τα διανεμημένα αντικείμενα στοιχείων, ή οι πόροι, μπορούν να καθοριστούν και να εξεταστούν, τονίζοντας την εύκολη ανταλλαγή των πληροφοριών και της εξελιξιμότητας.

WADL: Το Web Application Description Language (WADL) είναι μια βασισμένη σε XML μορφή αρχείου που παρέχει μια περιγραφή των βασισμένων σε HTTP εφαρμογών Ιστού αναγνώσιμη από μηχανή. Αυτές οι εφαρμογές είναι χαρακτηριστικά REST υπηρεσίες Ιστού.

Ο σκοπός του WADL είναι να επιτραπούν οι υπηρεσίες στο διαδίκτυο (ή σε οποιοδήποτε άλλο δίκτυο IP) να περιγραφεί με έναν τρόπο που μπορεί να επεξεργαστεί από τις μηχανές, για να το καταστήσει ευκολότερο να δημιουργήσει εφαρμογές ύφους Web 2.0 και να δημιουργήσει έναν δυναμικό τρόπο δημιουργίας υπηρεσιών. Πριν από αυτό, ήταν απαραίτητο να πάει σε μια υπάρχουσα υπηρεσία Ιστού, να τη μελετήσει και να γραφεί η εφαρμογή με το χέρι.Το WADL μπορεί να θεωρηθεί ως REST ισοδύναμο της έκδοσης WSDL 1.1. Η έκδοση 2.0 WSDL μπορεί να χρησιμοποιηθεί για να περιγράψει τις υπηρεσίες Ιστού REST, κατά συνέπεια είναι ανταγωνιστικό με το WADL.

1.8 Choreography και Orchestration

1.8.1 Choreography

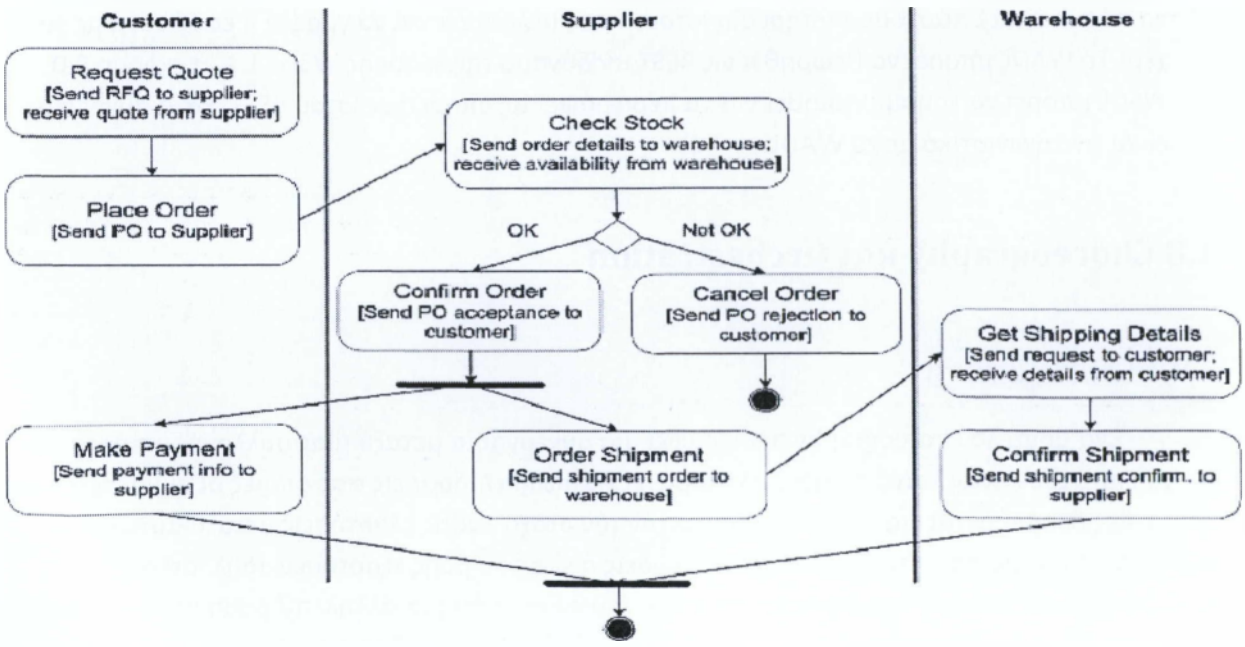
Ένα μοντέλο choreography περιγράφει μια συνεργασία μεταξύ μιας συλλογής υπηρεσιών για να επιτύχουν έναν κοινό στόχο. Συλλαμβάνει τις αλληλεπιδράσεις στις οποίες οι συμμετέχουσες υπηρεσίες δεσμεύονται για να επιτύχουν αυτόν τον στόχο και τις εξαρτήσεις μεταξύ αυτών των αλληλεπιδράσεων, που περιλαμβάνουν: αιτιώδεις ή έλεγχου ροής εξαρτήσεις (δηλ. ότι η αλληλεπίδραση πρέπει να εμφανιστεί πριν από ένα άλλο, ή ότι μια αλληλεπίδραση προκαλεί μια άλλη), τις εξαρτήσεις αποκλεισμού (δηλαδή ότι μια δεδομένη αλληλεπίδραση αποκλείει ή

αντικαθιστά μια άλλη), τις εξαρτήσεις ροής πληροφοριών, το συσχετισμό αλληλεπίδρασης, τους χρονικούς περιορισμούς, τις συναλλαγές εξαρτήσεων, κ.λπ.

Μια choreography δεν περιγράφει οποιαδήποτε εσωτερική δράση μιας συμμετέχουσας υπηρεσίας που δεν οδηγεί άμεσα σε μια εξωτερικά ορατή επίδραση, όπως ένας εσωτερικός υπολογισμός ή ένας μετασχηματισμός στοιχείων. Μια choreography περιγράφει τις αλληλεπιδράσεις από μια σφαιρική έννοια προοπτικής. Αυτό σημαίνει ότι όλες οι συμμετέχουσες υπηρεσίες αντιμετωπίζονται εξίσου. Με άλλα λόγια, μια choreography καλύπτει όλες τις αλληλεπιδράσεις μεταξύ των συμμετεχουσών υπηρεσιών που είναι σχετικές όσον αφορά το στόχο της.

Μια choreography, ενός καλά κατανοητού σεναρίου αλληλεπίδρασης υπηρεσιών, παρουσιάζεται υπό μορφή UML δραστηριότητας στο διάγραμμα 2 του σχήματος 1.3. Τρεις υπηρεσίες εμπλέκονται σε αυτήν: μια που αντιπροσωπεύει έναν «πελάτη», μια άλλη έναν «προμηθευτή» και μια τρίτη την «αποθήκη». Οι στοιχειώδεις ενέργειες στο διάγραμμα αντιπροσωπεύουν τις επιχειρησιακές δραστηριότητες που οδηγούν στα μηνύματα που στέλνονται ή λαμβάνονται. Για παράδειγμα η δράση «order goods» που συνεπάγεται από τα αποτελέσματα πελατών στέλνεται σε ένα μήνυμα στον προμηθευτή (αυτό περιγράφεται ως κειμενική σημείωση κάτω από το όνομα της). Φυσικά, κάθε μήνυμα που στέλνει τη δράση έχει μια αντίστοιχη δράση παραλαβής μηνυμάτων αλλά για να αποφευχθεί η καταστροφή του διαγράμματος, μόνο η αποστολή ή η δράση παραλαβών (όχι και οι δύο) παρουσιάζεται για κάθε ανταλλαγή μηνυμάτων. Για παράδειγμα, η δράση «στέλνει το RFQ στον προμηθευτή» στη δραστηριότητα «απόσπασμα αιτήματος» υπονοεί ότι υπάρχει η αντίστοιχη δράση «λαμβάνει το RFQ από τον πελάτη» από την πλευρά του προμηθευτή, αλλά αυτή η τελευταία δράση δεν παρουσιάζεται στο διάγραμμα.

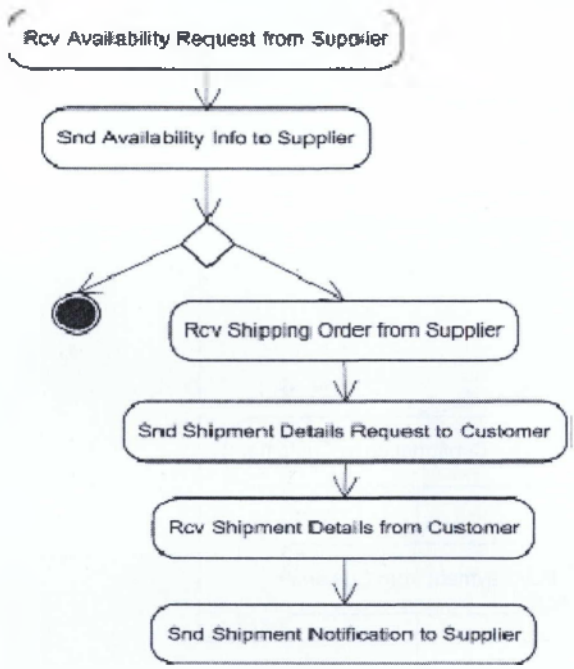
Σημειώστε ότι το σχήμα 1.3 δεν περιλαμβάνει τις δραστηριότητες και τις εναλλακτικές πορείες που απαιτούνται για να εξεταστούν τα λάθη και οι εξαιρέσεις που κάποια θα μπορούσαν ρεαλιστικά να αναμένονται στο εν λόγω σενάριο. Η συμπερίληψη αυτών των πληροφοριών θα πρόσθετε αρκετά στην πολυπλοκότητα του προτύπου.



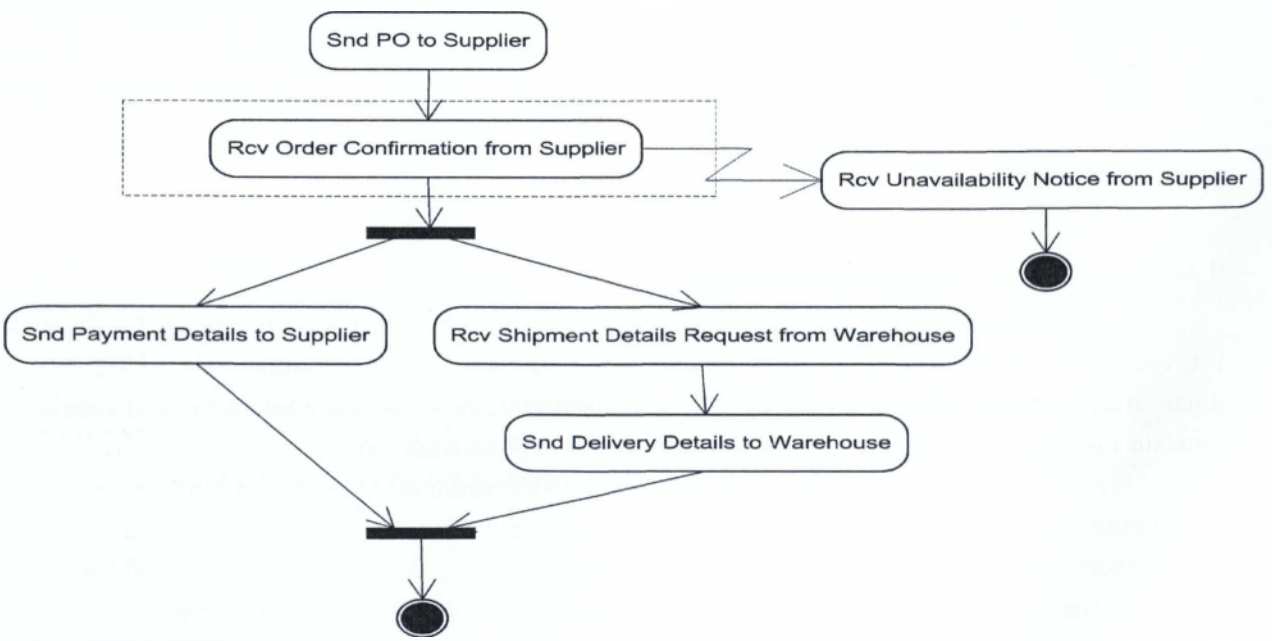
Σχήμα 1.3

1.8.2 Orchestration

Ένα πρότυπο orchestration περιγράφει τις ενέργειες επικοινωνίας και τις εσωτερικές ενέργειες στις οποίες μια υπηρεσία συμμετέχει. Οι εσωτερικές ενέργειες περιλαμβάνουν τους μετασχηματισμούς και τις κλήσεις στοιχείων στις εσωτερικές ενότητες λογισμικού. Ένα orchestration μπορεί επίσης να περιέχει τις ενέργειες ή τις εξαρτήσεις επικοινωνίας μεταξύ των ενεργειών επικοινωνίας που δεν εμφανίζονται σε οποιεσδήποτε από τα behavioral interfaces των υπηρεσιών. Αυτό συμβαίνει επειδή τα behavioral interfaces μπορούν να τεθούν στην διάθεση των εξωτερικών συμβαλλόμενων μερών και έτσι θα πρέπει μόνο να παρουσιάσουν πληροφορίες που να είναι ορατές στα υπόλοιπα μέρη. Τα Orchestrations λέγονται επίσης «εκτελέσιμες διαδικασίες» δεδομένου ότι προορίζονται να εκτελεσθούν από μια μηχανή orchestration.

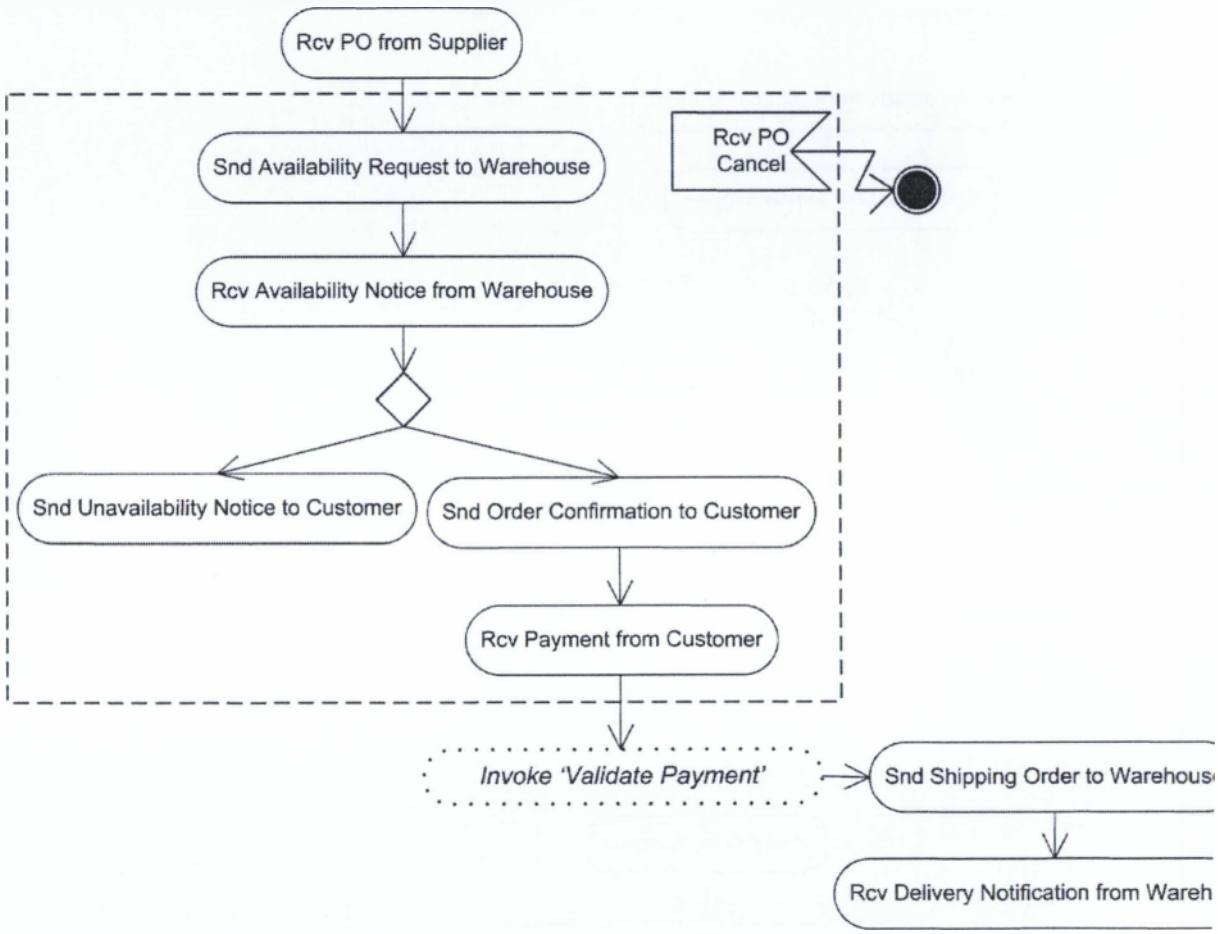


Σχήμα 1.4 behavioral interface της αποθήκης.



Σχήμα 1. 5 behavioral interface του πελάτη

Το σχήμα 1.6 παρουσιάζει ένα orchestration μιας υπηρεσίας προμηθευτών. Αυτό το orchestration περιλαμβάνει μια εσωτερική δράση για την επικύρωση της πληρωμής, που παρουσιάζεται στις διαστιγμένες γραμμές στο διάγραμμα. Αυτό μπορεί να αντιστοιχεί για παράδειγμα σε μια αλληλεπίδραση με μια υπηρεσία. Αυτό δεν εκτίθεται στον εξωτερικό κόσμο. Άλλες εσωτερικές ενέργειες μπορούν να περιληφθούν στο orchestration. Το orchestration του σχήματος 1.6 επίσης υποστηρίζει τη δυνατότητα ενός αιτήματος ακύρωσης διαταγής που παραλαμβάνεται από τον πελάτη οποτεδήποτε πριν από την πληρωμή, που οδηγεί στη λήξη της διαδικασίας.



Σχήμα 1. 6 Η orchestration υπηρεσία του πελάτη.

1.9 Ασφάλεια υπηρεσιών Ιστού

Η WS-ασφάλεια περιγράφει τις προσθήκες στο μήνυμα SOAP για να παρέχει την ποιότητα της προστασίας μέσω της ακεραιότητας, της εμπιστευτικότητας , και της ενιαίας επικύρωσης μηνυμάτων. Αυτοί οι μηχανισμοί μπορούν να χρησιμοποιηθούν για να προσαρμόσουν μια ευρεία ποικιλία των προτύπων ασφάλειας και των τεχνολογιών κρυπτογράφησης.

Η προδιαγραφή ασφάλειας υπηρεσιών Ιστού (WS-ασφάλεια) παρέχει ένα σύνολο μηχανισμών για να βοηθήσει τους υπεύθυνους για την ανάπτυξη των υπηρεσιών Ιστού να δημιουργούν ασφαλή μηνύματα SOAP. Συγκεκριμένα, η WS-ασφάλεια περιγράφει προσθήκες στο υπάρχον μήνυμα SOAP για να παρέχει ποιότητα προστασίας μέσω της εφαρμογής της ακεραιότητας , της εμπιστευτικότητας , και της ενιαίας επικύρωσης μηνυμάτων στα μηνύματα

SOAP. Αυτοί οι βασικοί μηχανισμοί μπορούν να συνδυαστούν με διάφορους τρόπους για να δημιουργηθεί μια ευρεία ποικιλία προτύπων ασφάλειας χρησιμοποιώντας ποικίλες κρυπτογραφικές τεχνολογίες.

Η WS-ασφάλεια παρέχει επίσης έναν γενικής χρήσης μηχανισμό για τα σημεία ασφάλειας με μηνύματα. Εντούτοις, κανένας συγκεκριμένος τύπος σημείου ασφάλειας δεν απαιτείται από τη WS-ασφάλεια. Έχει ως σκοπό να είναι επεκτάσιμο (π.χ. υποστήριξη πολλαπλών μορφών token ασφαλείας) για να προσαρμόσει σε ποικίλους μηχανισμούς επικύρωσης και έγκρισης. Για παράδειγμα, ένας πελάτης πρέπει να χορηγήσει την απόδειξη της ταυτότητας και μιας υπογεγραμμένης αξίωσης ότι έχει μια ιδιαίτερη επιχειρησιακή πιστοποίηση. Μια υπηρεσία Ιστού, που λαμβάνει ένα τέτοιο μήνυμα θα μπορούσε έπειτα να καθορίσει τι είδους εμπιστοσύνη τοποθετούν στην αξίωση.

Επιπλέον, η WS-ασφάλεια περιγράφει πώς να κωδικοποιήσει τα δυαδικά tokens ασφάλειας και να τα συνδέσει με τα μηνύματα SOAP. Συγκεκριμένα, οι προδιαγραφές του σχεδιαγράμματος της WS-ασφάλειας περιγράφουν πώς να κωδικοποιήσουν τα tokens ονόματος χρήστη, τα tokens X.509, τα tokens SAML, τα tokens REL και τα tokens Kerberos καθώς επίσης και πώς να περιλάβουν τα αδιαφανή κρυπτογραφημένα κλειδιά ως δείγμα των διαφορετικών δυαδικών συμβολικών τύπων. Με την WS-ασφάλεια, η περιοχή αυτών των μηχανισμών μπορεί να επεκταθεί από τις πληροφορίες επικύρωσης μεταφοράς στα αιτήματα των υπηρεσιών Ιστού. Η WS-ασφάλεια περιλαμβάνει επίσης επεκτάσιμους μηχανισμούς που μπορούν να χρησιμοποιηθούν για να περιγράψουν περαιτέρω τα πιστοποιητικά που συμπεριλαμβάνονται σε ένα μήνυμα. Η WS-ασφάλεια είναι μια δομική μονάδα που μπορεί να χρησιμοποιηθεί από κοινού με άλλα πρωτόκολλα υπηρεσιών Ιστού για να εξετάσει μια ευρεία ποικιλία απαιτήσεων της ασφάλειας μιας εφαρμογής.

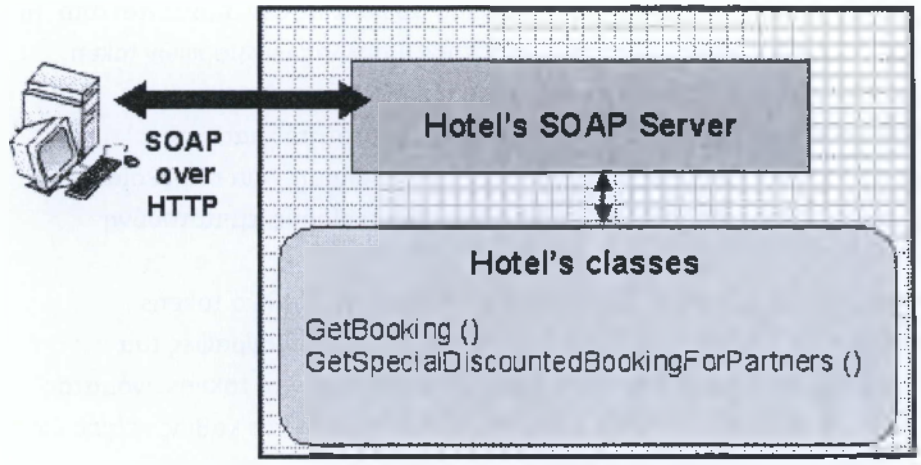
Η ακεραιότητα μηνυμάτων παρέχεται με τα XML σημεία υπογραφών και ασφάλειας για να εξασφαλίσει ότι τα μηνύματα έχουν προέλθει από τον κατάλληλο αποστολέα και δεν τροποποιήθηκαν κατά τη μεταφορά. Ομοίως, η XML κρυπτογράφηση, η εμπιστευτικότητα των μηνυμάτων και tokens ασφάλειας διατηρούν τα τμήματα ενός μηνύματος SOAP εμπιστευτικά.

Με τη χρήση του επεκτάσιμου προτύπου SOAP, οι βασισμένες σε SOAP προδιαγραφές έχουν ως σκοπό να συνδυαστούν η μια με την άλλη για να παρέχουν ένα πλούσιο περιβάλλον μηνύματος. Από μόνη της, η WS-ασφάλεια δεν εξασφαλίζει ασφάλεια ούτε παρέχει μια πλήρης λύση. Η WS-ασφάλεια είναι μια δομική μονάδα που χρησιμοποιείται από κοινού με άλλες υπηρεσίες Ιστού, εφαρμογές και πρωτόκολλα για να προσαρμόσει μια ευρεία ποικιλία των προτύπων ασφάλειας και των τεχνολογιών κρυπτογράφησης. Η εφαρμογή της WS-ασφάλειας δεν σημαίνει ότι μια υπηρεσία δεν μπορεί να δεχτεί επίθεση ή ότι η ασφάλεια δεν μπορεί να συμβιβαστεί.

1.9.1 Οι απαιτήσεις ασφαλείας για τις βασισμένες στο SOAP υπηρεσίες ιστού

Ας πάρουμε ένα παράδειγμα με βάση τον τουρισμό. Κοιτώντας το σχήμα 1.7 όπου η υπηρεσία Ιστού ξενοδοχείων εκθέτει δύο μεθόδους: GetBooking και GetSpecialDiscountedBookingForPartners. Υποθέστε ότι η μέθοδος GetBooking προορίζεται για δημόσια χρήση: καθένας μπορεί να έχει πρόσβαση σε αυτήν την μέθοδο για να πάρει μια κράτηση δωματίων. Από την άλλη, η μέθοδος GetSpecialDiscountedBookingForPartners προορίζεται να είναι αυστηρά ιδιωτική μεταξύ των συνεργατικών επιχειρήσεων. Μόνο ένας συνεργάτης παραγωγός

ταξιδιών, με τον οποίο το ξενοδοχείο έχει μια καθιερωμένη επιχειρησιακή σχέση, πρέπει να είναι σε θέση να έχει πρόσβαση στα πρόσθετα απορριμμένα ποσοστά κρατήσεων.



Σχήμα 1. 7 Η υπηρεσία Ιστού ενός ξενοδοχείου που εκθέτει δύο μεθόδους

Είναι δυνατό να επιτευχθεί αυτός ο τύπος ασφάλειας με τις network-level firewalls; Ένας κεντρικός υπολογιστής SOAP φυλάσσει μόνο πληροφορίες σχετικές με τις υπηρεσίες Ιστού που φιλοξενεί (ονόματα των υπηρεσιών, ονόματα των μεθόδων σε κάθε υπηρεσία, πού βρίσκονται οι πραγματικές κλάσεις που υλοποιούν τις υπηρεσίες Ιστού, και ούτω καθεξής) και έχει την ικανότητα της επεξεργασίας των εισερχόμενων αιτημάτων SOAP. Παρόλα αυτά, ο ίδιος ο κεντρικός υπολογιστής SOAP δεν έχει την ικανότητα να ελέγξει εάν το εισερχόμενο αίτημα SOAP προέρχεται από έναν ανώνυμο πελάτη ή έναν γνωστό συνέταιρο. Το SOAP δεν μπορεί να διακρίνει μεταξύ των ευαίσθητων και μη ευαίσθητων υπηρεσιών Ιστού και δεν μπορεί να εκτελέσει την επικύρωση, την έγκριση, και το έλεγχο προσπέλασης χρηστών.

Είναι σαφές ότι ένας απομακρυσμένος πελάτης που έχει πρόσβαση σε έναν κεντρικό υπολογιστή SOAP απολαμβάνει την ευκαιρία να κάνει χρήση οποιασδήποτε μεθόδου οποιωνδήποτε υπηρεσιών που φιλοξενούνται στον κεντρικό υπολογιστή SOAP. Μερικοί αναγνώστες μπορούν σωστά να έχουν καταλήξει στο συμπέρασμα ότι δεν είναι ασφαλές να φιλοξενούμε υπηρεσίες Ιστού διαφορετικών επιπέδων ευαισθησίας στον ίδιο κεντρικό υπολογιστή SOAP.

Ακόμα κι αν επεκταθεί ένα δίκτυο-ισόπεδο firewall που προστατεύει από τους εισβολείς, δεν θα είναι σε θέση να διακρίνει τη διαφορά μεταξύ ενός πελάτη και ενός συνεργάτη μόλις φθάσει στον κεντρικό υπολογιστή SOAP. Είναι δυνατό ένας εισβολέας να επικυρώνεται όπως ένας ανώνυμος πελάτης, φθάνει στον κεντρικό υπολογιστή SOAP, και επικαλείται τις ευαίσθητες υπηρεσίες Ιστού που είναι προορισμένες για τους συνεργάτες. Κατά συνέπεια, ένας κεντρικός υπολογιστής SOAP μοιάζει με μια τρύπα στο δίκτυό.

Υπάρχουν δύο λύσεις σε αυτό το πρόβλημα:

1. Να χρησιμοποιηθεί ένας διαφορετικός κεντρικός υπολογιστής SOAP για κάθε επίπεδο ευαισθησίας, έτσι ώστε οι διαφορετικές πολιτικές επικύρωσης να μπορούν να επιβληθούν σε κάθε επίπεδο ευαισθησίας. Αυτή η λύση μπορεί να φανεί κατάλληλη για τις υπηρεσίες Ιστού σήμερα. Εντούτοις το πραγματικό πλεονέκτημα των υπηρεσιών Ιστού βρίσκεται στις επόμενες γενεές όπου οι υπηρεσίες Ιστού θα κληθούν όχι μόνο από τους browser-βοηθημένους ανθρώπινους πελάτες, αλλά θα καλούν η μια την άλλη με μορφή chained or transactional διαδικασιών. Τέτοια σύνθετη υποδομή υπηρεσιών Ιστού είναι πολύ δύσκολο και ακριβό για να χτιστεί με την ιδέα της κατοχής ενός χωριστού κεντρικού υπολογιστή SOAP για κάθε πολιτική έγκρισης. Επιπλέον, αυτή η ιδέα επιτρέπει μετά βίας τις επαναχρησιμοποιήσιμες ή off-the-shelf λύσεις ασφάλειας.
2. Η δεύτερη επιλογή είναι να γίνει το firewall ενήμερο στην XML και το SOAP. Το firewall θα είναι σε θέση να επιθεωρήσει τα μηνύματα SOAP, που θα προσπαθούν να ταιριάξουν με τους ρόλους χρηστών από τους καταλόγους πρόσβασης, πολιτικά επίπεδα, και ούτω καθεξής. Αυτή η λύση είναι μια καλύτερη προσέγγιση. Επιτρέπει επίσης τα XML-βασισμένα τυποποιημένα πρωτόκολλα ασφάλειας, τα οποία μπορούν να υιοθετηθούν από τους vendors ασφάλειας για να εξασφαλίσουν διαλειτουργικότητα.

   Οι χρήστες υπηρεσιών Ιστού μπορούν να προσθέσουν τις πληροφορίες ασφάλειας (υπογραφή, token ασφάλειας, ονόματα αλγορίθμου) μέσα στα μηνύματα SOAP, σύμφωνα με τα XML-βασισμένα πρωτόκολλα ασφάλειας. Το ενήμερο firewall σε XML και SOAP θα ελέγξει το μήνυμα προτού να φθάσει στον κεντρικό υπολογιστή SOAP, έτσι ώστε να είναι σε θέση να ανιχνεύσει και να σταματήσει τους εισβολείς προτού να φθάσουν στην υπηρεσία.

   Με βάση τη δεύτερη προσέγγιση που περιγράφεται παραπάνω, οι W3C και OASIS αναπτύσσουν διάφορα βασισμένα σε XML πρωτόκολλα ασφάλειας. Αυτά τα πρωτόκολλα θα καθορίσουν τις διάφορες ιδιότητες ασφαλείας ενός ενήμερου firewall σε XML και SOAP.

1.9.2 Ασφάλεια βασισμένη στην XML για τις υπηρεσίες ιστού.

Η XML προδιαγραφή υπογραφών είναι μια προσπάθεια από κοινού του W3C και του IETF. Στοχεύει να παρέχει τα χαρακτηριστικά γνωρίσματα ακεραιότητας και επικύρωσης στοιχείων (μήνυμα και επικύρωση υπογραφόντων), συμπεριλαμβανόμενα στο εσωτερικό σχήμα XML.

Η προδιαγραφή κρυπτογράφησης του W3C για την XML αντιμετωπίζει το ζήτημα της εμπιστευτικότητας των στοιχείων χρησιμοποιώντας τις τεχνικές κρυπτογράφησης. Το κρυπτογραφημένο στοιχείο εμπεριέχεται μέσα στις ετικέτες XML που καθορίζονται από την προδιαγραφή κρυπτογράφησης XML.

Η WS-ασφάλεια από την OASIS καθορίζει το μηχανισμό για την ακεραιότητα, την εμπιστευτικότητα, και τα ενιαία χαρακτηριστικά γνωρίσματα επικύρωσης μηνυμάτων μέσα σε ένα μήνυμα SOAP. Η WS-ασφάλεια χρησιμοποιεί τις προδιαγραφές υπογραφών XML και κρυπτογράφησης XML και καθορίζει πώς να συμπεριλάβει τις ψηφιακές υπογραφές, τις αφομοιώσεις μηνυμάτων, και τα κρυπτογραφημένα στοιχεία σε ένα μήνυμα SOAP.

Η Security Assertion Markup Language (SAML) από την OASIS παρέχει μέσα για τις εφαρμογές συνεργατών για να μοιραστούν την επικύρωση χρηστών και τις πληροφορίες έγκρισης. Αυτό είναι ουσιαστικά το single sign-on (SSO) χαρακτηριστικό γνώρισμα που προσφέρεται από

όλους τους σημαντικούς vendors στα προϊόντα ηλεκτρονικού εμπορίου τους. Ελλείψει οποιουδήποτε τυποποιημένου πρωτοκόλλου σχετικά με τη διανομή των πληροφοριών επικύρωσης, οι προμηθευτές χρησιμοποιούν κανονικά τα cookies στην επικοινωνία HTTP για να εφαρμόσουν SSO. Με την εμφάνιση της SAML, αυτό το ίδιο στοιχείο μπορεί να συμπεριληφθεί μέσα σε XML με έναν τυποποιημένο τρόπο, έτσι τα cookies δεν απαιτούνται και το διαλειτουργικό SSO μπορεί να επιτευχθεί.

Η EXtensible Access Control Markup Language (XACML) που παρουσιάζεται από την OASIS αφήνει να εκφραστούν πολιτικές έγκρισης και πρόσβασης σε XML. Η XACML καθορίζει ένα λεξιλόγιο για να διευκρινίσει τα θέματα, τα δικαιώματα, τα αντικείμενα, και τους όρους - τα ουσιαστικά κομμάτια όλων των πολιτικών έγκρισης στις σημερινές εφαρμογές ηλεκτρονικού εμπορίου.

Κεφάλαιο 2: Τα συστατικά των υπηρεσιών Ιστού

2.1 Εισαγωγή

Στο προηγούμενο κεφάλαιο μελετήθηκαν γενικότερα οι υπηρεσίες ιστού. Σε αυτό θα γίνει μια παρουσίαση των γνωστότερων συστατικών των υπηρεσιών ιστού. Αυτό συμβαίνει διότι οι επικοινωνίες μεταξύ των υπηρεσιών ιστού είναι βασισμένες στο μήνυμα. Τα μηνύματα είναι σχηματοποιημένα με XML και ανταλλάσσονται μεταξύ αυτών των συστατικών. Χαρακτηριστικά η παράδοση χρησιμοποιεί το HTTP αλλά και άλλα πρωτόκολλα μπορούν να το αντικαταστήσουν. Το SOAP επιτρέπει την απομακρυσμένη κλήση μεθόδων και το UDDI περιλαμβάνει τις διαδικασίες για ενημέρωση και αναζήτηση των service registries. Το WSDLείναι μία γλώσσα για να περιγράφει τις διεπαφές των client/server. Η επικοινωνία μεταξύ τους είναι μία εφαρμογή που βασίζεται στο SOAP.

Σ αυτό το κεφάλαιο λοιπόν θα επεξηγήσουμε τα πρωτόκολλα, τα σχήματα και τα αρχιτεκτονικά ύφη που αντικαθιστούν το HTTP η συνυπάρχουν μαζί του κατά τη διαδικασία ανταλλαγής μηνυμάτων ή είναι υπεύθυνα για τη μορφή τους. Ξεκινώντας με την XML θα παρατεθούν οι κανόνες της, καθώς και παραδείγματα χρήσης της. Στη συνέχεια θα αναλυθεί το πρωτόκολλο SOAP η μορφη που δίνει στα μηνύματα (SOAP envelope) ,οι μηχανισμοί του όπως αυτός των χειρισμών λαθών (SOAP faults) η λειτουργία των request/response και άλλα. Επίσης θα συζητηθεί το WSDL περιγράφοντας τα στοιχεία που το απαρτίζουν. Ακόμα θα γίνει σύγκριση του πρωτοκόλλου SOAP με το αρχιτεκτονικό ύφος REST. Θα αναφερθούμε στο πως προκύπτει το συγκεκριμένο ύφος παίρνοντας ως αφετηρία το αρχιτεκτονικό ύφος NULL και θα αναλύσουμε τα χαρακτηριστικά των RESTfull services. Επιπλέον θα γίνει αναφορά στο UDDI και στο WADL. Τέλος για όλα τα παραπάνω θα παρατεθούν αναλυτικοί πίνακες, σχήματα και εικόνες.

2.2 XML

Η XML (*Extensible Markup Language*) είναι μία γλώσσα σήμανσης που περιέχει ένα σύνολο κανόνων για την ηλεκτρονική κωδικοποίηση κειμένων. Ορίζεται, κυρίως, στην προδιαγραφή XML 1.0 που δημιούργησε ο διεθνής οργανισμός προτύπων W3C (World Wide Web Consortium), αλλά και σε διάφορες άλλες σχετικές προδιαγραφές ανοιχτών προτύπων.

Η XML σχεδιάστηκε δίνοντας έμφαση στην απλότητα, τη γενικότητα και τη χρησιμότητα στο Διαδίκτυο. Είναι μία μορφοποίηση δεδομένων κειμένου, με ισχυρή υποστήριξη Unicode για όλες τις γλώσσες του κόσμου. Αν και η σχεδίαση της XML εστιάζετα στα κείμενα, χρησιμοποιείται ευρέως και για την αναπαράσταση αυθαίρετων δομών δεδομένων, που προκύπτουν στις υπηρεσίες ιστού.

Υπάρχει μία ποικιλία διεπαφών προγραμματιστικών εφαρμογών, που μπορούν να χρησιμοποιούν οι προγραμματιστές, για να προσπελαύνουν δεδομένα XML, αλλά και διάφορα συστήματα σχημάτων XML, τα οποία είναι σχεδιασμένα για να βοηθούν στον ορισμό γλωσσών, που προκύπτουν από την XML. Έως το 2009, έχουν αναπτυχθεί εκατοντάδες γλώσσες που βασίζονται στην XML, συμπεριλαμβανομένων του RSS, του SOAP και της XHTML.

Ερώτημα όμως είναι γιατί πρέπει να επιλέξουμε τη χρήση της XML από μεταγενέστερες γλώσσες που βασίζονται σε αυτή. Η XML παρουσιάζει χαρακτηριστικά που την κάνουν καταλληλότερη σύμφωνα με τη φιλοσοφία των υπηρεσιών ιστού.

Αρχικά η XML είναι καθαρό κείμενο χωρίς τη δυαδική πληροφορία που έχει ως αποτέλεσμα την πιο εύκολη επεξεργασία. Ο ίδιος ο υπολογιστής χρησιμοποιεί XML μορφοποιήσεις οι οποίες είναι επεξεργάσιμες αναπαραστάσεις για πόρους που επιτρέπουν την εφαρμογή νέων εργαλείων σε παλιά δεδομένα. Ακόμα με την XML απλουστεύεται η διασύνδεση με απομακρυσμένα συστήματα και επιπρόσθετα οι διαδικτυακές εφαρμογές διαβάζουν XML και μέσω αυτής μπορούν να μοιράζονται δεδομένα. Οποιαδήποτε εφαρμογή μπορεί να επικοινωνήσει με μια άλλη χρησιμοποιώντας XML ανεξάρτητα από την πλατφόρμα και αυτό αποτελεί πλεονέκτημα σε αντίθεση με τη δυαδική πληροφορία η οποία παρουσιάζει εξαρτήσεις. Τέλος πρέπει να αναφερθεί ότι η XML είναι μια μέθοδος για να τοποθετούμε δομημένη πληροφορία σε ένα αρχείο κειμένου.

Ιδιαίτερα δημοφιλής περιγραφική γλώσσα είναι η XHTML. Όμως σε σύγκριση με την XML δεν μπορεί να προσφέρει τις ίδιες δυνατότητες στις υπηρεσίες ιστού. Αυτό συμβαίνει γιατί η HTML χρησιμοποιείται κυρίως για την κατασκευή ιστοσελίδων. Οι ιστοσελίδες είναι σχεδιασμένες για να είναι κατανοητές από ανθρώπους που ενδιαφέρονται για τη διάταξη και το styling και όχι για καθαρή πληροφορία όπως στις υπηρεσίες ιστού.

Κάθε URI πρέπει να έχει μία human-readable(κατανοητό από τον άνθρωπο) και μία machine-processable (επεξεργάσιμη από τη μηχανή) αναπαράσταση.

- Οι πελάτες των υπηρεσιών ιστού ζητούν την machine-processable
- Οι browsers ζητούν την human-readable.

Παρόλα αυτά η πληροφορία σε μερικές σελίδες είναι υπερβολικά πολύπλοκη για να γίνει κατανοητή από τις μηχανές.

2.2.1 Κανόνες XML:

Για να γίνει η πληροφορία κατανοητή από τις μηχανές θα πρέπει τα δεδομένα να υπόκεινται σε κάποιους κανονισμούς. Οι κανόνες τις XML που κάνουν την πληροφορία απλή για να γίνεται αντιληπτή από τις μηχανές χωρίς δυσκολία είναι οι ακόλουθοι:

1. well-formed και τι τα valid έγγραφα

   Βασικά, υπάρχουν δυο τύποι XML εγγράφων : τα well-formed και τα valid.Ένα well-formed XML έγγραφο ακολουθεί τους γενικούς κανόνες σύνταξης της XML, οι οποίοι είναι πιο αυστηροί από αυτούς της HTML και της SGML. Οι χαρακτήρες δεδομένων της XML δεν μένουν ποτέ δίχως ένα markup τέλους οποιουδήποτε είδους , είτε end-tag όπως το ζεύγος <MYTAG></MYTAG>, είτε ένα empty element tag με το σύμβολο της καθέτου πριν το σύμβολο >, όπως <MYTAG/>. Το markup της XML ξεκινάει πάντοτε με το σύμβολο < ή με το σύμβολο &. Οι τύποι των στοιχείων και τα ονόματα των εισαγωγικών είναι case sensitive. Τα χαρακτηριστικά απαιτούν εισαγωγικά κ.α.

   Τα valid XML έγγραφα ακολουθούν ένα συγκεκριμένο Document Type Definition(DTD) Ευθύνη των συγγραφέων και των εκδοτών είναι να επιβεβαιώνουν την εγκυρότητα των XML εγγράφων, ενώ οι ικανοί XML browsers χρειάζονται μόνον τον έλεγχο για καλή μορφοποίηση εάν θέλουν να

διαβάσουν XML έγγραφα. Έτσι κάθε XML parser ελέγχει το έγγραφο για καλή μορφοποίηση και εγκυρότητα ενώ ο browser αναζητά μονάχα την καλή μορφοποίηση.

Αν ένα data object είναι well-formed είναι ένα XML έγγραφο. Ένα well-formed XML έγγραφο μπορεί να είναι valid εάν πλήρη κάποιους περιορισμούς.

Κάθε XML έγγραφο έχει μια λογική και μια φυσική δομή. Φυσικά, το κείμενο συνθέτεται από μονάδες που καλούνται *οντότητες (entities)*.Η οντότητα μπορεί να αναφέρεται σε άλλες οντότητες για να προκαλέσει τον συνυπολογισμό τους στο έγγραφο. Το έγγραφο ξεκινάει από την *«αφετηρία» ("root")* ή από την *οντότητα του εγγράφου (document entity)*. Λογικά, το έγγραφο αποτελείται από δηλώσεις, στοιχεία, σχόλια, αναφορές σε χαρακτήρες και οδηγίες εκτέλεσης , καθένα από τα οποία φαίνονται στο έγγραφο με σαφές markup.

2. XML document(.xml):

   Είναι ουσιαστικά τα XML στιγμιότυπα του σχήματος(.xsd).

3. DTD και XML schema.

   Τα είδη των στοιχείων που επιτρέπεται να εμφανίζονται μέσα στο XML έγγραφο όπως και οι συνδυασμοί τους.

   I. DTD – Document Type Definition: Πρόκειται για την πιο παλιά γλώσσα (στην ουσία κληρονομήθηκε από την SGML). Χρησιμοποιεί μια σχετικά πυκνή (και μερικές φορές δυσνόητη στον άνθρωπο) μορφή γραφής για να περιγράψει τις οντότητες και τις επιτρεπτές ετικέτες (tags).

   II. XML schema:

   Το σχήμα XML, που δημοσιεύεται ως W3C σύσταση τον Μάιο του 2001, είναι μια από τις διάφορες γλώσσες σχημάτων XML. Ήταν η πρώτη χωριστή γλώσσα σχημάτων για XML που κατάφερε να επιτύχει σύσταση από το W3C. Λόγω της σύγχυσης μεταξύ του σχήματος XML ως συγκεκριμένη προδιαγραφή του W3C, και της χρήσης του ίδιου όρου για να περιγράψουν τις γλώσσες σχημάτων γενικά, μερικά μέρη της κοινότητας χρηστών αναφέρθηκαν σε αυτήν την γλώσσα ως WXS, ένα αρκτικόλεξο για το σχήμα XML του W3C, ενώ άλλοι αναφέρθηκαν σε αυτό ως XSD, ένα αρκτικόλεξο για το XML Schema Document ένα εγγραφο που γράφεται σε σχήμα XML , χαρακτηριστικά περιέχει το «xsd» ένα XML namespace αποθηκευμένο με τη κατάληξη .xsd . Στην έκδοση 1.1 (την περίοδο του Ιουλίου του 2011 μια σύσταση υποψηφίων), το W3C έχει επιλέξει να υιοθετήσει το XSD ως προτεινόμενο όνομα, και αυτό είναι το όνομα που χρησιμοποιείται εδώ.

   Όπως όλες τις γλώσσες σχημάτων XML, το XSD μπορεί να χρησιμοποιηθεί για να εκφράσει ένα σύνολο κανόνων στο οποίο ένα έγγραφο XML πρέπει να προσαρμοστεί προκειμένου να θεωρηθεί «έγκυρο» σύμφωνα με εκείνο το σχήμα. Εντούτοις, αντίθετα από τις περισσότερες άλλες γλώσσες σχημάτων, το XSD σχεδιάστηκε επίσης με την πρόθεση ότι ο προσδιορισμός της ισχύος ενός εγγράφου θα παρήγαγε μια συλλογή πληροφοριών που εμμένουν στους συγκεκριμένους τύπους στοιχείων. Μια τέτοια επικύρωση infoset μπορεί να είναι χρήσιμη στην ανάπτυξη του λογισμικού επεξεργασίας εγγράφων XML, αλλά η γλωσσική εξάρτηση των σχημάτων στους συγκεκριμένους τύπους στοιχείων έχει προκαλέσει κριτική.

a. Σχήματα και σχήματα εγγράφων.

Τεχνικά, ένα σχήμα είναι μια αφηρημένη συλλογή μεταδεδομένων, που αποτελούνται από ένα σύνολο τμημάτων σχημάτων: κυρίως δηλώσεις στοιχείων και ιδιοτήτων και σύνθετοι ή απλοί ορισμοί τύπων. Αυτά τα συστατικά δημιουργούνται συνήθως με την επεξεργασία μιας συλλογής εγγράφων σχημάτων, τα οποία περιέχουν την πηγαία γλώσσα και τους ορισμούς αυτών των συστατικών. Παρόλα αυτά συνήθως, ένα έγγραφο σχημάτων αναφέρεται συχνά ως σχήμα.

Τα έγγραφα σχημάτων οργανώνονται από namespace: όλα τα ονομασμένα τμήματα σχημάτων ανήκουν σε έναν συγκεκριμένο namespace, και αυτό το namespace είναι μία ιδιότητα του εγγράφου σχημάτων συνολικά. Ένα έγγραφο σχημάτων μπορεί να συμπεριλάβει άλλα έγγραφα σχημάτων στο ίδιο namespace, και μπορεί να εισαγάγει τα έγγραφα σχημάτων σε ένα διαφορετικό namespace.

Όταν ένα στιγμιότυπο ενός εγγράφου επικυρώνεται ενάντια σε ένα σχήμα (μια διαδικασία γνωστή ως αξιολόγηση), το σχήμα που χρησιμοποιείται για την επικύρωση μπορεί είτε να παρασχεθεί ως παράμετρος στη μηχανή επικύρωσης, είτε αυτό μπορεί να παραπεμφθεί άμεσα από το έγγραφο περίπτωσης χρησιμοποιώντας δύο πρόσθετες ιδιότητες, xsi: schemaLocation και xsi: noNamespaceSchemaLocation.(ο τελευταίος μηχανισμός απαιτεί από τον πελάτη που επικαλείται την επικύρωση να εμπιστευθεί το έγγραφο για να ξέρει ότι επικυρώνεται έναντι στο σωστό σχήμα. «το xsi» είναι το συμβατικό πρόθεμα για το namespace)

b. Τύποι πληροφορίας

Αντίθετα από τα DTDs, ένα σχήμα XML επιτρέπει στο περιεχόμενο ενός στοιχείου ή μιας ιδιότητας να επικυρωθεί έναντι σε έναν τύπο στοιχείων. Για παράδειγμα, μια ιδιότητα να περιοριστεί και να κρατήσει μόνο μια έγκυρη ημερομηνία ή έναν δεκαδικό αριθμό. Το XSD παρέχει ένα σύνολο 19 βασικών τύπων δεδομένων (anyURI, base64Binary, boolean, date, dateTime, decimal, double, duration, float, hexBinary, gDay, gMonth, gMonthDay, gYear, gYearMonth, NOTATION, QName, string, and time). Επιτρέπει στους νέους τύπους στοιχείων να κατασκευαστούν από τους βασικούς μέσω τριών μηχανισμών:

- περιορισμός (που μειώνει το σύνολο επιτρεπόμενων τιμών),
- λίστα(που επιτρέπει μια ακολουθία τιμών), και
- ένωση (που επιτρέπει μια επιλογή τιμών από διάφορους τύπους).

Οι είκοσι πέντε παραγόμενοι τύποι καθορίζονται μέσα στην ίδια προδιαγραφή, και οι περαιτέρω παραγόμενοι τύποι μπορούν να καθοριστούν από τους χρήστες στα σχήματά τους.

c. Post schema validation Infoset

Αφότου βασισμένη στο XML σχήμα επικύρωση, είναι δυνατόν να εκφράσει τη δομή και την περιεκτικότητα σε έγγραφο XML από την άποψη του προτύπου κατά τη διάρκεια της επικύρωσης. Το πρότυπο στοιχείων σχημάτων XML περιλαμβάνει:

- το λεξιλόγιο (ονόματα στοιχείων και ιδιοτήτων)
- το ικανοποιημένο πρότυπο (σχέσεις και δομή)
- οι τύποι στοιχείων.

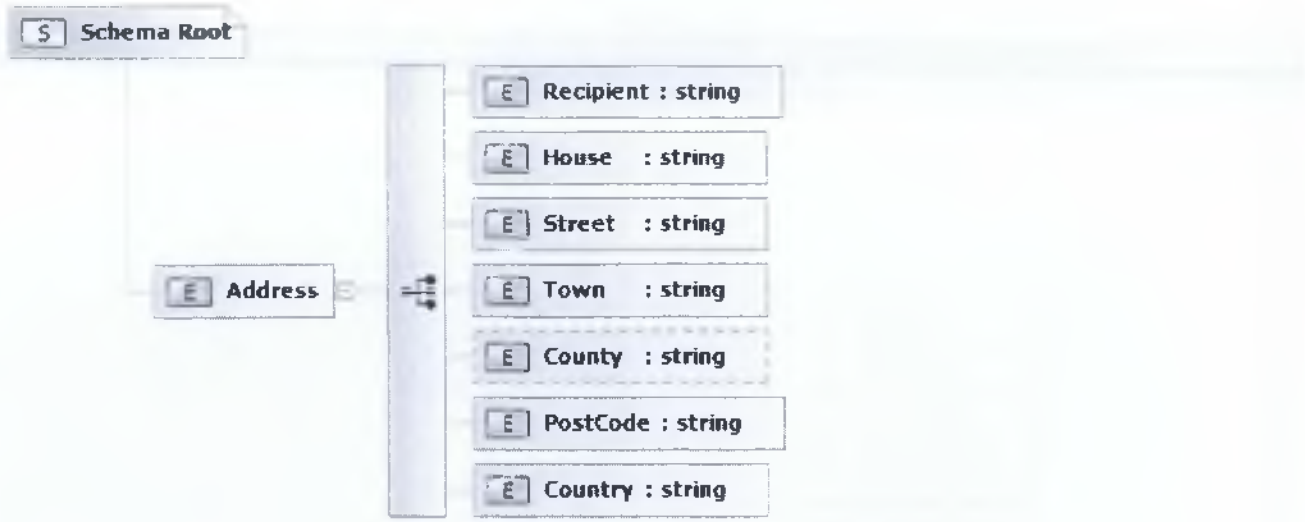
Αυτή η συλλογή πληροφοριών καλείται Post schema validation Infoset (PSVI). Το PSVI δίνει σε ένα έγκυρο έγγραφο XML «τον τύπο του» και διευκολύνει τη μεταχείριση του

εγγράφου ως αντικείμενο, χρησιμοποιώντας αντικειμενοστραφή παραδείγματα προγραμματισμού (OOP).

Αυτό είναι ένα παράδειγμα ενός μάλλον απλού εγγράφου σχημάτων για να περιγράψει μια διεύθυνση.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Address">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Recipient" type="xs:string" />
        <xs:element name="House" type="xs:string" />
        <xs:element name="Street" type="xs:string" />
        <xs:element name="Town" type="xs:string" />
        <xs:element name="County" type="xs:string" minOccurs="0" />
        <xs:element name="PostCode" type="xs:string" />
        <xs:element name="Country">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="IN" />
              <xs:enumeration value="DE" />
              <xs:enumeration value="ES" />
              <xs:enumeration value="UK" />
              <xs:enumeration value="US" />
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Διάφορα εργαλεία ανάπτυξης μπορούν να χρησιμοποιηθούν για να δημιουργήσουν μια γραφική αντιπροσώπευση ενός σχήματος. Πολλά από αυτά δημιουργούν διαγράμματα παρόμοια με αυτό που παρουσιάζεται παρακάτω:



Σχήμα 2. 1

2.3 SOAP

SOAP (Simple Object Access Protocol) είναι ένας τρόπος για ένα πρόγραμμα που τρέχει σε ένα είδος λειτουργικού συστήματος (όπως τα Windows) για να επικοινωνήσει με ένα πρόγραμμα στο ίδιο ή σε άλλο είδος λειτουργικού συστήματος (όπως Linux) με τη χρήση του Hypertext Transfer Protocol του World Wide Web (HTTP) και της Extensible Markup Language του (XML) ως μηχανισμούς για την ανταλλαγή πληροφοριών. Δεδομένου ότι τα πρωτόκολλα Ιστού είναι εγκατεστημένα και διαθέσιμα προς χρήση από όλες τις σημαντικές πλατφόρμες λειτουργικών συστημάτων, το HTTP και η XML παρέχουν μια λύση στο πρόβλημα για το πώς τα προγράμματα που τρέχουν σε πλαίσιο διαφορετικών λειτουργικών συστημάτων σε ένα δίκτυο μπορούν να επικοινωνήσουν το ένα με το άλλο. Το SOAP διευκρινίζει ακριβώς πώς να κωδικοποιήσει μια επικεφαλίδα HTTP και ένα αρχείο XML έτσι ώστε ένα πρόγραμμα σε ένα υπολογιστή να μπορεί να καλέσει ένα πρόγραμμα σε έναν άλλο υπολογιστή και να περάσει πληροφορίες. Διευκρινίζει επίσης πώς το πρόγραμμα που έχει κληθεί μπορεί να επιστρέψει μια απάντηση.

Το SOAP αναπτύχθηκε με σκοπό να καταφέρει να διαπερνά τα firewalls και τα proxy servers που παλαιότερες τεχνολογίες όπως το RPC (Remote Procedure Calls) αδυνατούσαν να το επιτύχουν. Και το RPC όπως και το SOAP χρησιμοποιούν το πρωτόκολλο HTTP(είναι ένα πρωτόκολλο επικοινωνίας εφαρμογής που χρησιμοποιείται για την αποστολή των μηνυμάτων μέσω του Διαδικτύου και είναι ανεξάρτητο από γλώσσα και πλατφόρμα) διότι είναι ο καλύτερος τρόπος επικοινωνίας αφού υποστηρίζεται από όλους τους Internet servers και browsers.

Το SOAP αντιπροσώπευε μια φορά το « Simple Object Access Protocol» αλλά αυτό το αρκτικόλεξο σταμάτησε να ισχύει με την έκδοση 1.2 των προτύπων. Στην έκδοση 1.2 έγινε μια σύσταση από το W3C στις 24 Ιουνίου 2003. Το αρκτικόλεξο είναι μερικές φορές συγκεχυμένο με τον όρο SOA, το οποίο αντιπροσωπεύει την προσανατολισμένη προς τις υπηρεσίες αρχιτεκτονική. Το πρότυπο SOAP μπορεί να είναι μέρος μιας εφαρμογής SOA, αλλά τα αρκτικόλεξα είναι ανεξάρτητα.

Το SOAP σχεδιάστηκε αρχικά από τους Dave Winer, Don Box, , Bob Atkinson, and Mohsen Al-Ghosein το 1998 σε ένα πρόγραμμα για τη Microsoft (όπου Atkinson και Al-Ghosein δούλευαν ήδη εκεί), ως πρωτόκολλο πρόσβασης αντικειμένου. Η προδιαγραφή SOAP διατηρείται αυτήν την περίοδο από την ομάδα εργασίας πρωτοκόλλου XML της World Wide Web Consortium.

Αφότου εισήχθη αρχικά το SOAP, έγινε το ελλοχεύον στρώμα ενός πιο σύνθετου συνόλου υπηρεσιών Ιστού, βασισμένου στη Web Services Description Language (WSDL) και την Universal Description Discovery and Integration(UDDI). Αυτές οι υπηρεσίες, ειδικά οι UDDI, έχουν αποδειχθεί πολύ λιγότερου ενδιαφέροντος, αλλά μια εκτίμηση τους δίνει μια πληρέστερη κατανόηση του αναμενόμενου ρόλου του SOAP έναντι στο πώς οι υπηρεσίες Ιστού έχουν εξελιχθεί πραγματικά.

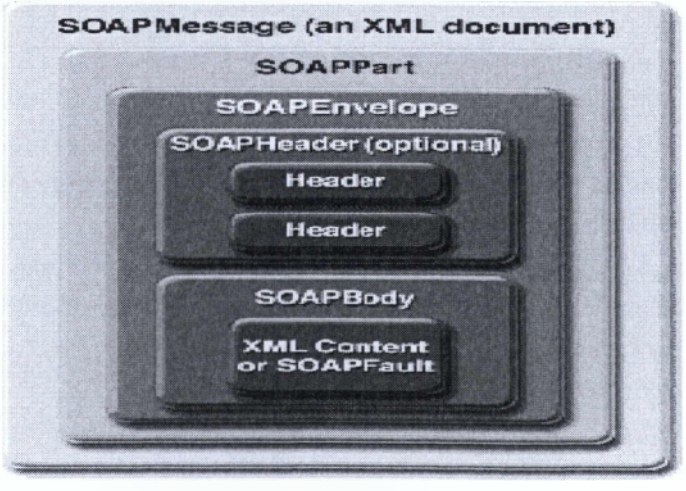
Τώρα το SOAP χρησιμοποιείται κυρίως για τους ακόλουθους λόγους:

1. Business to Business integration (Επιχείρηση στην επιχείρηση ολοκλήρωση): Το SOAP επιτρέπει στις επιχειρήσεις να αναπτύξει τις εφαρμογές τους, και να καταστήσει έπειτα εκείνες τις εφαρμογές διαθέσιμες σε άλλες επιχειρήσεις.
2. Distributed applications (Διανεμημένες εφαρμονές): προγράμματα όπως αυτά της διαχείρισης βάσεων δεδομένων θα μπορούσαν να αποθηκευτούν σε έναν κεντρικό υπολογιστή και να προσεγγίζονται και να ρυθμίζονται από τους πελάτες σε ολόκληρο το Διαδίκτυο.

2.3.1 SOAP envelope

Ένα μήνυμα SOAP είναι ένα έγγραφο XML που αποτελείται από έναν υποχρεωτικό φάκελο SOAP μια προαιρετική επιγραφή SOAP, και ένα υποχρεωτικό σώμα SOAP. Το παρόν έγγραφο XML αναφέρεται ως μήνυμα SOAP για το υπόλοιπο αυτής της προδιαγραφής. Το προσδιοριστικό namespace για τα στοιχεία και τις ιδιότητες που καθορίζονται σε αυτό το τμήμα είναι «το http://schemas.xmlsoap.org/soap/envelope/». Ένα μήνυμα SOAP περιέχει τα εξής:

- Ο φάκελος είναι το κορυφαίο στοιχείο του εγγράφου XML που αντιπροσωπεύει το μήνυμα.
- Η επικεφαλίδα είναι ένας γενικός μηχανισμός για προσθέτεις χαρακτηριστικά γνωρίσματα σε ένα μήνυμα SOAP με έναν αποκεντρωμένο τρόπο χωρίς προγενέστερη συμφωνία μεταξύ των επικοινωνούντων συμβαλλόμενων μερών. Το SOAP καθορίζει μερικές ιδιότητες που μπορούν να χρησιμοποιηθούν για να προσδιορίσουν ποιος πρέπει να εξετάσει ένα χαρακτηριστικό γνώρισμα για το εάν είναι προαιρετικό ή υποχρεωτικό
- Το σώμα είναι ένα για τις υποχρεωτικές πληροφορίες προοριζόμενες για τον τελευταίο παραλήπτη του μηνύματος. Το SOAP καθορίζει ένα στοιχείο για το σώμα, το οποίο είναι το στοιχείο ελαττωμάτων που χρησιμοποιείται για την υποβολή εκθέσεων των λαθών.



Σχήμα 2. 2 Φάκελος SOAP

Οι κανόνες γραμματικής είναι οι ακόλουθοι:

1. Envelope
   - Το όνομα των στοιχείων είναι «envelope».
   - Το στοιχείο ΠΡΕΠΕΙ να είναι παρόν σε ένα μήνυμα SOAP
   - Το στοιχείο ΕΙΝΑΙ ΠΙΘΑΝΟΝ να περιέχει namespace δηλώσεις καθώς επίσης και τις πρόσθετες ιδιότητες. Εάν το παρόν, περιέχει τέτοιες πρόσθετες ιδιότητες ΠΡΕΠΕΙ να είναι κατάλληλο σύμφωνα με το namespace. Ομοίως, το στοιχείο ΕΙΝΑΙ ΠΙΘΑΝΟ να περιέχει πρόσθετα υποστοιχεία. Εάν περιέχονται αυτά τα στοιχεία ΠΡΕΠΕΙ να είναι κατάλληλα με βάση το namespace και ΠΡΕΠΕΙ να ακολουθήσουν το στοιχείο σώματος SOAP.
2. Header
   - Το όνομα των στοιχείων είναι «Header».
   - Το στοιχείο ΜΠΟΡΕΙ να είναι παρόν σε ένα μήνυμα SOAP. Εάν περιέχεται το στοιχείο ΠΡΕΠΕΙ να είναι το πρώτο άμεσο στοιχείο παιδιών(που κληρονομεί άμεσα) ενός στοιχείου SOAP envelope.
   - Το στοιχείο ΕΙΝΑΙ ΠΙΘΑΝΟ να περιέχει ένα σύνολο καταχωρήσεων header το κάθε ένα από τα οποία είναι ένα άμεσο στοιχείο παιδιών του στοιχείου header SOAP. Όλα τα άμεσα στοιχεία που κληρονομούν από τα στοιχεία header SOAP ΠΡΕΠΕΙ να είναι κατάλληλα σύμφωνα με τα namespace.
3. Body
   - Το όνομα των στοιχείων είναι «body».
   - Το στοιχείο ΠΡΕΠΕΙ να είναι παρόν σε ένα μήνυμα SOAP και ΠΡΕΠΕΙ να είναι ένα άμεσο στοιχείο παιδί ενός στοιχείου του envelope SOAP. ΠΡΕΠΕΙ άμεσα να ακολουθήσει το στοιχείο header SOAP εάν είναι παρόν. Διαφορετικά ΠΡΕΠΕΙ να είναι το πρώτο άμεσο στοιχείο που κληρονομεί το στοιχείο του SOAP envelope.
   - Το στοιχείο ΕΙΝΑΙ ΠΙΘΑΝΟ να περιέχει ένα σύνολο καταχωρήσεων body σε κάθε ένα που είναι ένα άμεσο στοιχείο παιδιών του στοιχείου SOAP body. Τα άμεσα στοιχεία παιδιών του στοιχείου SOAP body ΜΠΟΡΟΥΝ να είναι κατάλληλα σύμφωνα με τα namespace. Το SOAP καθορίζει το στοιχείο SOAP fault, το οποίο χρησιμοποιείται για να δείξει τα μηνύματα λάθους.

2.3.2 SOAP Fault

Το στοιχείο SOAP Fault χρησιμοποιείται για να φέρει τις πληροφορίες λάθους ή και θέσης μέσα σε ένα μήνυμα SOAP .Το στοιχείο SOAP Fault ΠΡΕΠΕΙ να εμφανιστεί ως είσοδος σωμάτων και ΔΕΝ ΠΡΕΠΕΙ να εμφανιστεί περισσότερο από μία φορά μέσα σε ένα στοιχείο σώματος.
Το στοιχείο SOAP Fault καθορίζει τα ακόλουθα τέσσερα υποστοιχεία:

1. Faultcode: Το στοιχείο faultcode προορίζεται για χρήση από το λογισμικό και παρέχει έναν αλγοριθμικό μηχανισμό για το fault. Το faultcode ΠΡΕΠΕΙ να είναι παρόν σε ένα στοιχείο SOAP Fault και η αξία faultcode ΠΡΕΠΕΙ να είναι ένα κατάλληλο όνομα. Το SOAP καθορίζει ένα μικρό σύνολο κωδίκων SOAP Fault που καλύπτουν τα βασικά SOAP Fault.
2. Faultstring: Το faultstring στοιχείο προορίζεται να παρέχει μια κατανοήσιμη από τον άνθρωπο εξήγηση του ελαττώματος και δεν προορίζεται για την αλγοριθμική επεξεργασία. Το faultstring στοιχείο είναι παρόμοιο με τη «'Reason-Phrase'» που καθορίζεται από το

HTTP. ΠΡΕΠΕΙ να είναι παρόν σε ένα στοιχείο SOAP Fault και ΠΡΕΠΕΙ να παρέχει τουλάχιστον κάποιες πληροφορίες που εξηγούν τη φύση του ελαττώματος.

3. Faultactor:Το στοιχείο faultactor προορίζεται να παρέχει τις πληροφορίες για ποιος ανάγκασε το ελάττωμα για να συμβεί μέσα στην πορεία μηνυμάτων. Είναι παρόμοιο με τις ιδιότητες δραστών SOAP αλλά αντί της ένδειξης του προορισμού της εισόδου επικεφαλίδων, δείχνει την πηγή του ελαττώματος. Η αξία των ιδιοτήτων faultactor είναι ένα URI που προσδιορίζει την πηγή. Οι εφαρμογές που δεν ενεργούν ως τελευταίος προορισμός του μηνύματος SOAP ΠΡΕΠΕΙ να περιλάβουν το στοιχείο faultactor σε ένα στοιχείο SOAP Fault. Ο τελευταίος προορισμός ενός μηνύματος είναι δυνατόν να χρησιμοποιεί το στοιχείο faultactor για να δείξει ρητά ότι παρήγαγε το ελάττωμα.
4. Details: Το στοιχείο της λεπτομέρειας προορίζεται για συγκεκριμένες πληροφορίες λάθους εφαρμογής μεταφοράς, σχετικές με το στοιχείο σώματος. ΠΡΕΠΕΙ να είναι παρόν εάν το περιεχόμενο του στοιχείου σώματος μπορεί να υποβληθεί σε επεξεργασία επιτυχώς. ΔΕΝ ΠΡΕΠΕΙ να χρησιμοποιηθεί για να φέρει τις πληροφορίες για τις πληροφορίες λάθους που ανήκουν στις καταχωρήσεις επικεφαλίδων. Οι λεπτομερείς πληροφορίες λάθους που ανήκουν στις καταχωρήσεις επικεφαλίδων ΠΡΕΠΕΙ να φερθούν μέσα στις καταχωρήσεις επιγραφών.

   Η απουσία του στοιχείου λεπτομέρειας στο στοιχείο ελαττωμάτων δείχνει ότι το ελάττωμα δεν συσχετίζεται με την επεξεργασία του στοιχείου σώματος. Αυτό μπορεί να χρησιμοποιηθεί για να διακρίνει εάν το στοιχείο σώματος υποβλήθηκε σε επεξεργασία ή όχι σε περίπτωση κατάστασης ελαττωμάτων.

   Όλα τα άμεσα στοιχεία παιδιών του στοιχείου λεπτομέρειας καλούνται καταχωρήσεις λεπτομέρειας και κάθε είσοδος λεπτομέρειας κωδικοποιείται ως ανεξάρτητο στοιχείο μέσα στο στοιχείο λεπτομέρειας

   Οι κανόνες κωδικοποίησης για τις καταχωρήσεις λεπτομέρειας είναι οι ακόλουθοι:

   Πρώτων μια είσοδος λεπτομέρειας προσδιορίζεται πλήρως - κατάλληλο όνομα στοιχείων, το οποίο αποτελείται από το namespace URI και το τοπικό όνομα. Τα άμεσα στοιχεία παιδιών του στοιχείου λεπτομέρειας ΜΠΟΡΟΥΝ να είναι κατάλληλα σύμφωνα με τα namespace.Και δεύτερον οι ιδιότητες SOAP encodingStyle ΜΠΟΡΟΥΝ να χρησιμοποιηθούν για να δείξουν το ύφος κωδικοποίησης που χρησιμοποιείται για τις καταχωρήσεις λεπτομέρειας.

   Άλλα υποστοιχεία ελαττωμάτων ΜΠΟΡΟΥΝ να είναι παρόντα, υπό τον όρο ότι είναι κατάλληλα σύμφωνα με τα namespace.

2.3.3 SOAP request

Το ακόλουθο δείγμα παρουσιάζει χαρακτηριστικό αίτημα SOAP που στέλνεται σε έναν κεντρικό υπολογιστή που τρέχει Microsoft® SQLXML.

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <GetCustomerInfo xmlns="http://SERVER/VDir/VName">
      <CustomerID>ALFKI</CustomerID>
      <OutputParam />
    </GetCustomerInfo>
  </soap:Body>
</soap:Envelope>
```

Το αίτημα SOAP συμπεριλαμβάνεται στο στοιχείο <Body>. Σε αυτό το αίτημα, ο πελάτης ζητά μια λειτουργία GetCustomerInfo, η οποία είναι μια αποθηκευμένη διαδικασία που περιγράφεται στην Initial Setup for the SOAP Sample Applications. Η παράμετρος λειτουργίας <CustomerID> (με την αξία «ALFKI») και η παράμετρος λειτουργίας <OutputParam> συμπεριλαμβάνονται ως στοιχεία παιδιών του στοιχείου <GetCustomerInfo>.

Οι παράμετροι εισαγωγής αντιμετωπίζονται ως εξής:

- Εάν μια λειτουργία SOAP απαιτεί μια παράμετρο εισαγωγής και αυτή η παράμετρος δεν συμπεριλαμβάνεται στο αίτημα SOAP, καμία τιμή δεν περνά στην διαδικασία (ή το template) που καλείται. Η default δράση που καθορίζεται στην αποθηκευμένη διαδικασία (ή το template) πραγματοποιείται.
- Εάν μια λειτουργία SOAP απαιτεί μια παράμετρο εισαγωγής και αυτή η παράμετρος συμπεριλαμβάνονται στο αίτημα αλλά καμία τιμή δεν ορίζεται σε αυτη την παράμετρο, περνά στην αποθηκευμένη διαδικασία (ή το template) με μια κενή σειρά ως τιμή (όχι μια ΜΗΔΕΝΙΚΗ τιμή).
- Μια παράμετρος λειτουργίας μπορεί να τεθεί ως στόχος ΝΑ ΑΧΡΗΣΤΕΥΣΕΙ την παροχή ενός xsi: μηδενικές ιδιότητες στο αίτημα SOAP. Για παράδειγμα:

```
<methodName>
        <Param xsi:nil="true" />
</methodName>
```

2.3.4 SOAP response

Ο ακόλουθος σκελετός είναι η δομή απάντησης SOAP που επιστρέφεται από τον κεντρικό υπολογιστή που τρέχει Microsoft® SQLXML:

```
<?xml version="1.0" encoding="utf-8" ?>
 <SOAP-ENV:Envelope
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:sqltypes=
       "http://schemas.microsoft.com/SQLServer/2001/12/SOAP/types"
       xmlns:sqlmessage=
         "http://schemas.microsoft.com/SQLServer/2001/12/SOAP/
          types/SqlMessage"
        xmlns:sqlresultstream=
         "http://schemas.microsoft.com/SQLServer/2001/12/SOAP/types
          /SqlResultStream"
        xmlns:tns="http://server/nwind2/soap"
        <!-- additional namespace declarations --> >
  <SOAP-ENV:Body>
    <tns:MethodNameResponse>
      <tns:MethodNameResult
xsi:type="sqlresultstream:SqlResultStream">
          <!--
             the results here depend on how the method
             is configured to return the results (for example,
             XML objects, DataSet objects or a single DataSet)
          -->
      </tns:MethodNameResult>
      <tns:OutputParam>Value</tns:OutputParam>
    </tns:MethodNameResponse>
   </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

<*MethodName*Result> element

Τα αποτελέσματα της μεθόδου επιστρέφονται ως στοιχεία παιδιών του στοιχείου <MethodNameResult>. Το serialization σχήμα εξαρτάται από τον τρόπο με τον οποίο έχει διαμορφωθεί η μέθοδος κατά τη διάρκεια της διαμόρφωσης του εικονικού ονόματος του τύπου SOAP.

- Εάν η μέθοδος διαμορφώνεται για να επιστρέψει τα αποτελέσματα σαν αντικείμενα DataSet (ή ένα μοναδικό αντικείμενο DataSet) στο Microsoft Visual Studio®, τα αποτελέσματα δημοσιεύονται σε συνέχειες με το σχήμα DiffGram. Το σχήμα DiffGram εισάγεται στο τμήμα συνόλου δεδομένων του πλαισίου της Microsoft .NET. Σε αυτήν την περίπτωση, η απάντηση περιλαμβάνει επίσης ένα σχήμα XSD πριν από τα στοιχεία.
- Εάν η μέθοδος διαμορφώνεται για να επιστρέψει τα αποτελέσματα όπως XML αντικείμενα, τα αποτελέσματα δημοσιεύονται σε συνέχειες ως έγκυρα έγγραφα XML (κανένα σχήμα DiffGram). Κάθε έγγραφο XML εμπεριέχεται σε ένα στοιχείο παιδιών <SqlXml>.

<OutputParam> element

Μια αποθηκευμένη διαδικασία μπορεί να έχει τις παραμέτρους output. Κάθε μια από τις τιμές της παραμέτρου output δημοσιεύεται σε συνέχειες στην απάντηση SOAP ως στοιχείο. Το όνομα του στοιχείου είναι το όνομα της παραμέτρου παραγωγής. Αυτό το στοιχείο εμφανίζεται μετά από το στοιχείο <MethodNameResult>.

<returnValue> element

Αυτό το στοιχείο δεν συμπεριλαμβάνεται στη βασική δομή απάντησης SOAP που παρουσιάζεται στην αρχή αυτού του θέματος. Οι αποθηκευμένες διαδικασίες και οι καθορισμένες από το χρήστη λειτουργίες (UDFs) έχουν μια τιμή επιστροφής . Η τιμή επιστροφής επιστρέφεται στο στοιχείο <MethodNameResult> ως στοιχείο παιδιών του <SqlResultCode> (που εμφανίζεται ως αντικείμενο τύπου SqlResultCode στη σειρά αντικειμένου) με αυτήν την εξαίρεση:

Εάν η μέθοδος είναι μια αποθηκευμένη διαδικασία που διαμορφώνεται για να επιστρέψει ένα ενιαίο αντικείμενο DataSet ή είναι ένα UDF που επιστρέφει μια τιμή απλού τύπου, η τιμή επιστροφής δημοσιεύεται σε συνέχειες στην απάντηση SOAP ως στοιχείο <returnValue> που εμφανίζεται μετά από το στοιχείο <MethodNameResult>. Η τιμή επιστροφής δεν είναι διαθέσιμη στη πίνακα αντικειμένων. Σε αυτήν την περίπτωση, η τιμή επιστρέφεται ως παράμετρος όταν καλείται η μέθοδος.

2.3.5 Τα πλεονεκτήματα και τα μειονεκτήματα του SOAP

Πλεονεκτήματα SOAP

- Οι κλήσεις του προγράμματος είναι πιθανότερο να μην περάσουν τα firewalls των servers που διώχνουν εκτός τα αιτήματα από εκείνους που δεν θεωρούνται γνωστές εφαρμογές (μέσω του οριζόμενου μηχανισμού port). Δεδομένου ότι τα αιτήματα HTTP επιτρέπονται συνήθως μέσω των firewalls, τα προγράμματα που χρησιμοποιούν το SOAP για να επικοινωνούν είναι σίγουρα ότι μπορούν να επικοινωνήσουν με άλλα προγράμματα οπουδήποτε.

- Το SOAP είναι ένα ευπροσάρμοστο πρωτόκολλο που μπορεί να μεταφέρει τα μηνύματα σε πολλά πρωτόκολλα μεταφορών όπως JMS και SMTP, όχι μόνο HTTP.
- Ένα όφελος που θα αποκομισθεί από τη χρησιμοποίηση του SOAP είναι η απλότητα. Το SOAP είναι απλώς XML και HTTP που συνδυάζονται για να σταλθούν και να ληφθούν τα μηνύματα μέσω του Διαδικτύου. Δεν περιορίζεται με τη γλώσσα εφαρμογής (java, C#, Perl) ή την πλατφόρμα (Windows, UNIX, MAC), και αυτό το καθιστά πιό ευπροσάρμοστο από άλλες λύσεις.

Μειονεκτήματα SOAP

- Λόγω του XML σχήματος, το SOAP μπορεί να είναι αρκετά πιό αργό από τις ανταγωνιστικές τεχνολογίες υλικολογισμικού όπως η CORBA. Αυτό μπορεί να μην είναι ένα ζήτημα όταν στέλνονται μόνο μικρά μηνύματα. Για να βελτιώσει την απόδοση για τη ειδική περίπτωση της XML με τα ενσωματωμένα δυαδικά αντικείμενα, εισήχθηκε ο μηχανισμός βελτιστοποίησης μετάδοσης μηνυμάτων.
- Βασιζόμενο στο HTTP ως πρωτόκολλο μεταφορών και χωρίς τη χρήση της WS-Addressing ή ενός ESB, οι ρόλοι της αλληλεπίδρασης των συμβαλλόμενων μερών καθορίζονται. Μόνο ένα συμβαλλόμενο μέρος (ο πελάτης) μπορεί να χρησιμοποιήσει τις υπηρεσίες του άλλου. Οι υπεύθυνοι για την ανάπτυξη πρέπει να χρησιμοποιήσουν polling αντί της notification σε αυτές τις κοινές περιπτώσεις.
- Μπορεί επίσης να επιβραδύνει την επεξεργασία επειδή όταν χρησιμοποιείται το HTTP ως πρωτόκολλο μεταφορών, το firewall που σχεδιάζεται για να επιτρέψει το web-browsing πρέπει να εκτελέσει μια πιό λεπτομερή ανάλυση των πακέτων HTTP, τα οποία μπορούν να επιβραδύνουν τη διαδικασία.

2.4 WSDL

Το WSDL είναι τ΄ αρχικά του Web Services Description Language. Είναι δυνατό να θεωρηθεί ότι ένα WSDL αρχείο είναι ένα XML έγγραφο που περιγράφει μια σειρά μηνυμάτων SOAP αλλά και πως γίνεται η ανταλλαγή των μηνυμάτων. Με άλλα λόγια το WSDL είναι για το SOAP ότι το IDL για το CORBA. Αφού το WSDL είναι XML είναι κατανοητό και τροποποιήσιμο από τον άνθρωπο αλλά στις περισσότερες περιπτώσεις παράγεται και καταναλώνεται από το λογισμικό.

Για να γίνει κατανοητή η αξία του WSDL δίνεται το ακόλουθο παράδειγμα. Για να κληθεί μια μέθοδος SOAP πρέπει τα request και τα response μηνύματα να έχουν συμβατότητα στα δεδομένα. Εάν χρησιμοποιηθούν μερικά δείγματα SOAP μηνυμάτων για να γραφτεί αίτηση για παραγωγή και κατανάλωση μηνυμάτων που μοιάζουν με τα δείγματα αυτό τα κάνει επιρρεπή στα λάθη (error prone). Για παράδειγμα μπορεί ο κωδικός ενός πελάτη να έχει γραφεί ως int ενώ μπορεί να είναι string. Το WSDL ορίζει τι θα πρέπει να περιέχει ένα αίτημα και πως πρέπει να μοιάζει το μήνυμα απάντησης με σαφή συμβολισμό.

Ο συμβολισμός που χρησιμοποιεί ένα WSDL αρχείο για να περιγράφει τις μορφές των μηνυμάτων βασίζεται στο πρότυπο XML schema το οποίο σημαίνει ότι είναι ουδέτερη γλώσσα προγραμματισμού και βασίζεται σε πρότυπα που το κάνει ιδανικό για περιγραφή interface των XML υπηρεσιών ιστού που είναι προσβάσιμα από μια μεγάλη ποικιλία από πλατφόρμες και γλώσσες προγραμματισμού. Εκτός από την περιγραφή του περιεχομένου του μηνύματος το WSDL ορίζει που θα είναι διαθέσιμη μια υπηρεσία και ποιο πρωτόκολλο επικοινωνίας θα χρησιμοποιεί για να

επικοινωνεί. Αυτό σημαίνει ότι το WSDL αρχείο ορίζει τα πάντα που απαιτούνται να γραφτεί ένα πρόγραμμα που θα δουλεύει με μια XML υπηρεσία ιστού. Υπάρχουν διάφορα αρχεία διαθέσιμα για να διαβάσουν ένα WSDL αρχείο και να παράγουν τον κώδικα που απαιτείται για να επικοινωνήσει με μια XML υπηρεσία ιστού. Μερικά από τα πιο ικανά από αυτά τα εργαλεία βρίσκονται στο Microsoft Visual Studio.NET.

Πολλές τρέχουσες εργαλειοθήκες SOAP περιλαμβάνουν εργαλεία για τη δημιουργία WSDL αρχείων από τα υπάρχουσα interface προγράμματα αλλά υπάρχουν λίγα εργαλεία για τη συγγραφή WSDL κώδικα άμεσα και τα εργαλεία υποστήριξης για το WSDL δεν είναι ακόμα όσο ολοκληρωμένα θα έπρεπε.

2.4.1 Περιγραφή

Το WSDL περιγράφει τις υπηρεσίες σαν συλλογές από endpoint δικτύου ή ports. Η προδιαγραφή WSDL παρέχει μια XML μορφή για έγγραφα γι' αυτό το σκοπό. Οι αφηρημένοι ορισμοί των ports και των μηνυμάτων είναι χωρισμένοι από την συγκεκριμένη χρήση ή το στιγμιότυπο τους επιτρέποντας την επαναχρησιμοποίηση αυτών των ορισμών. Μια port ορίζεται συνδέοντας μια διεύθυνση δικτύου με ένα επαναχρησιμοποιούμενο binding και μια συλλογή από ports ορίζει μια service. Τα μηνύματα είναι μια αόριστη περιγραφή πληροφορίας που ανταλλάσσεται και οι τύποι port είναι αφηρημένες συλλογές από μεθόδους που υποστηρίζονται. Το συγκεκριμένο πρωτόκολλο και η μορφή των προδιαγραφών της πληροφορίας για ένα συγκεκριμένο τύπο port αποτελούν ένα επαναχρησιμοποιήσιμο binding όπου οι μέθοδοι και τα μηνύματα είναι δεσμευμένα σ' ένα συγκεκριμένο πρωτόκολλο δικτύου και μορφή μηνυμάτων. Μ' αυτόν τον τρόπο το WSDL περιγράφει ένα δημόσιο interface στην υπηρεσία Ιστού. Το WSDL συχνά χρησιμοποιείται σε συνδυασμό με το SOAP και ένα XML schema για να παρέχει υπηρεσίες ιστού μέσα στο Internet. Ένα client πρόγραμμα συνδέεται σε μια υπηρεσία ιστού που μπορεί να διαβάσει ένα WSDL αρχείο για να καθορίσει ποιες μέθοδοι είναι διαθέσιμοι στον server. Τυχών ειδικές περιπτώσεις τύπων δεδομένων που χρησιμοποιούνται είναι ενσωματωμένες στο αρχείο WSDL με τη μορφή XML schema. Ο client στη συνέχεια μπορεί να χρησιμοποιήσει SOAP για να κάνει κλήση σε μια από τις μεθόδους που περιλαμβάνονται στο αρχείο WSDL χρησιμοποιώντας XML ή HTTP.

2.4.2 Ιστορική αναδρομή

- Το WSDL 1.0 (Σεπτέμβριος του 2000) έχει αναπτυχθεί από τις IBM, Microsoft και Ariba για να περιγράψει τις υπηρεσίες ιστού και την εργαλειοθήκη του SOAP. Την δημιούργησαν συνδυάζοντας δύο περιγραφικές γλώσσες υπηρεσιών την NASSL (Network Application Service Specification Language) της IBM και την SDL (Service Description Language) της Microsoft.
- Το WSDL 1.1 δημιουργήθηκε τον Μάρτιο του 2001 είναι η επισημοποίηση του WSDL 1.0. Δεν υπήρχαν σημαντικές αλλαγές μεταξύ των εκδόσεων 1.0 και 1.1 .
- Το WSDL 1.2 (Ιούνιος του 2003) είναι ακόμα ένα λειτουργικό σχέδιο στο W3C. Σύμφωνα με αυτό το WSDL 1.2 είναι πιο εύκολο και πιο ευέλικτο για τους προγραμματιστές από τη προηγούμενη έκδοση. Το WSDL 1.2 επιχειρεί την άρση των μη διαλειτουργικών χαρακτηριστικών και επίσης ορίζεται το καλύτερο HTTP 1.1 binding. Το WSDL 1.2 δεν υποστηρίχτηκε από την πλειοψηφία των SOAP Server/ vendors.

- Το WSDL 2.0 έγινε σύσταση του W3C τον Ιούνιο του 2007. Το WSDL 1.2 μετονομάστηκε WSDL 2.0 γιατί είχε σημαντικές διαφορές από το WSDL 1.1. Οι αλλαγές ήταν :

  1. Η προσθήκη επιπλέον σημασιολογίας στην περιγραφική γλώσσα.
  2. Απομάκρυνση από τη κατασκευή μηνυμάτων.
  3. Δεν υπάρχει υποστήριξη για υπερφόρτωση τελεστών.
  4. Οι τύποι ports μετονομάστηκαν σε διεπαφές.
  5. Το port μετονομάστηκε σε endpoint.

2.4.3 τα στοιχεία του WSDL

Τα στοιχεία του WSDL είναι:
<Services>: Η υπηρεσία μπορεί να θεωρηθεί ως ένα δοχείο για ένα σύνολο των λειτουργιών του συστήματος που έχουν εκτεθεί σε Web-based πρωτόκολλο.
<Port/Endpoints>: Δεν κάνει τίποτα περισσότερο από να ορίσει τη διεύθυνση ή το σημείο σύνδεσης σε μια υπηρεσία Web. Συνήθως εκπροσωπείται από ένα απλό HTTP URL string.
<Binding>: Το binding ορίζει το interface καθώς και το στυλ του SOAP binding (RPC/Document) και τη μεταφορά (πρωτόκολλο SOAP). Το τμήμα binding ορίζει ακόμα και τις λειτουργίες.
<Port types>: Το στοιχείο port type μετονομάστηκε σε interface στο WSDL 2.0 και ορίζει μια Web υπηρεσία ,τις λειτουργίες που μπορούν να εκτελέσουν καθώς και τα μηνύματα που χρησιμοποιούνται για την εκτέλεση μιας λειτουργίας.
<Types>: Ο σκοπός των τύπων σε WSDL είναι να περιγράψει τα δεδομένα, ένα XML schema χρησιμοποιείται (είτε εσωκλειόμενο είτε με αναφορά) γι' αυτό το σκοπό.
<Messages>: Συνήθως ένα μήνυμα αντιστοιχεί σε μια λειτουργία. Το μήνυμα περιέχει τις πληροφορίες που απαιτούνται για την εκτέλεση κάθε πράξης. Κάθε μήνυμα αποτελείται από ένα ή παραπάνω λογικά μέρη.

Για να γίνουν κατανοητά θα παρουσιαστεί το ακόλουθο παράδειγμα που όχι μόνο χρησιμοποιεί σχεδόν κάθε όψη του WSDL , έχει ένα κομμάτι XML schema αλλά αναδεικνύει και τα πλεονεκτήματα του SOAP binding στο WSDL.

Το **<definitions>** στοιχείο περιγράφει ένα set από σχετικές υπηρεσίες.

Το στοιχείο **<types>** επιτρέπει τον προσδιορισμό από χαμηλού επιπέδου πληκτρολόγησης πληροφορίας για ένα μήνυμα ή για το περιεχόμενο μιας διαδικασίας. Διαφορετικοί μηχανισμοί επιτρέπονται λόγο της επέκτασης του namespace αλλά το XML schemas είναι πιθανό να είναι επιλογή για τους περισσότερους χρήστες. Και αυτό χρησιμοποιείται στο παράδειγμα. Αυτό ορίζει ένα απλό στοιχείο μοντέλου περιεχομένου που ταιριάζει στην ανταλλαγή δειγμάτων στο ex1 και ex2. Το WSDL παρέχει ένα σύστημα για την εισαγωγή τύπου δεδομένων με προδιαγραφές που βρίσκονται ως ξεχωριστοί πόροι. Μπορεί να υπάρξουν μερικοί τέτοιοι πόροι σε περιπτώσεις περίπλοκων μηνυμάτων σε πολλαπλή χρήση domains.

Το **<messages>** στοιχείο ορίζει τη μορφή πληροφορίας από κάθε ατομική μετάδοση στην επικοινωνία. Στην περίπτωση του παραδείγματος ένα μήνυμα αναπαριστά το Endorsing Boarder request και ένα άλλο το response. Στο παράδειγμα αυτό είναι μια απλή εντολή όπου το σώμα του μηνύματος είναι συγκεκριμένο στοιχείο από το schema στην περιοχή των τύπων. Ο χωρισμός μιας μετάδοσης σε μέρη μηνυμάτων εξαρτάται στη λογική άποψη των δεδομένων. Για παράδειγμα εάν η μετάδοση είναι μια κλήση απομακρυσμένης διαδικασίας το μήνυμα μπορεί να χωρίζεται σε πολλαπλά τμήματα. Ένα εκ των οποίον είναι το όνομα της διαδικασίας, τα metadata και το

υπόλοιπο είναι οι παράμετροι της διαδικασίας. Υπάρχει φυσικά μια σχέση μεταξύ της διακριτικότητας των στοιχείων τον καθορισμό των ομάδων και τη λεπτομερή ανάλυση του μηνύματος σε τμήματα.

Το **<port types>** στοιχείο ομαδοποιεί μηνύματα που σχηματίζουν μια λειτουργία. Για παράδειγμα μπορεί να έχουμε ένα Endorsing Boarder request το οποίο ενεργοποιεί ένα Endorsing Boarder response ή σε περίπτωση ενός λάθους ή ενός exception ένα Endorsing Boarder Fault. Η συγκεκριμένη ανταλλαγή είναι ομαδοποιημένη σ' ένα WSDL port type. Όπως μπορούμε να δούμε η σχέση με τα μηνύματα είναι φτιαγμένη από qualified name reference.

Υπάρχουν μόνο 4 μορφές λειτουργίας που υποστηρίζονται στο WSDL: one-way, request-response, solicit-response και notification. Οι δυο τελευταίες είναι αντίστροφη των 2πρώτων. Η μόνη διαφορά είναι εάν το υπό εξέταση είναι αυτό που δέχεται ή στέλνει δεδομένα ή τερματίζει το αρχικό μήνυμα. Βασικά το WSDL υποστηρίζει μονής κατεύθυνσης (one-way) και διπλής κατεύθυνσης (request-response, solicit-response) port types. Τα faults (λάθη) υποστηρίζονται μόνο στα αμφίδρομα port type.

Το WSDL έγγραφο μετακινήθηκε από το συγκεκριμένο και φυσικό (data-typing) στο αφηρημένο και λογικό (messages and port types). Με κάποιες αναφορές μεταξύ των δυο. Το **<binding>** στοιχείο είναι το bit το οποίο δυνατά παρέχει τη σύνδεση μεταξύ λογικού και φυσικού μοντέλου. Στη συγκεκριμένη περίπτωση παίρνει την operation που έχει ορισθεί μέσω του αφηρημένου port type και το συνδέει με συγκεκριμένη περιγραφή του πως θα μεταφερθεί μέσω SOAP.

Το δείγμα binding ορίζει το Get Endorsing Boarder Portly σαν να έχει SOAP Style Document. Το στυλ μπορεί να είναι RPC ή Document. Το προηγούμενο δείχνει μια πιο διαδικασιακή κλήση προς την επικοινωνία και το τελευταίο μια message-exchange κατεύθυνση. Βέβαια η διαχωριστική γραμμή μεταξύ αυτών είναι αρκετά ευρεία.

Το binding που χρησιμοποιείται στο παράδειγμα ακόμα καθορίζει και τη μεταφορά του δικτύου σαν HTTP-SOAP που μπορεί να μεταδοθεί και με άλλα μέσα όπως SMTP. Τα στοιχεία <soap: operation> ορίζουν τη χαρτογράφηση κάθε μηνύματος στο port type για τον ορισμό των μεταδόσεων SOAP που τους ωθούν. Έχει ακόμα καθοριστεί μια SOAP Action η οποία απαιτείται για το SOAP σε συνδυασμό με HTTP. Η δεδομένη τιμή πρέπει να χρησιμοποιείται στις κεφαλίδες του HTTP των πραγματικών μηνυμάτων προκειμένου να σηματοδοτήσει την "πρόθεση" του μηνύματος. Αυτό κάποια μέρα θα επιτρέψει την έξυπνη χρήση των proxy και firewalls στην κυκλοφορία του SOAP.

Το τελευταίο στοιχείο **<services>** ορίζει μια φυσική τοποθεσία για το endpoint της επικοινωνίας . Χρησιμοποιεί το port type και το binding που ορίστηκε νωρίτερα και βασικά δίνει την διεύθυνση του ιστού ή URI για συγκεκριμένο ένα φορέα παροχής της υπηρεσίας που περιγράφεται.

2.4.4Παράδειγματα

Ακολουθούν παραδείγματα τα οποία θα βοηθήσουν στην καλύτερη κατανόηση του WSDL.

Παράδειγμα 1: SOAP request

```
POST /EndorsementSearch HTTP/1.1
Host: www.snowboard-info.com
Content-Type: text/xml; charset="utf-8"
Content-Length: 261
SOAPAction: "http://www.snowboard-info.com/EndorsementSearch"
```

```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <m:GetEndorsingBoarder xmlns:m="http://namespaces.snowboard-info.com">
   <manufacturer>K2</manufacturer>
   <model>Fatbob</model>
  </m:GetEndorsingBoarder>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Παράδειγμα 2: SOAP response

```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <m:GetEndorsingBoarderResponse xmlns:m="http://namespaces.snowboard-info.com">
   <endorsingBoarder>Chris Englesmann</endorsingBoarder>
  </m:GetEndorsingBoarderResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Παράδειγμα 3 : Μια περιγραφή WSDL για μια ερώτηση επικύρωσης Snowboarding

```
<?xml version="1.0"?>

<definitions name="EndorsementSearch"

targetNamespace="http://namespaces.snowboard-info.com"

 xmlns:es="http://www.snowboard-info.com/EndorsementSearch.wsdl"

xmlns:esxsd="http://schemas.snowboard-info.com/EndorsementSearch.xsd"

xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"

xmlns="http://schemas.xmlsoap.org/wsdl/">

<types>

    <schema targetNamespace="http://namespaces.snowboard-info.com"

        xmlns="http://www.w3.org/1999/XMLSchema">

    <element name="GetEndorsingBoarder">

        <complexType>

            <sequence>
```

```
      <element name="manufacturer" type="string"/>
      <element name="model" type="string"/>
    </sequence>

            </complexType>

      </element>

      <element name="GetEndorsingBoarderResponse">
            <complexType>
                  <all>
                  <element name="endorsingBoarder" type="string"/>
                  </all>

            </complexType>

      </element>

<element name="GetEndorsingBoarderFault">
  <complexType>
    <all>
     <element name="errorMessage" type="string"/>
    </all>

 </complexType>

 </element>
</schema>

</types>

<message name="GetEndorsingBoarderRequest">

      <part name="body" element="esxsd:GetEndorsingBoarder"/>

</message>

<message name="GetEndorsingBoarderResponse">

      <part name="body" element="esxsd:GetEndorsingBoarderResponse"/>

</message>

<portType name="GetEndorsingBoarderPortType">

      <operation name="GetEndorsingBoarder">

      <input message="es:GetEndorsingBoarderRequest"/>

      <output message="es:GetEndorsingBoarderResponse"/>
```

```
        <fault message="es:GetEndorsingBoarderFault"/>

        </operation>

    </portType>


    <binding name="EndorsementSearchSoapBinding"
        type="es:GetEndorsingBoarderPortType">

        <soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http"/>

        <operation name="GetEndorsingBoarder">

            <soap:operation

             soapAction="http://www.snowboard-info.com/EndorsementSearch"/>

            <input>

             <soap:body use="literal"

namespace="http://schemas.snowboard-info.com/EndorsementSearch.xsd"/>

            </input>

            <output>
      <soap:body use="literal"


namespace="http://schemas.snowboard-info.com/EndorsementSearch.xsd"/>

             </output>

            <fault>

             <soap:body use="literal"

namespace="http://schemas.snowboard-info.com/EndorsementSearch.xsd"/>

            </fault>

        </operation>

    </binding>


    <service name="EndorsementSearchService">
```

```
<documentation>snowboarding-info.com Endorsement Service</documentation>Â

        <port name="GetEndorsingBoarderPort"
      binding="es:EndorsementSearchSoapBinding">

            <soap:address
location="http://www.snowboard-info.com/EndorsementSearch"/>

        </port>

  </service>

</definitions>
```

ΓΕΝΙΚΑ ΓΙΑ ΤΟ ΠΑΡΑΔΕΙΓΜΑ

Μερικές γενικές παρατηρήσεις σχετικά με το WSDL παραδείγμα μας, αναφέρονται εδώ. Μπορείτε να δείτε ότι το WSDL στηρίζεται σε μεγάλο βαθμό σε XML namespaces. Τα XML namespaces τα οποία δίνονται στο στοιχείο <definitions> αναφέρονται στο χαρακτηριστικό target Namespace που εξ ορισμού συνδέεται με τα ονόματα που χρησιμοποιούνται για άλλα υψηλού επιπέδου WSDL στοιχεία. Οι προγραμματιστές μπορούν να χρησιμοποιήσουν ονόματα για να αναφερθούν σ' αυτά τα στοιχεία χρησιμοποιώντας προθέματα (prefixes) από τα συγκεκριμένα namespace που δηλώνονται στο πεδίο εφαρμογής. Πρέπει να σημειωθεί ότι τα default namespaces δεν εφαρμόζονται σε μη-πρόθεμα (un-prefixed) ονόματα μέσα στα WSDL χαρακτηρίστηκα. Αυτό είναι σύμφωνο σε άλλα μέρη όπου ο μηχανισμός των XML namespaces τον έχουν δανειστεί για χρήση σε αποσαφηνισμένα ονόματα. Επίσης τα XML namespaces χρησιμοποιούνται για να συνδέουν τα WSDL στοιχεία με το data-typing που παρέχεται στο <types> στοιχείο.

- Στο παράδειγμα χρησιμοποιούμε default namespace http://schemas.xmlsoap.org/wsdl , για να αναφέρουμε τα επίσημα στοιχεία του WSDL. Παρόλα αυτά υπάρχει η δυνατότητα επέκτασης των στοιχείων του πυρήνα σε άλλα namespaces.
- Συνολικά το παράδειγμα είναι αρκετά απλό. Περιγράφει την επικοινωνία που αποτελείται από σύντομες μεταδόσεις SOAP με δυο σειρές εισόδου και μια έξοδο σε κάθε operation. Το WSDL μπορεί τόσο απλά να ορίσει πολλαπλά port types που αποτελούνται από μυριάδες μηνύματα που χρησιμοποιούν πλήρες εύρος από XML schemas. Κατόπιν τουλάχιστον βραχυπρόθεσμα πιο απλή επικοινωνία μεταξύ υπηρεσίας και χρηστών είναι πιθανό να επιτευχθεί.

2.5 UDDI

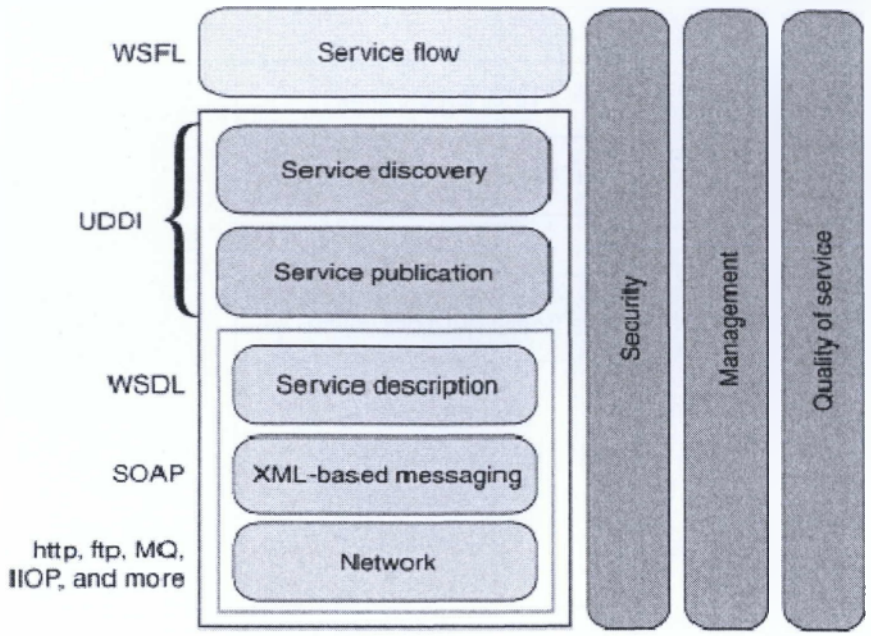
Το UDDI είναι βασισμένο στα υπάρχοντα πρότυπα, όπως η Extensible Markup Language (XML) και το απλό πρωτόκολλο πρόσβασης αντικειμένου (SOAP). Όλες οι συμβατές εφαρμογές UDDI υποστηρίζουν την προδιαγραφή UDDI. Η δημόσια προδιαγραφή αναπτύσσεται σε μια ανοικτή διαδικασία από τα μέλη της οργάνωσης. Η πρόθεση είναι να παραχθούν και να εφαρμοστούν τρεις διαδοχικές εκδόσεις της προδιαγραφής πριν εξελιχθεί μελλοντικά ως ανάπτυξη ανεξάρτητου σώματος προτύπων. Η UDDI έκδοση 1 της προδιαγραφής δημοσιεύθηκε το Σεπτέμβρη του 2000, και η έκδοση 2 δημοσιεύθηκε τον Ιούνιο του 2001. Η έκδοση 3 έγινε διαθέσιμη το 2002. Η έκδοση 1 καθιέρωσε μια βάση για το registry, ενώ η έκδοση 2 πρόσθεσε τα χαρακτηριστικά γνωρίσματα όπως τις επιχειρησιακές σχέσεις. Η έκδοση 3 συνεχίζει να απευθύνεται σε περιοχές σημαντικές στην τρέχουσα ανάπτυξη των υπηρεσιών Ιστού, όπως η ασφάλεια, η βελτιωμένη διεθνοποίηση, η διαλειτουργικότητα του registry, και οι διάφορες βελτιώσεις εφαρμογών που επιτρέπουν περαιτέρω βελτιώσεις σχεδίασης.

Πιο συγκεκριμένα όμως σε ότι αφορά την ιστορία του UDDI. Το UDDI γράφτηκε τον Αύγουστο του 2000, σε μία εποχή που οι συντάκτες είχαν το όραμα ενός κόσμου στον οποίο οι καταναλωτές των υπηρεσιών Ιστού θα μπορούσαν να συνδεθούν με τους παρόχους μέσω ενός δημόσιου ή ιδιωτικού δυναμικού συστήματος μεσιτειών. Σε αυτό το όραμα, ο οποιοσδήποτε χρειάζεται μια υπηρεσία όπως η αυθεντικοποίηση μιας πιστωτικής κάρτας θα πάει στο μεσίτη υπηρεσιών και θα επιλέξει ένα σύστημα που υποστηρίζει το επιθυμητό SOAP ή άλλη διεπαφή υπηρεσίας που ικανοποιεί άλλα κριτήρια. Σε έναν τέτοιο κόσμο, ο δημόσια χρησιμοποιημένος κόμβος UDDI ή ο μεσίτης θα ήταν ζωτικής σημασίας. Για τον καταναλωτή, οι δημόσιοι ή ανοικτοί μεσίτες θα επέστρεφαν μόνο τις υπηρεσίες που απαριθμούνται για τη δημόσια ανακάλυψη από άλλες, ενώ για έναν παραγωγό υπηρεσιών, που παίρνει μια καλή τοποθέτηση στη μεσιτεία-στήριξη στα μεταδεδομένα -θα ήταν βασικό για την αποτελεσματική τοποθέτηση.

Το UDDI περιλήφθηκε στα πρότυπα διαλειτουργικότητας υπηρεσιών Ιστού (WS-ι) ως κεντρικός στυλοβάτης της υποδομής υπηρεσιών Ιστού, και οι προδιαγραφές UDDI υποστήριξαν ένα δημόσια προσιτό καθολικό επιχειρησιακό ληξιαρχείο. Μέσα στο οποίο ένα σύστημα χτίστηκε γύρω από τον UDDI-οδηγούμενο μεσίτη υπηρεσιών.

Το UDDI δεν έχει υιοθετηθεί ευρέως με τον τρόπο που οι σχεδιαστές του είχαν ελπίσει. Η IBM, η Microsoft, και η SAP ανήγγειλαν ότι έκλειναν τους δημόσιους κόμβους UDDI τον Ιανουάριο του 2006. Η ομάδα που καθορίζει τον ορισμό του UDDI, OASIS Universal Description, Discovery, and Integration (UDDI) Specification Technical Committee ψήφισε να ολοκληρώσει την εργασία της στα τέλη του 2007 και να έχουν κλείσει. Τον Σεπτέμβρη του 2010, η Microsoft ανήγγειλε ότι θα αφαιρούσαν τις υπηρεσίες UDDI από τις μελλοντικές εκδόσεις του λειτουργικού συστήματος Windows. Αντ' αυτού, αυτή η ικανότητα θα κινούταν προς τον Biztalk.

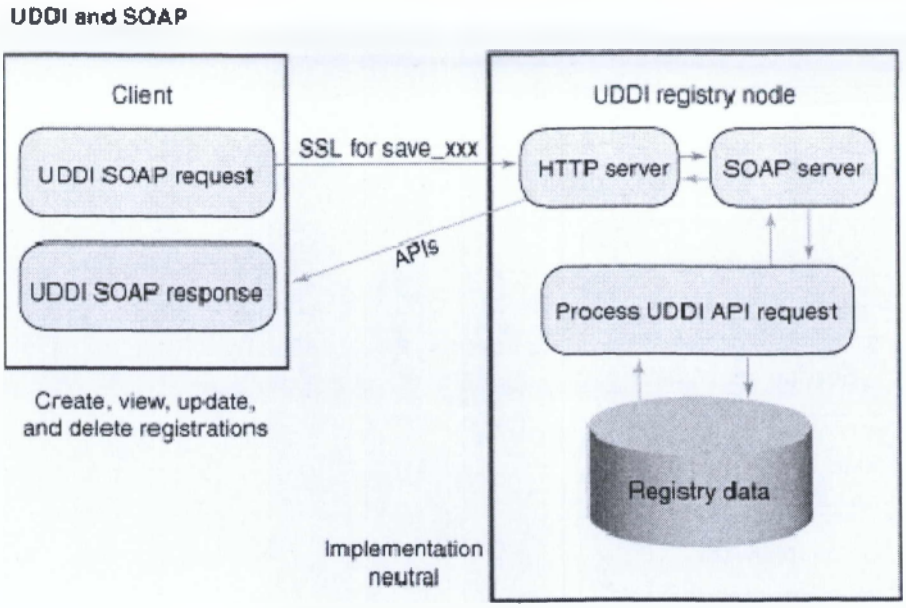
Μερικοί βεβαιώνουν ότι η πιό κοινή θέση που μπορεί να βρεθεί ένα σύστημα UDDI είναι μέσα σε μια επιχείρηση όπου μπορεί να χρησιμοποιηθεί για δυναμική διασύνδεση συστημάτων πελατών και εφαρμογών. Θα έλεγαν ότι ένα μεγάλο μέρος των μεταδεδομένων αναζήτησης που επιτρέπονται στο UDDI δεν χρησιμοποιείται για αυτόν τον σχετικά απλό ρόλο.



Σχήμα 2. 3 Ο διαστρωματωμένος σωρός των υπηρεσιών ιστού με UDDI

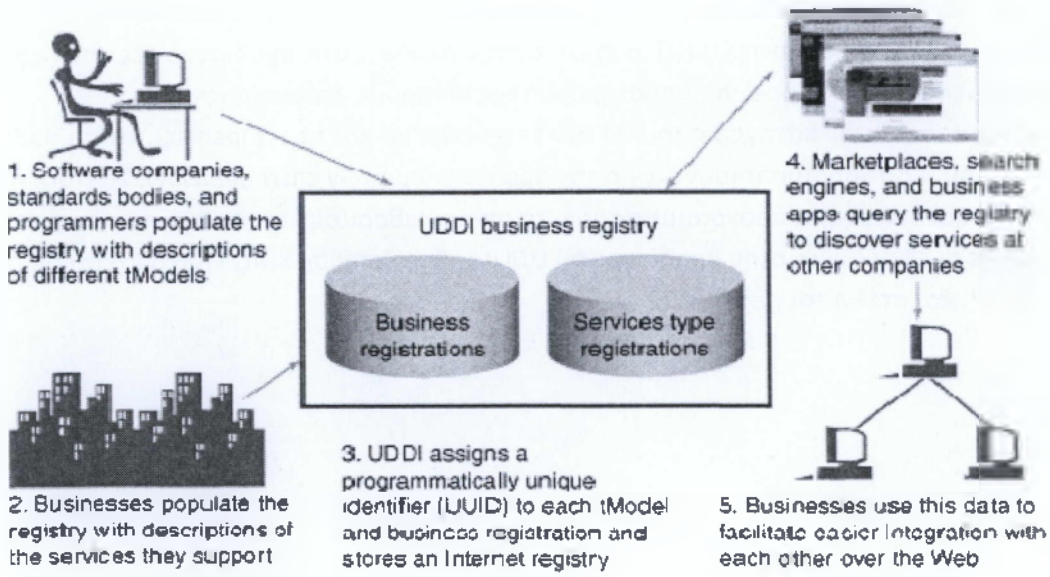
Το UDDI στηρίζεται σε ένα στρώμα μεταφορών δικτύων και ένα βασισμένο σε SOAP στρώμα μηνύματος XML. Οι γλώσσες περιγραφής υπηρεσιών, όπως η γλώσσα περιγραφής υπηρεσιών Ιστού (Web Services Description Language ,WSDL), παρέχουν ένα ομοιόμορφο λεξιλόγιο XML (παρόμοιο με μια διαλογική γλώσσα στοιχείων (Interactive Data Language,IDL)) για χρήση στην περιγραφή των υπηρεσιών Ιστού και των διεπαφών τους. Μπορείτε να χτίσετε επάνω σε αυτό το γενικό θεμέλιο με την προσθήκη των βαλμένων σε στρώσεις ικανοτήτων, όπως οι περιγραφές ροής της δουλειάς των υπηρεσιών Ιστού χρησιμοποιώντας τη γλώσσα ροής υπηρεσιών Ιστού (Web Services Flow Language ,WSFL), την ασφάλεια, τη διαχείριση, και τα χαρακτηριστικά γνωρίσματα όπως η ποιότητα υπηρεσιών, τα οποία εξετάζουν την αξιοπιστία και τη διαθεσιμότητα συστημάτων.

Ένα UDDI registry περιέχει προγραμματιστικά προσβάσιμες περιγραφές των επιχειρήσεων και των υπηρεσιών που υποστηρίζει. Περιέχει επίσης αναφορές στις προδιαγραφές βιομηχανίας που μια υπηρεσία Ιστού μπορεί να υποστηρίξει, τους ορισμούς ταξονομίας (που χρησιμοποιούνται για τη κατηγοριοποίηση των επιχειρήσεων και τις υπηρεσίες), και τα συστήματα προσδιορισμού (που χρησιμοποιούνται για τον προσδιορισμό των επιχειρήσεων). Το UDDI παρέχει ένα πρότυπο και ένα σχήμα προγραμματισμού, τα οποία καθορίζουν τους κανόνες για με το registry. Όλες οι εφαρμογές στην προδιαγραφή UDDI καθορίζεται σε XML, εμπεριέχεται σε έναν φάκελο SOAP, και στέλνεται μέσω HTTP.



Σχήμα 2. 4

Το σχήμα 2.4 επεξηγεί τη μεταφορά μηνυμάτων UDDI, από το αίτημα SOAP του πελάτη μέσω HTTP σε έναν registry κόμβο και το αντίστροφο. Ο SOAP server του registry server χειρίζεται το SOAP-UDDI μήνυμα, το επεξεργάζεται, και επιστρέφει μια SOAP απάντηση στον πελάτη. Ως θέμα της registry πολιτικής, τα αιτήματα πελατών που είναι πιθανό να τροποποιήσουν τα δεδομένα πρέπει για να είναι ασφαλή και οι συναλλαγές να είναι αξιόπιστες.



Σχήμα 2. 5

Το σχήμα 2.5 επεξηγεί πώς ένα registry UDDI είναι γεμάτο μεταδεδομένα και πώς οι πελάτες ανακαλύπτουν και χρησιμοποιούν αυτές τις πληροφορίες. Ένα registry UDDI στηρίζεται στα δεδομένα που παρέχονται από τους πελάτες του. Υπάρχουν διάφορα βήματα για να καταστήσουν τα δεδομένα χρήσιμα σε UDDI. Όπως φαίνεται στο βήμα 1, οι χρήσιμες πληροφορίες έκδοσης στο registry αρχίζουν όταν καθορίζουν οι εταιρείες λογισμικού και οι οργανισμοί προτύπων τις προδιαγραφές σχετικά με μια βιομηχανία ή μια επιχείρηση, τις οποίες εγγράφουν σε UDDI. Αυτα γνωστά ως technical models η tModels.

Στο βήμα 2, οι εταιρίες καταχωρούν επίσης τις περιγραφές των επιχειρήσεών τους και των υπηρεσιών που προσφέρουν. Ένα regisrty UDDI παρακολουθεί όλων τις οντότητες αναθέτοντας στην καθεμία τους ένα μοναδικό προγραμματιστικό προσδιοριστικό, γνωστό ως Unique Universal Identifier (UUID)όπως φαίνεται στο βήμα 3. Ένα κλειδί UUID είναι εγγυημένο ότι είναι μοναδικό και δεν αλλάζει ποτέ μέσα σε ένα registry UDDI. Αυτά τα κλειδιά μοιάζουν με μια μορφοποιημένη τυχαία δεκαεξαδική σειρά (παραδείγματος χάριν, C0B9FE13-179F-413D-8A5B-5004DB8E5BB2). Μπορούν να χρησιμοποιηθούν για να παραπέμψουν την οντότητα με την οποία συνδέονται. Τα κλειδιά UUID που δημιουργούνται σε ένα registry είναι σημαντικά μόνο μέσα στο πλαίσιο εκείνου του registry.

Άλλοι πελάτες, όπως οι e-Marketplaces, ,οι μηχανές αναζήτησης, και οι επιχειρηματικές εφαρμογές, στο βήμα 4, χρησιμοποιούν ένα registry UDDI για να ανακαλύψουν τις υπηρεσίες που προκαλούν ενδιαφέρον. Στη συνέχεια, οι άλλες επιχειρήσεις μπορούν να επικαλεσθούν αυτές τις υπηρεσίες, που επιτρέπουν την απλή και δυναμική ολοκλήρωση όπως διευκρινίζεται στο βήμα 5.

Τα στοιχεία σε ένα registry UDDI μπορούν να διαιρεθούν εννοιολογικά σε τέσσερις κατηγορίες, κάθε μια από τις οποίες αντιπροσωπεύει μια κορυφαία οντότητα σε UDDI. Σε κάθε τέτοια οντότητα ορίζεται UUID της και μπορεί πάντα να βρεθεί στα πλαίσια εκείνου του registry UDDI με αυτό το προσδιοριστικό:

Τεχνικά πρότυπα
Επιχειρήσεις
Υπηρεσίες επιχείρησης
Συνδέσεις υπηρεσιών

Οι πληροφορίες registry για τις επιχειρήσεις και τις υπηρεσίες μπορούν να θεωρηθούν μέσα τρεις ομάδες: άσπρες, κίτρινες, και πράσινες σελίδες

- Άσπρες σελίδες (White Pages) - διεύθυνση, επαφή, και γνωστά προσδιοριστικά
- Κίτρινες σελίδες (Yellow Pages)- βιομηχανικές κατηγοριοποιήσεις βασισμένες στις τυποποιημένες ταξονομίες
- Πράσινες σελίδες (Green Pages)- τεχνικές πληροφορίες για τις υπηρεσίες που εκτίθενται από από την επιχείρηση

Πιο αναλυτικά:

- White Pages : Οι άσπρες σελίδες δίνουν τις πληροφορίες για την επιχείρηση που παρέχει την υπηρεσία. Αυτό περιλαμβάνει το όνομα της επιχείρησης και μια περιγραφή - ενδεχομένως σε πολλές γλώσσες. Χρησιμοποιώντας αυτές τις πληροφορίες, είναι δυνατό να βρεθεί μια υπηρεσία για την οποία κάποιες πληροφορίες να είναι ήδη γνωστές (για παράδειγμα, ο εντοπισμός μιας υπηρεσίας με βάση το όνομα του προμηθευτή). Παρέχονται επίσης τα στοιχεία επαφής για την επιχείρηση – για παράδειγμα η επιχειρησιακή διεύθυνση ,ο αριθμός τηλεφώνου και άλλες πληροφορίες όπως το Dun & Bradstreet Universal Numbering System number.

- Yellow Pages : Οι κίτρινες σελίδες παρέχουν μια ταξινόμηση της υπηρεσίας ή της επιχείρησης, βασισμένη στις τυποποιημένες ταξονομίες. Αυτές περιλαμβάνουν την Standard Industrial Classification (SIC), το North American Industry Classification System (NAICS) ,ή το United Nations Standard Products and Services Code (UNSPSC). Επειδή μια ενιαία επιχείρηση μπορεί να παρέχει διάφορες υπηρεσίες, μπορούν να υπάρξουν διάφορες κίτρινες σελίδες (κάθε μια που περιγράφει μια υπηρεσία)και να συνδέονται με μια άσπρη σελίδα (που δίνει τις γενικές πληροφορίες για την επιχείρηση).
- Green Pages: Οι πράσινες σελίδες χρησιμοποιούνται για να περιγράψουν πώς γίνεται η πρόσβαση σε μια υπηρεσία Ιστού, με πληροφορίες για τις συνδέσεις υπηρεσιών. Μερικές από τις πληροφορίες συσχετίζονται με την υπηρεσία Ιστού - όπως η διεύθυνση της υπηρεσίας , οι παράμετροι, και οι αναφορές στις προδιαγραφές των διεπαφών . Άλλες πληροφορίες δεν συσχετίζονται άμεσα με την υπηρεσία Ιστού - αυτό περιλαμβάνει τις λεπτομέρειες ηλεκτρονικού ταχυδρομείου, το FTP, το CORBA και πληροφορίες τηλεφώνων για την υπηρεσία. Επειδή μια υπηρεσία Ιστού μπορεί να έχει τις πολλαπλάσιες συνδέσεις (όπως καθορίζεται στην περιγραφή του WSDL), μια υπηρεσία μπορεί να έχει τις πολλαπλές πράσινες σελίδες, όμως κάθε σύνδεση θα πρέπει να προσεγγιστεί διαφορετικά.

2.6 REST

Το REST (representational state transfer) είναι μια προσέγγιση για να παρθούν πληροφορίες περιεχομένου από έναν ιστότοπο με την ανάγνωση της οριζόμενης ιστοσελίδας που περιέχει ένα αρχείο XML (Extensible Markup Language) που περιγράφει και περιλαμβάνει το επιθυμητό περιεχόμενο. Για παράδειγμα, το REST θα μπορούσε να χρησιμοποιηθεί, από έναν εκδότη που βρίσκεται σε απευθείας σύνδεση, για να καταστήσει το περιεχόμενο διαθέσιμο. Περιοδικά, ο εκδότης θα προετοίμαζε και θα ενεργοποιούσε ιστοσελίδες που θα περιελάμβαναν περιεχόμενο και δηλώσεις XML που περιέγραψαν το περιεχόμενο. Οι συνδρομητές θα πρέπει μόνο να ξέρουν το URL (Uniform Resource Locator) της σελίδα όπου το αρχείο XML βρέθηκε, διαβάστηκε από έναν Web browser, ερμηνεύτηκαν τα στοιχεία του περιεχομένου χρησιμοποιώντας τις πληροφορίες XML, και επαναμορφοποιήθηκε και χρησιμοποιήθηκε κατάλληλα (ίσως με κάποια μορφή σε απευθείας σύνδεση δημοσίευσης).

Όπως περιγράφεται σε μια διατριβή από το Roy Fielding, το REST είναι ένα «αρχιτεκτονικό ύφος» που βασικά εκμεταλλεύεται την υπάρχοντα τεχνολογία και τα πρωτόκολλα του Ιστού, συμπεριλαμβανομένου του HTTP (πρωτόκολλο μεταφοράς υπερκειμένων) και της XML. Το REST είναι απλούστερο για να χρησιμοποιηθεί από τη γνωστή προσέγγιση SOAP (Simple Object Access Protocol), η οποία απαιτεί ένα παρεχόμενο πρόγραμμα κεντρικών υπολογιστών (για να εξυπηρετήσει τα δεδομένα) και ένα πρόγραμμα πελατών (για να ζητήσει τα στοιχεία). Το SOAPs, εντούτοις, προσφέρει ενδεχομένως περισσότερη ικανότητα.

Το REST είναι σύμφωνο με μια προσέγγιση έκδοσης πληροφοριών που ένας αριθμός από Web log sites χρησιμοποιεί για να περιγράψει μερικές πτυχές του περιεχομένου τους , γνωστό ως RDF Site Summary (RSS). Το RSS χρησιμοποιεί το Resource Description Framework (RDF), έναν τυποποιημένο τρόπο για να περιγραφεί ένας ιστότοπος ή άλλος πόρος Διαδικτύου.

Συνεχίζοντας θα αναφέρουμε λίγα πράγματα για την ιστορία του. Το αρχιτεκτονικό ύφος REST αναπτύχθηκε παράλληλα με HTTP/1.1, βασισμένο στο υπάρχον σχέδιο HTTP/1.0. Η μεγαλύτερη εφαρμογή ενός συστήματος που προσαρμόζεται στο αρχιτεκτονικό ύφος REST είναι το World Wide Web. Το REST εξηγεί πώς η αρχιτεκτονική του Ιστού προέκυψε με το χαρακτηρισμό και

τον περιορισμό των μακρο-αλληλεπιδράσεων των τεσσάρων συστατικών του Ιστού, δηλαδή servers, gateways, proxies και clients, χωρίς την επιβολή περιορισμών στους μεμονωμένους συμμετέχοντες.

Η λογική του REST βασίζεται επίσης σε clients και servers. Οι πελάτες αρχίζουν τα αιτήματα στους κεντρικούς υπολογιστές , οι κεντρικοί υπολογιστές επεξεργάζονται τα αιτήματα και επιστρέφουν τις κατάλληλες απαντήσεις. Τα αιτήματα και οι απαντήσεις χτίζονται γύρω από τη μεταφορά των αντιπροσωπεύσεων των πόρων. Ένας πόρος μπορεί να είναι ουσιαστικά οποιαδήποτε συνεπής και σημαντική έννοια που μπορεί να διευθυνσιοδοτηθεί. Μια αντιπροσώπευση ενός πόρου είναι συνήθως ένα έγγραφο που συλλαμβάνει την τρέχουσα ή προοριζόμενη κατάσταση ενός πόρου.

Οποιοδήποτε στιγμή, ένας πελάτης μπορεί είτε να είναι στη μετάβαση μεταξύ της κατάστασης εφαρμογών είτε «at rest ». Ένας πελάτης σε κατάσταση rest είναι σε θέση να αλληλεπιδράσει με το χρήστη , αλλά δεν δημιουργεί, φορτώνει η καταναλώνει καμία αποθήκευση ανά-πελάτη στους κεντρικούς υπολογιστές ή στο δίκτυο.

Ο πελάτης αρχίζει τα αιτήματα όταν είναι έτοιμος να κάνει τη μετάβαση σε μια νέα κατάσταση. Ενώ ένα ή περισσότερα αιτήματα είναι σημαντικά, ο πελάτης θεωρείται σε κατάσταση μετάβασης. Η αντιπροσώπευση κάθε κατάσταση της εφαρμογής περιέχει τις συνδέσεις που μπορούν να χρησιμοποιηθούν την επόμενη φορά που θα επιλέξει ο πελάτης για να αρχικοποιήσει μια νέα κατάσταση μεταφοράς.

Το REST περιγράφηκε αρχικά στα πλαίσια του HTTP, αλλά δεν περιορίζεται σε εκείνο το πρωτόκολλο. Οι RESTful αρχιτεκτονικές μπορούν να βασιστούν σε άλλα πρωτόκολλα εφαρμογών διαστρωμάτωσης εάν παρέχουν ήδη ένα πλούσιο και ομοιόμορφο λεξιλόγιο για τις εφαρμογές βασισμένες στη μεταφορά της κατάστασης αναπαράστασης. Οι RESTful εφαρμογές μεγιστοποιούν τη χρήση της προϋπάρχουσας, καθορισμένης και με σαφήνεια διεπαφής και άλλων ενσωματωμένων ικανοτήτων που παρέχονται από το επιλεγμένο πρωτόκολλο δικτύων, τέλος ελαχιστοποιούν την προσθήκη των νέων οριζόμενων από εφαρμογή χαρακτηριστικών γνωρισμάτων στην κορυφή του.

Η λογική σχεδίου πίσω από την αρχιτεκτονική Ιστού μπορεί να περιγραφεί από ένα αρχιτεκτονικό ύφος που αποτελείται από το σύνολο περιορισμών που εφαρμόζονται στα στοιχεία μέσα στην αρχιτεκτονική. Με την εξέταση του αντίκτυπου κάθε περιορισμού όπως προστίθεται στο εξελισσόμενο ύφος, μπορούμε να προσδιορίσουμε τις ιδιότητες που προκαλούνται από τους περιορισμούς του Ιστού. Οι πρόσθετοι περιορισμοί μπορούν έπειτα να εφαρμοστούν για να διαμορφώσουν ένα νέο αρχιτεκτονικό ύφος που απεικονίζει καλύτερα τις επιθυμητές ιδιότητες μιας σύγχρονης αρχιτεκτονικής Ιστού. Αυτό το τμήμα παρέχει μια γενική επισκόπηση του REST μέσω της διαδικασίας προσεγγισής του ως αρχιτεκτονικό ύφος και παρακάτω θα περιγράφουν λεπτομερέστερα οι συγκεκριμένοι περιορισμοί που συνθέτουν το ύφος REST.

Έναρξη με το null ύφος: Υπάρχουν δύο κοινές προοπτικές στη διαδικασία του αρχιτεκτονικού σχεδίου, εάν είναι για τα κτήρια ή για το λογισμικό. Ο πρώτος είναι ότι ένας σχεδιαστής αρχίζει με τίποτα-μια κενή πλάκα, whiteboard, ή μια επιτροπή σχεδιασμού-και κατασκευή-επάνω σε μια αρχιτεκτονική από τα εξοικειωμένα συστατικά έως ότου ικανοποιεί τις ανάγκες του προοριζόμενου συστήματος. Ο δεύτερος είναι ότι ένας σχεδιαστής αρχίζει να χειρίζεται το σύστημα συνολικά, χωρίς περιορισμούς, και έπειτα αυξητικά προσδιορίζει και εφαρμόζει τους περιορισμούς στα στοιχεία του συστήματος προκειμένου να διαφοροποιηθεί το

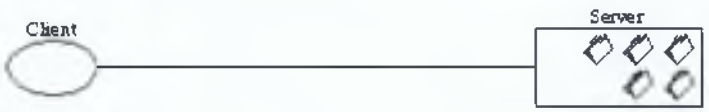
διάστημα σχεδίου και να επιτραπούν οι δυνάμεις που επηρεάζουν τη συμπεριφορά του συστήματος για να ρεύσουν φυσικά, σε αρμονία με το σύστημα. Όπου ο πρώτος υπογραμμίζει τη δημιουργικότητα και το απεριόριστο όραμα, ο δεύτερος υπογραμμίζει τον περιορισμό και την κατανόηση του πλαισίου συστημάτων. Το REST έχει αναπτυχθεί χρησιμοποιώντας την τελευταία διαδικασία.

Τα σχήματα 2.6.1 μέχρι 2.6.8 απεικονίζουν γραφικά αυτό από την άποψη για το πώς οι εφαρμοσμένοι περιορισμοί θα διαφοροποιούσαν την διαδικασία μιας αρχιτεκτονικής καθώς το αυξητικό σύνολο περιορισμών εφαρμόζεται.

Το null ύφος είναι απλά ένα κενό σύνολο περιορισμών. Από μια αρχιτεκτονική προοπτική, το null ύφος περιγράφει ένα σύστημα στο οποίο δεν υπάρχει κανένα διακεκριμένο όριο μεταξύ των συστατικών. Είναι η αφετηρία για την περιγραφή του REST.

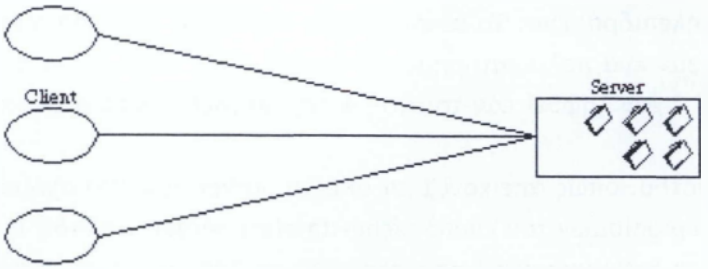
Σχήμα 2.6. 1 Το null ύφος

Client - Server:Οι πρώτοι περιορισμοί που προστίθενται στο υβριδικό πρότυπο μας είναι εκείνοι του αρχιτεκτονικού ύφους κεντρικών Client - Server (σχήμα 2.6.2). Ο χωρισμός των ανησυχιών είναι η αρχή πίσω από τους περιορισμούς Client - Server. Με το χωρισμό των ενδιάμεσων με τον χρήστη ανησυχιών από τις ανησυχίες αποθήκευσης στοιχείων, βελτιώνουμε τη φορητότητα του ενδιάμεσου με τον χρήστη στις πολλαπλές πλατφόρμες και βελτιώνουμε την εξελισιμότητα με την απλούστευση των τμημάτων κεντρικών υπολογιστών. Ίσως ο σημαντικότερος για τον Ιστό, είναι ότι ο χωρισμός επιτρέπει στα συστατικά να εξελιχθούν ανεξάρτητα, υποστηρικτικός κατά συνέπεια στην απαίτηση Διαδίκτυο-κλίμακας των πολλαπλών οργανωτικών τομέων



Σχήμα 2.6. 2 client-server

Stateless:Προσθέτουμε έπειτα έναν περιορισμό στην αλληλεπίδραση client-server: η επικοινωνία πρέπει να είναι stateless φύσης, όπως στον client-stateless-server (CSS) (σχήμα 2.6.3), έτσι ώστε κάθε αίτημα από τον πελάτη στον κεντρικό υπολογιστή να περιέχει όλες τις πληροφορίες που είναι απαραίτητες για να καταλάβουν το αίτημα, και δεν μπορεί να εκμεταλλευτεί οποιοδήποτε αποθηκευμένο πλαίσιο στον κεντρικό υπολογιστή. Το session state επομένως κρατιέται εξ ολοκλήρου στον πελάτη.

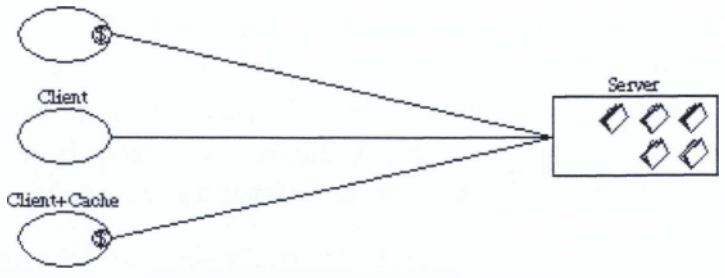


Σχήμα 2.6. 3 client-stateless-server

Αυτός ο περιορισμός βελτιώνει τις ιδιότητες της διαφάνειας, της αξιοπιστίας, και της εξελιξιμότητας. Η διαφάνεια βελτιώνεται επειδή ένα σύστημα παρακολούθησης δεν είναι απαραίτητο να κοιτάξει πέρα από ένα ενιαίο στοιχείο αιτήματος προκειμένου να καθοριστεί η πλήρης φύση του αιτήματος. Η αξιοπιστία βελτιώνεται επειδή διευκολύνει το στόχο από τις μερικές αποτυχίες. Η εξελιξιμότητα βελτιώνεται όχι επειδή πρέπει να αποθηκευτεί το state μεταξύ των αιτημάτων αλλά διότι επιτρέπει στο τμήμα κεντρικών υπολογιστών να παραμένει γρήγορο και ελεύθερο όσο αφορά τους πόρους, και απλοποιεί περαιτέρω την εφαρμογή επειδή ο κεντρικός υπολογιστής δεν είναι απαραίτητο να διαχειριστεί τη χρήση των πόρων στα αιτήματα.

Όπως οι περισσότερες αρχιτεκτονικές επιλογές, ο stateless περιορισμός απεικονίζει μια ανταλλαγή σχεδίου. Το μειονέκτημα είναι ότι μπορεί να μειώσει την απόδοση δικτύων με την αύξηση των επαναλαμβανόμενων στοιχείων (per-interaction overhead) που στέλνονται σε μία σειρά αιτημάτων, δεδομένου ότι εκείνο το στοιχείο δεν μπορεί να αφεθεί στον κεντρικό υπολογιστή σε ένα κοινό πλαίσιο. Επιπλέον, η τοποθέτηση της κατάστασης της εφαρμογής στην πλευρά του πελάτη μειώνει τον έλεγχο του κεντρικού υπολογιστή και συνεπώς της συμπεριφοράς της εφαρμογής, δεδομένου ότι η εφαρμογή γίνεται εξαρτώμενη από τη σωστή εφαρμογή της σημασιολογίας στις πολλαπλές εκδόσεις πελατών.

Cache: Προκειμένου να βελτιωθεί η αποδοτικότητα δικτύων, προσθέτουμε τους περιορισμούς cache για να διαμορφώσουμε το ύφος client-cache-stateless-server (σχήμα 2.6.4). Οι περιορισμοί cache απαιτούν ότι τα στοιχεία μέσα σε μια απάντηση σε ένα αίτημα θα χαρακτηρίζονται ως cacheable ή μη-cacheable. Εάν μια απάντηση είναι cacheable, κατόπιν σε μια cach-client δίνεται το δικαίωμα να επαναχρησιμοποιήσει όλα τα στοιχεία απάντησης για τα πιό πρόσφατα, ισοδύναμα αιτήματα.

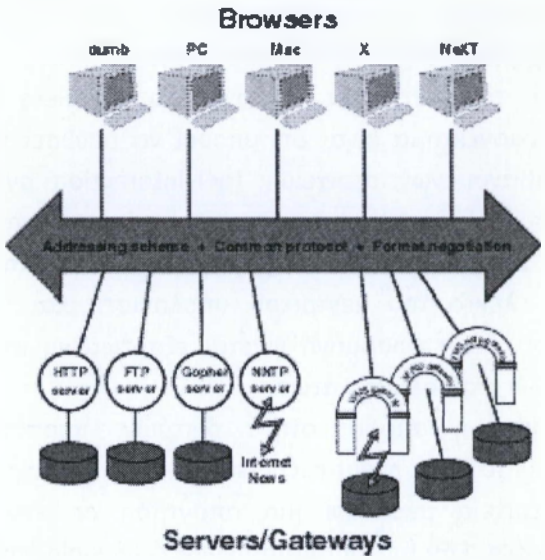


Σχήμα 2.6. 4 client-cache-stateless-server

Το πλεονέκτημα των περιορισμών της cache είναι ότι έχουν τη δυνατότητα μερικώς ή πλήρως να αποβάλλουν μερικές αλληλεπιδράσεις, βελτιώνοντας την αποδοτικότητα, την εξελιξιμότητα, και την αντιληπτή απόδοση του χρήστη με τη μείωση της μέσης λανθάνουσας

κατάστασης μέσω μιας σειράς αλληλεπιδράσεων. Το μειονέκτημα, εντούτοις, είναι ότι μια cache μπορεί να μειώσει την αξιοπιστία εάν ένα πολυδιατηρημένο στοιχείο μέσα στην cache διαφέρει σημαντικά από τα στοιχεία που θα είχαν ληφθεί εάν το αίτημα είχε σταλεί άμεσα στον κεντρικό υπολογιστή.

Η αρχιτεκτονική Ιστού, όπως απεικονίζεται από το διάγραμμα στο σχήμα 2.6.5, καθορίστηκε από το σύνολο των περιορισμών του client-cache-stateless-server. Δηλαδή η λογική σχεδίου που παρουσιάστηκε για την αρχιτεκτονική Ιστού πριν από το 1994 εστίασε στην client-stateless-server για την ανταλλαγή των στατικών εγγράφων μέσω του Διαδικτύου. Τα πρωτόκολλα για την επικοινωνία των αλληλεπιδράσεων είχαν τη στοιχειώδη υποστήριξη για τις μη-κοινές cache, αλλά δεν περιόρισαν τη διεπαφή σε ένα συνεπές σύνολο σημασιολογίας για όλους τους πόρους. Αντ' αυτού, ο Ιστός στηρίχθηκε στη χρήση μιας κοινής βιβλιοθήκης εφαρμογής client-servers (Κέντρο Πυρηνικών Μελετών και Ερευνών (CERN) libwww) για να διατηρήσει τη συνέπεια στις εφαρμογές Ιστού.

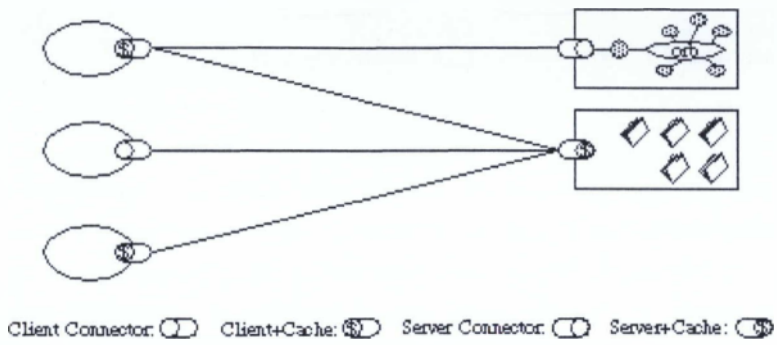


Σχήμα 2.6. 5

Οι υπεύθυνοι για την ανάπτυξη των εφαρμογών Ιστού είχαν υπερβεί ήδη το σχέδιο. Εκτός από τα στατικά έγγραφα, τα αιτήματα θα μπορούσαν να προσδιορίσουν τις υπηρεσίες που παρήγαγαν δυναμικά τις απαντήσεις, όπως οι εικόνα-χάρτες [Kevin Hughes] και τα server side scripts [Rob McCool]. Εργασία είχε αρχίσει επίσης και στα ενδιάμεσα συστατικά, υπό μορφή πληρεξούσιων και κοινών κρυπτών cache, αλλά και επεκτάσεις στα πρωτόκολλα απαιτήθηκαν για να επικοινωνούν αξιόπιστα .Τα επόμενα τμήματα περιγράφουν τους περιορισμούς που προστίθενται στο αρχιτεκτονικό ύφος του Ιστού προκειμένου να καθοδηγηθούν οι επεκτάσεις που διαμορφώνουν τη σύγχρονη αρχιτεκτονική Ιστού.

Uniform Interface: Το κεντρικό χαρακτηριστικό γνώρισμα που διακρίνει το αρχιτεκτονικό ύφος REST, από άλλες βασισμένες στο δίκτυο μορφές, είναι η έμφαση που δίνει σε μια ομοιόμορφη διεπαφή μεταξύ των συστατικών (σχήμα 2.6.6). Με την εφαρμογή της αρχής της γενικότητας στη διεπαφή, η αρχιτεκτονική συστημάτων απλοποιείται και η διαφάνεια των αλληλεπιδράσεων βελτιώνεται. Οι εφαρμογές αποσυνδέονται από τις υπηρεσίες που παρέχουν, το οποίο ενθαρρύνει το ανεξάρτητο evolvability. Το μειονέκτημα είναι ότι μια ομοιόμορφη διεπαφή

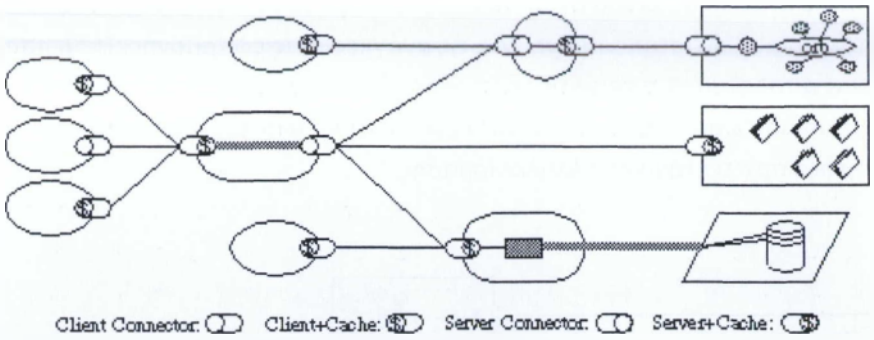
υποβιβάζει την αποδοτικότητα, δεδομένου ότι οι πληροφορίες μεταφέρονται σε μια τυποποιημένη μορφή παρά σε μια που είναι βελτιστοποιημένη για τις ανάγκες μιας εφαρμογής. Η διεπαφή REST σχεδιάζεται για να είναι αποδοτική για τη μεταφορά δεδομένων υπερμεσών large-grain, που είναι βέλτιστη για την κοινή περίπτωση του Ιστού, αλλά έχει ως συνέπεια μια διεπαφή που δεν είναι βέλτιστη για άλλες μορφές αρχιτεκτονικής αλληλεπίδρασης.



Σχήμα 2.6. 6 uniform-client-cache-stateless-server.

Προκειμένου να διατηρηθεί μια ομοιόμορφη διεπαφή, απαιτούνται πολλοί αρχιτεκτονικοί περιορισμοί για να καθοδηγήσουν τη συμπεριφορά των συστατικών. Το REST καθορίζεται από τέσσερις περιορισμούς διεπαφών: προσδιορισμός των πόρων(identification of resources), χειρισμός των πόρων μέσω των αντιπροσωπεύσεων (manipulation of resources through representations), μόνος-περιγραφικά μηνύματα(self-descriptive messages) και υπερμέσα ως μηχανή της κατάστασης εφαρμογής(hypermedia as the engine of application state).

Layered System (διαστρωματωμένα συστήματα): Προκειμένου να βελτιωθεί περαιτέρω η συμπεριφορά για τις απαιτήσεις της κλίμακας Διαδικτύου, προσθέτουμε τους περιορισμούς των διαστρωματωμένων συστημάτων (σχήμα 2.6.7). Το ύφος των διαστρωματωμένων συστημάτων που επιτρέπουν σε μια αρχιτεκτονική να αποτελείται από ιεραρχικά στρώματα με τον περιορισμό η συμπεριφορά των συστατικών να είναι τέτοια έτσι ώστε κάθε συστατικό να μην μπορεί «να δει» πέρα από το άμεσο στρώμα με το οποίο αλληλεπιδρά. Περιορίζοντας τη γνώση του συστήματος σε ένα ενιαίο στρώμα, τοποθετούμε ένα όριο στη γενική πολυπλοκότητα του συστήματος και προάγουμε ανεξαρτησία στο υπόστρωμα. Τα στρώματα μπορούν να χρησιμοποιηθούν για να συμπεριλάβουν τις υπηρεσίες που κληρονομούν και για να προστατεύσουν τις νέες υπηρεσίες από τους πελάτες που κληρονομούν, απλοποιώντας συστατικά μετακινώντας τη λειτουργία προς έναν κοινό μεσάζοντα. Οι μεσάζοντες μπορούν επίσης να χρησιμοποιηθούν για να βελτιώσουν την εξελιξιμότητα των συστημάτων διευκολύνοντας την εξισορροπημένη φόρτωση των υπηρεσιών σε πολλαπλά δίκτυα και επεξεργαστές.

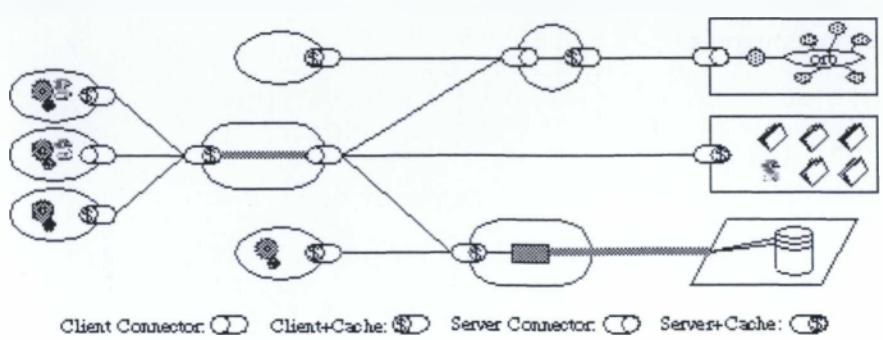


Σχήμα 2.6. 7 uniform-layered-client-cache-stateless-server.

Το κύριο μειονέκτημα των διαστρωματωμένων συστημάτων είναι ότι προσθέτουν τη λανθάνουσα κατάσταση στην επεξεργασία των στοιχείων, έτσι μειώνουν την αντιληπτή απόδοση του χρήστη. Για ένα βασισμένο στο δίκτυο σύστημα που υποστηρίζει τους περιορισμούς cache, μπορεί να αντισταθμιστεί από τα οφέλη της κοινής cache στους μεσάζοντες. Η τοποθέτηση των κοινών cache στα όρια μιας οργανωτικής περιοχής μπορεί να οδηγήσει σε σημαντικά οφέλη απόδοσης. Τέτοια στρώματα επιτρέπουν επίσης στις πολιτικές ασφαλείας για να επιβληθούν στα στοιχεία που διασχίζουν το οργανωτικό όριο, όπως απαιτείται από τα firewalls.

Ο συνδυασμός διαστωματομένου συστήματος και των ομοιόμορφων περιορισμών διεπαφών προκαλεί αρχιτεκτονικές ιδιότητες παρόμοιες με εκείνους του ομοιόμορφου ύφους pipe-and-filter. Αν και η αλληλεπίδραση REST είναι διπλής κατεύθυνσης, οι ροές στοιχείων large-grain της αλληλεπίδρασης υπερμεσών μπορούν να υποβληθούν σε επεξεργασία όπως ένα δίκτυο ροής πληροφοριών, με τμήματα φίλτρων να εφαρμόζονται επιλεκτικά στο ρεύμα στοιχείων προκειμένου να μετασχηματιστεί το περιεχόμενο καθώς περνά. Μέσα στο REST, τα ενδιάμεσα συστατικά μπορούν ενεργά να μετασχηματίσουν το περιεχόμενο των μηνυμάτων επειδή τα μηνύματα είναι self-descriptive και η σημασιολογία τους είναι ορατή στους μεσάζοντες.

Code-On-Demand(κώδικας σε ζήτηση): Η τελική προσθήκη στο σετ των περιορισμών μας που τίθεται για το REST προέρχεται από το ύφος code-on-demand (σχήμα 2.6.8). Το REST επιτρέπει στη λειτουργία πελατών να επεκταθεί κατεβάζοντας και εκτελώντας τον κώδικα υπό μορφή applets ή scripts. Αυτό απλοποιεί τους πελάτες με τη μείωση του αριθμού χαρακτηριστικών γνωρισμάτων που απαιτούνται για να προ-εφαρμοστούν. Η άδεια των χαρακτηριστικών γνωρισμάτων για να μεταφορτωθεί μετά από την επέκταση βελτιώνει την επέκταση των συστημάτων. Εντούτοις, μειώνει τη διαφάνεια, και είναι μόνο ένας προαιρετικός περιορισμός μέσα στο REST.

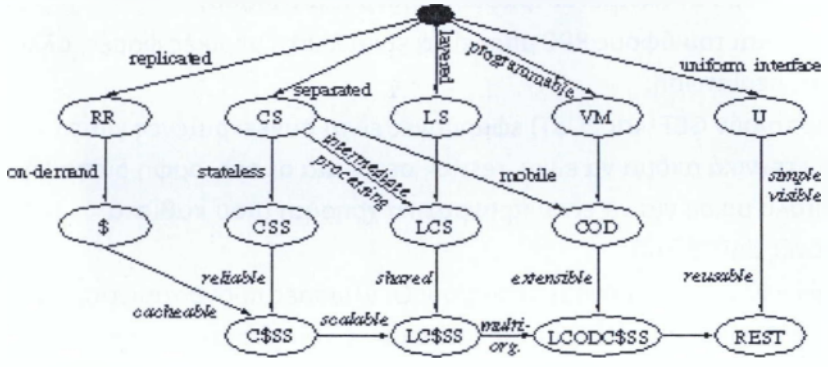


Σχήμα 2.6. 8 REST

Η έννοια ενός προαιρετικού περιορισμού μπορεί να φανεί οξύμωρος. Εντούτοις, έχει σαν σκοπό σε ένα αρχιτεκτονικό σχέδιο ενός συστήματος που καλύπτει τα πολλαπλάσια οργανωτικά όρια. Σημαίνει ότι η αρχιτεκτονική αποκομίζει μόνο το όφελος (και υφίσταται τα μειονεκτήματα) των προαιρετικών περιορισμών όταν είναι γνωστοί για να είναι σε ισχύ για κάποια σφαίρα του γενικού συστήματος. Για παράδειγμα, εάν όλο το λογισμικό πελατών μέσα σε μια οργάνωση είναι γνωστό ότι υποστηρίζει java applets , τότε οι υπηρεσίες μέσα σε εκείνη την οργάνωση μπορούν να κατασκευαστούν έτσι ώστε να αποκομίζουν το όφελος της ενισχυμένης λειτουργίας μέσω των τηλεκατεγραφόμενων κλάσεων της java. Συγχρόνως, το firewall της οργάνωσης μπορεί να αποτρέψει τη μεταφορά των java applets από τις εξωτερικές πηγές, και έτσι στο υπόλοιπο του Ιστού θα εμφανιστεί ότι εκείνοι οι πελάτες δεν υποστηρίζουν τον code-on-demand. Ένας προαιρετικός περιορισμός επιτρέπει σε μας να σχεδιάσουμε μια αρχιτεκτονική που υποστηρίζει την επιθυμητή συμπεριφορά στη γενική περίπτωση, αλλά έχοντας κατανοήσει ότι μπορεί να τεθεί εκτός λειτουργίας μέσα σε μερικά πλαίσια.

Περίληψη παραγωγής ύφους: Το REST αποτελείται από ένα σύνολο αρχιτεκτονικών περιορισμών που επιλέγονται για τις ιδιότητες που προκαλούν στις αρχιτεκτονικές των υποψηφίων. Αν και κάθε ένας από αυτούς τους περιορισμούς μπορεί να εξεταστεί μόνος του, η περιγραφή τους από την άποψη της παραγωγής τους από τις κοινές αρχιτεκτονικές μορφές το καθιστά ευκολότερο να καταλάβει τη λογική πίσω από την επιλογή τους. Το σχήμα 6.2.9 απεικονίζει την παραγωγή των περιορισμών του REST γραφικά από την άποψη των βασισμένων στο δίκτυο αρχιτεκτονικών.

Σχήμα 2.6. 9 Παραγωγή του REST με βάση τους περιορισμούς.



2.6.1 Τα πλεονεκτήματα και τα μειονεκτήματα του REST

Πλεονεκτήματα REST:

- Εξελιξιμότητα στα συστατικά που αλληλεπιδρούν.
- Γενικότητα των διεπαφών
- Ανεξάρτητη επέκταση του κάθε συστατικού.
- Τα ενδιάμεσα συστατικά για να μειώσουν τη λανθάνουσα κατάσταση, επιβάλλουν την ασφάλεια και τοποθετούν τα συστήματα κληρονομιών σε cache.
- Χωρίζει την εφαρμογή των server από την αντίληψη του client για τους πόρους (« Τα URIs δεν αλλάζουν»)
- Οι κλίμακες είναι ιδανικές για τους μεγάλους αριθμούς πελατών

- Επιτρέπει τη μεταφορά των στοιχείων μέσω ροής απεριόριστου μεγέθους και τύπου

Το REST έχει εφαρμοστεί για να περιγράψει την επιθυμητή αρχιτεκτονική Ιστού, για να βοηθήσει να προσδιοριστούν τα υπάρχοντα προβλήματα, να συγκρίνει τις εναλλακτικές λύσεις, και να εξασφαλίσει ότι οι επεκτάσεις πρωτοκόλλου δεν θα παραβίαζαν τους περιορισμούς πυρήνων που καθιστούν τον Ιστό επιτυχή.

Η τοποθέτηση περιγράφει την επίδραση του REST στην εξελιξιμότητα έτσι:

Ο χωρισμός client-server του REST των ανησυχιών απλοποιεί τη εφαρμογή, μειώνει την πολυπλοκότητα της σημασιολογίας συνδετήρων, βελτιώνει την αποτελεσματικότητα του συντονισμού απόδοσης, και αυξάνει την εξελιξιμότητα των καθαρών τμημάτων των server. Οι περιορισμοί των διαστρωματοποιημένων συστημάτων επιτρέπουν τα μεσάζων- proxies, gateways, και firewalls- να εισαχθούν σε διάφορα σημεία στην επικοινωνία χωρίς την αλλαγή των διεπαφών μεταξύ των συστατικών, επιτρέποντας κατά συνέπεια να βοηθήσει την επικοινωνία ή να βελτιώσει την απόδοση μέσω της μεγάλης κλίμακας, και της διαμοιραζόμενης cache. Το REST επιτρέπει την ενδιάμεση επεξεργασία θέτοντας ως περιορισμό τα μηνύματα να είναι self-descriptive: η αλληλεπίδραση είναι stateless μεταξύ των αιτημάτων, οι τυποποιημένοι μέθοδοι και οι τύποι μέσων χρησιμοποιούνται για να δείξουν τις πληροφορίες σημασιολογίας και ανταλλαγής, και οι απαντήσεις δείχνουν ρητά το cacheability.

Μειονεκτήματα REST

- Μόνο λίγα εργαλεία που αφορούν το ύφος REST
- Οι περιορισμοί για το μήκος του GET μπορούν μερικές φορές να είναι ένα πρόβλημα.
- Καμία άμεση γέφυρα στον κόσμο αντικειμενοστραφούς προγραμματισμού.
- Η διαφορά μεταξύ του REST και του ύφους RPC μπορεί να είναι λεπτή μερικές φορές, αλλά όχι απαραιτήτως σε γενική περίπτωση.
- «Lo-rest» (οποίες χρησιμοποιούν GET και POST) εφαρμογές είναι συγκεκριμένες για το REST μέσω του HTTP: Ενώ τεχνικά ακόμα να είναι restful, σε με μια ομοιόμορφη διεπαφή με 2 ρήματα είναι πάρα πολύ μικρή για να είναι πραγματικά χρήσιμη (που καθιστά πράγματι πολλές εφαρμογές unRESTful).
- Περιορισμοί των σημερινών γλωσσών προγραμματισμού: Οι γλώσσες προγραμματισμού δεν είναι προσανατολισμένες προς τους πόρους έτσι οι κώδικες που διαχειρίζονται τους χάρτες URIs τείνουν να είναι ακατάστατοι. Αφ' ενός είναι σχετικά δύσκολο να κατασταθεί το υπερκείμενο του REST API οδηγούμενο (που είναι ένας περιορισμός του REST).

Γιατί όμως πρέπει να χρησιμοποιηθεί το REST και όχι το SOAP εφόσον είναι προορισμένα να χρησιμοποιούνται για τον ίδιο λόγο; Παρακάτω ακολουθεί μια σύγκριση μεταξύ των δύο υφών αρχιτεκτονικής για να γίνουν κατανοητές οι διαφορές τους.

Το REST περιγράφεται ως αρχιτεκτονικό ύφος για να χτίσει εφαρμογές οι οποίες αξιοποιούν περισσότερο της τρέχουσες τεχνολογίες Ιστού για τη μεταφορά των στοιχείων μεταξύ των συστημάτων ηλεκτρονικών υπολογιστών. Από μία άποψη, το REST παίρνει τις ίδιες τεχνολογικές έννοιες πίσω από έναν browser που ζητά στοιχεία από μια εφαρμογή που τρέχει σε έναν κεντρικό υπολογιστή, και τις εφαρμόζει σε επικοινωνίες τύπου εφαρμογής-εφαρμογής. Η αρχιτεκτονική χρησιμοποιεί XML, URLs, και «get», «put» εντολές που χρησιμοποιούνται συνήθως μέσω του Διαδικτύου.

Από την άλλη πλευρά, το SOAP χτίζει ένα remote procedure call (RPC) μέσω του HTTP και της XML. Συνήθως χρησιμοποιούνται σε LANs, τα RPCs είναι μια προγραμματιστική διεπαφή που

επιτρέπει σε μια εφαρμογή να χρησιμοποιήσει υπηρεσίες από ένα μακρινό σύστημα. Το SOAP χρησιμοποιεί σύνταξη XML για να στείλει εντολές κειμένου μεταξύ των εφαρμογών σε ολόκληρο το Διαδίκτυο και προωθείται ως ελαφρύς και λιγότερο εντατικός τρόπος προγραμματισμού από τα RPC ή τις παρόμοιες τεχνολογίες όπως η DCOM της Microsoft.

Αλλά σαν τεχνολογία ύφους RPC, το SOAP παίρνει πολλά από τα προβλήματα του αντικειμενοστραφούς υπολογισμού στον Ιστό. Επομένως, έχει περισσότερο νόημα για την αξιοποίηση όσο το δυνατόν περισσότερη τρέχουσα τεχνολογία Ιστού, υποστηρίζουν μερικοί εμπειρογνώμονες. "REST says, 'Let's make Web services an extension of what's great about the Web already,' in terms of a small set of verbs and a document-centric philosophy,"(Το REST υποστηρίζει, κάντε τις υπηρεσίες Ιστού μια επέκταση αυτού που είναι σπουδαίο για τον Ιστό ήδη, από την άποψη ενός μικρού συνόλου ρημάτων και μιας έγγραφο-κεντρικής φιλοσοφίας) λέει ο Scott Means, CEO of Enterprise Web Machines Inc. στην Columbia, S.C, και συντάκτης διάφορων βιβλίων για την XML.

Για παράδειγμα, η χρήση του SOAP για να πάρει τις πληροφορίες για έναν υπάλληλο μπορεί να περιλάβει στοιχεία σχεδίων από διάφορες πηγές και να τα φέρει όλα μαζί για την επίδειξη ως ένα έγγραφο. Η REST αρχιτεκτονική, από την άλλη, θα ζητούσε ένα έγγραφο XML που περιέχει όλες τις πληροφορίες και θα το παρέδιδε σε μια άλλη εφαρμογή, ή τη χρήση ενός XSLT (eXtensible Stylesheet Language Transformation) για να το μετατρέψει σε HTML για την επίδειξη του σε έναν browser.

Ο Means υποστηρίζει ότι πολλές πληροφορίες αποθηκεύονται ήδη σε XML μορφή για τον Ιστό και μπορούν να παρασχεθούν ως υπηρεσίες Ιστού χωρίς οικοδόμηση των νέων SOAP-βασισμένων εφαρμογών. «Ο στόχος στο τέλος θα ήταν μια ενοποιημένη υποδομή, έτσι ώστε να μην υπάρχει ποτέ αναπαραχθείσα προσπάθεια από την άποψη ενός συστήματος να παραδοθούν πράγματα για δημόσια χρήση στο διαδίκτυο και για την παράδοση σε συνεργατικά sites.» είπε ο Means.

Φυσικά, το REST δεν είναι πανάκεια. Το SOAP θα ήταν η καλύτερη μέθοδος για μια εφαρμογή κληρονομικότητας στον Ιστό, υπό τον όρο ότι ο υπεύθυνος για την ανάπτυξη μπορεί να επεκτείνει έναν επαρκή μηχανισμό ασφάλειας. Αλλά το REST θα ήταν μια εναλλακτική λύση στο σωρό υπηρεσιών Ιστού εάν οι επιχειρήσεις συμφωνούσαν εκ των προτέρων να υιοθετήσουν την αρχιτεκτονική στα συγκροτήματα ηλεκτρονικών υπολογιστών τους για τα έγγραφα, όπως οι εντολές αγοράς, μέσω του Διαδικτύου.

«Είναι αρκετά απλό,» είπε ο Ronald Schmelzer, αναλυτής για την έρευνα ZapThink LLC. «Μπορείτε να κάνετε μερικά γερά πράγματα με αυτό, και δεν έχει τα στρώματα της πολυπλοκότητας που τα πρωτόκολλα υπηρεσιών Ιστού έχουν.»

Το REST είναι απλούστερο επειδή έχει τη λιγότερη ασάφεια από το σωρό υπηρεσιών Ιστού, και λιγότερα σημεία αποτυχίας. Οι υποστηρικτές του SOAP έπρεπε να διαμορφώσουν μια χωριστή ομάδα, τη Web Services Interoperability Organization, για να αναπτύξουν τα πρότυπα που εξασφαλίζουν ότι οι SOAP-εφαρμογές μπορούν να επικοινωνήσουν.

Εντούτοις, το SOAP και οι συγγενικές εφαρμογές του μπορούν να εκτελέσουν και το ασύγχρονο και σύγχρονο μήνυμα, ενώ το REST είναι καταλληλότερο για τα σύγχρονα. Επιπλέον, επειδή πολλοί vendors υποστηρίζουν το σωρό υπηρεσιών Ιστού, οι περισσότερες εφαρμογές θα υποστηρίξουν το SOAP, διευκολύνοντας τους υπεύθυνους για την ανάπτυξη να συνδεθεί το λογισμικό μέσω του Διαδικτύου χρησιμοποιώντας το SOAP.

Οι υποστηρικτές του REST διαφωνούν η ότι η προτιμημένη αρχιτεκτονική τους επιτρέπει σε ένα τμήμα ΤΠ για να εξαρτηθεί λιγότερο από τους vendors με την παράκαμψη του σωρού τεχνολογίας υπηρεσιών Ιστού, ο οποίος περιλαμβάνει SOAP, WSDL, και UDDI.

«Οι συζητήσεις είναι πραγματικά διενέξεις για τις υπηρεσίες Ιστού για τους ανθρώπους εναντίον των υπηρεσιών Ιστού για τις επιχειρήσεις,» λέει ο Schmelzer . «Υπάρχουν πολλοί άνθρωποι πίσω από το REST και πολλά vendor πράγματα πίσω από τις υπηρεσίες Ιστού.»

Εντούτοις, οι μεγάλες επιχειρήσεις δεν θα αναμιχθούν στους πολέμους και είναι πιθανότερο να κολλήσουν με την επιλογή που έχει την ισχυρότερη υποστήριξη από τους vendors. Ο σωρός υπηρεσιών Ιστού υποστηρίζεται από όλους τους σημαντικότερους vendors.

«Η επιχείρηση πρόκειται να πάει με το ασφαλέστερο στοίχημα και το ασφαλέστερο στοίχημα, για να είναι τίμιος , είναι ο σωρός υπηρεσιών Ιστού.» ισχυρίζεται ο Schmelzer .

2.6.2 RESTful Web Services.

Μια RESTful υπηρεσία Ιστού (επίσης αποκαλούμενη RESTful Web API) είναι μια απλή υπηρεσία Ιστού που χρησιμοποιεί το HTTP και τις αρχές του REST. Είναι μια συλλογή πόρων, με τρεις καθορισμένες πτυχές:

- Η βάση URI για την υπηρεσία Ιστού, είναι: http://example.com/resources/
- Ο τύπος Internet media των στοιχείων υποστηρίζεται από την υπηρεσία Ιστού. Αυτό είναι συχνά JSON, XML ή YAML αλλά μπορεί να είναι οποιοσδήποτε άλλος έγκυρος τύπος Internet media.
- Το σύνολο διαδικασιών που υποστηρίζονται από την υπηρεσία Ιστού χρησιμοποιεί τις μεθόδους HTTP (POST, GET, PUT η DELETE).

Ο ακόλουθος πίνακας επιδεικνύει πώς οι μέθοδοι HTTP χρησιμοποιούνται χαρακτηριστικά για να εφαρμοστούν σε μια υπηρεσία Ιστού.

Πόροι	GET	PUT	POST	DELETE
Συλλογή από URI όπως `http://example.com/resources/`	List: Απαριθμήστε το URIs και ίσως άλλες λεπτομέρειες των μελών της συλλογής.	Replace: Αντικαταστήστε ολόκληρη την συλλογή με μια άλλη συλλογή.	Create: Δημιουργήστε μια νέα είσοδο στη συλλογή. Το URL της νέας εισόδου ορίζεται αυτόματα και συνήθως επιστρέφεται από τη λειτουργία.	Delete: Διαγράψτ την ολόκληρ συλλογή.
Στοιχεία URI όπως http://example.com/resources/ef7d-xi36p	Retrieve: Ανακτήστε μια αντιπροσώπευση	Update: Ενημερώστε το ορισμένο μέλος της συλλογής.	Το ορισμένο μέλος ως συλλογή και	Διαγράψ το ορισ μέλος τ

	του ορισμένου μέλους της συλλογής, που εκφράζεται σε έναν κατάλληλο τύπο μέσων Διαδικτύου.		δημιουργήστε μια νέα είσοδο σε το.	συλλογής.

2.6.2.1 Χαρακτηριστικά των RESTfull υπηρεσιών.

Όπως είναι προφανές τα χαρακτηριστικά των RESTful εφαρμογών είναι παρόμοια με αυτά του αρχιτεκτονικού ύφους REST. Θα μπορούσε να θεωρηθεί παράληψη η μη αναφορά τους οπότε θα παρουσιαστούν χωρίς να προχωρήσουμε σε ιδιαίτερο βάθος.

1. Client-Server: ένα ύφος βασισμένο στην pull αλληλεπίδραση: κατανάλωση ολων των τμημάτων και χρήση αναπαραστάσεων.
2. Stateless: κάθε αίτημα από τον πελάτη στον κεντρικό υπολογιστή πρέπει να περιέχει όλες τις πληροφορίες απαραίτητες για να γίνει κατανοητό το αίτημα, και ο πελάτης δεν μπορεί να εκμεταλλευτεί οποιοδήποτε αποθηκευμένο πλαίσιο στον κεντρικό υπολογιστή.
3. cache: για να βελτιώσουν την αποδοτικότητα των δικτύων οι απαντήσεις πρέπει να είναι ικανές να χαρακτηριστούν ως cacheable ή μη-cacheable.
4. Ομοιόμορφη διεπαφή: όλοι οι πόροι προσεγγίζονται με μια γενική διεπαφή (π.χ., το HTTP GET, POST, PUT, DELETE).
5. Ονομασμένοι πόροι - το σύστημα αποτελείται από τους πόρους που ονομάζονται χρησιμοποιώντας ένα URL.
6. Διασυνδεμένες αντιπροσωπεύσεις των πόρων - οι αντιπροσωπεύσεις των πόρων διασυνδέονται χρησιμοποιώντας URLs, με αυτόν τον τρόπο επιτρέπουν σε έναν πελάτη για να προχωρήσουν από τη μία κατάσταση σε μία άλλη.
7. Διαστρωματοποιημένα συστατικά - οι μεσάζοντες, όπως οι proxy servers, cache servers, gateways,, κ.λπ., μπορούν να παρεμβληθούν μεταξύ των πελατών και των πόρων για να υποστηρίξουν την απόδοση, την ασφάλεια, κ.λπ.

2.6.2.2 Αρχές του σχεδίου των RESTfull services

1. Το κλειδί για τη δημιουργία των υπηρεσιών Ιστού σε ένα δίκτυο REST (δηλ., ο Ιστός) είναι να προσδιοριστούν όλες οι εννοιολογικές οντότητες που επιθυμείτε να εκθέσετε ως υπηρεσίες. Παραδείγματα πόρων: κατάλογος μερών, λεπτομερή στοιχεία μερών, εντολή αγοράς.
2. Δημιουργήστε ένα URL σε κάθε πόρο. Οι πόροι πρέπει να είναι ουσιαστικά, όχι ρήματα. Παραδείγματος χάριν, μην χρησιμοποιήστε αυτό:

http://www.parts-depot.com/parts/getPart?id=00345

Σημειώστε το ρήμα, getPart. Αντ' αυτού, χρησιμοποιήστε ένα ουσιαστικό:

3. Ταξινομήστε τους πόρους σας σύμφωνα με το εάν οι πελάτες μπορούν ακριβώς να λάβουν μια αντιπροσώπευση του πόρου, ή εάν οι πελάτες μπορούν να τροποποιήσουν τον πόρο. Για το πρώτο, καταστήστε εκείνους τους πόρους προσιτούς χρησιμοποιώντας ένα HTTP GET. Για το επόμενο, κάνετε εκείνους τους πόρους προσιτούς χρησιμοποιώντας HTTP POST, PUT, η DELETE.
4. Όλοι οι πόροι προσιτοί μέσω του HTTP GET πρέπει να είναι side-effect free. Δηλαδή ο πόρος πρέπει ακριβώς να επιστρέψει μια αντιπροσώπευση του πόρου. Η επίκληση του πόρου δεν πρέπει να οδηγήσει στην τροποποίηση του πόρου.
5. Πρέπει να τοποθετηθούν σύνδεσμοι υπερ-κειμένου μέσα στις αντιπροσωπεύσεις των πόρων για να επιτρέψει στους πελάτες να κάνουν drill down για περισσότερες πληροφορίες, ή για να λάβουν τις σχετικές πληροφορίες.
6. Το σχέδιο να αποκαλύψει τα στοιχεία βαθμιαία. Δεν πρεπει να αποκαλύφτουν όλα σε ένα ενιαίο έγγραφο απάντησης. Οι σύνδεσμοι υπερ-κειμένου παρέχονται για να ληφθούν περισσότερες λεπτομέρειες.
7. Ορίζουμε τη μορφή των στοιχείων απάντησης χρησιμοποιώντας ένα σχήμα (DTD, W3C Schema, RelaxNG, ή Schematron). Για εκείνες τις υπηρεσίες που απαιτούν POST ή PUT, παρέχετε επίσης ένα σχήμα για να διευκρινίσετε το σχήμα της απάντησης.
8. Περιγράψτε πώς οι υπηρεσίες σας πρέπει να επικαλεσθούν χρησιμοποιώντας είτε ένα έγγραφο WSDL, είτε απλά ένα έγγραφο HTML.

2.7 WADL

Η Web Application Description Language (WADL) είναι μια μορφή αρχείου βασισμένη σε XML που παρέχει μια αναγνώσιμη περιγραφή των βασισμένων σε HTTP εφαρμογών Ιστού από μηχανή. Αυτές οι εφαρμογές είναι υπηρεσίες Ιστού βασισμένες στο REST .

Ο σκοπός του WADL είναι να επιτραπεί στις υπηρεσίες στο διαδίκτυο (ή οποιοδήποτε δίκτυο IP) να περιγραφούν με έναν τρόπο επεξεργάσιμο από τις μηχανές, για να καταστήσει ευκολότερη τη δημιουργία εφαρμογών ύφους web 2.0 και να δημιουργήσει με έναν δυναμικό τρόπο υπηρεσίες. Πριν από αυτό, ήταν απαραίτητο να μελετηθεί μια υπάρχουσα υπηρεσία Ιστού, και να γραφτεί μια καινούρια εφαρμογή με το χέρι. Το WADL μπορεί να θεωρηθεί ως το ισοδύναμο (που αφορα όμως μόνο το REST) του Web Services Description Language της έκδοσης 1.1.

Το WADL προορίζεται για τις εφαρμογές που είναι βασισμένες στην υπάρχουσα αρχιτεκτονική του Ιστού. Όπως το WSDL, είναι ανεξάρτητο από πλατφόρμες και γλώσσες και στοχεύει να προωθήσει την επαναχρησιμοποίηση των εφαρμογών πέρα από τη βασική χρήση σε ένα web browser. Το WADL διαμορφώνει τους πόρους που παρέχονται από μια υπηρεσία, και τις σχέσεις μεταξύ τους.

Η υπηρεσία περιγράφεται χρησιμοποιώντας ένα σύνολο στοιχείων από πόρους. Κάθε ένας από αυτούς περιλαμβάνει στοιχεία παραμέτρων για να περιγράψει τις εισαγωγές, και τα στοιχεία τις μεθόδου περιγράφουν το αίτημα και την απάντηση ενός πόρου. Το στοιχείο αιτήματος διευκρινίζει πώς θα αντιπροσωπεύσει την εισαγωγή, ποιοι τύποι απαιτούνται και ποιες συγκεκριμένες επιγραφές HTTP. Η απάντηση περιγράφει την αντιπροσώπευση της απάντησης της υπηρεσίας, καθώς επίσης και οποιεσδήποτε πληροφορίες ελαττωμάτων, για να εξετάσει τα λάθη.

Το WADL δεν υποστηρίζεται ακόμα ευρέως. Έχει πλεονέκτημα σε σχέση με το πιο περίπλοκο WSDL δεδομένου ότι δεν επιβάλλει περαιτέρω επίπεδο αφαίρεσης στην περιγραφή υπηρεσιών παρόλα αυτά, καθώς τα εργαλεία γίνονται διαθέσιμα για τη ανάπτυξη εφαρμογών με WADL, είναι πιθανό να περιλαμβάνουν τρόπους για να παράγεται αυτόματα , και αυτό διακινδυνεύει να προσθέσει ένα ύφος RPC , που πηγαίνει ενάντια στην προοριζόμενη απλότητα του WADL.

Διάφορες επιχειρήσεις που είναι βασισμένες στο διαδύκτυο (όπως Google, Yahoo ,Amazon, Flickr και άλλες) αναπτύσσουν εφαρμογές ή υπηρεσίες που βασίζονται σε XML ή HTTP για να παρέχουν πρόσβαση στα εσωτερικά στοιχεία τους. Το WADL έχει ως σκοπό να παρέχει μια απλή εναλλακτική λύση σε αντίθεση με το WSDL που προορίζεται για χρήση σε εφαρμογές Ιστού βασισμένες σε XML/HTTP. Μέχρι σήμερα τέτοιες εφαρμογές έχουν περιγραφεί κυρίως χρησιμοποιώντας έναν συνδυασμό κειμενικής περιγραφής και σχήματος XML, στόχος του WADL είναι να παραθέσει μια επεξεργάσιμη περιγραφή μηχανών αυτών των εφαρμογών με ένα απλούστερο σχήμα από αυτό που υπάρχει όταν χρησιμοποιείται τοWSDL.

Εδώ είναι ένα παράδειγμα WADL περιγραφής της εφαρμογής αναζήτησης ειδήσεων του Yahoo για να διευκρινιστεί η γλώσσα:

```
<?xml version="1.0" standalone="yes"?>
<application targetNamespace="urn:yahoo:yn"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:yn="urn:yahoo:yn"
 xmlns:tns="urn:yahoo:yn"
 xmlns:ya="urn:yahoo:api"
 xmlns="http://research.sun.com/wadl">

 <types>
   <include
     href="http://.../NewsSearchService/V1/NewsSearchResponse.xsd"/>
   <include
     href="http://.../Api/V1/error.xsd"/>
 </types>

 <resources>
   <resource uri="http://.../NewsSearchService/V1/newsSearch">
     <operationRef ref="tns:NewsSearch"/>
   </resource>
 </resources>

 <operation name="NewsSearch" method="get">
   <request>
     <parameter name="appid" type="xsd:string" required="true"/>
     <parameter name="query" type="xsd:string" required="true"/>
     <parameter name="type" type="xsd:string"/>
     <parameter name="results" type="xsd:int"/>
     <parameter name="start" type="xsd:int"/>
     <parameter name="sort" type="xsd:string"/>
     <parameter name="language" type="xsd:string"/>
   </request>
   <response>
     <representation mediaType="text/xml" element="yn:ResultSet">
       <parameter name="totalResults"
         type="xsd:nonNegativeInteger"
         path="/ResultSet/@totalResultsAvailable"/>
       <parameter name="resultsReturned"
         type="xsd:nonNegativeInteger"
         path="/ResultSet/@totalResultsReturned"/>
       <parameter name="resultPosition"
         type="xsd:nonNegativeInteger"
         path="/ResultSet/@firstResultPosition"/>
       <parameter name="results" path="/ResultSet/Result"/>
     </representation>
     <fault name="SearchError" status="400"
       mediaType="text/xml" element="ya:Error">
```

```
48          <parameter name="msg" path="/Error/Message"
49            type="xsd:string"/>
50        </fault>
51      </response>
52    </operation>
53</application>
```

Οι γραμμές 2-8 αρχίζουν με μια περιγραφή της εφαρμογής και καθορίζουν τα XML namespaces που χρησιμοποιούνται στην περιγραφή υπηρεσιών. Οι γραμμές 10-15 καθορίζουν τα σχήματα XML που χρησιμοποιούνται από την υπηρεσία, σε αυτήν την περίπτωση δύο σχήματα περιλαμβάνονται από την αναφορά. Οι γραμμές 17-21 περιγράφουν τον πόρο Ιστού αναζήτησης ειδήσεων του Yahoo και τις διαδικασίες που υποστηρίζει. Οι γραμμές 23-52 περιγράφουν τη λειτουργία NewsSearch: οι γραμμές 24-32 περιγράφουν τις εισαγωγές στη λειτουργία και τέλος οι γραμμές 33-51 περιγράφουν τα πιθανά αποτελέσματα της λειτουργίας.

Η προδιαγραφή WADL και το σχήμα WADL περιγράφουν τα χαρακτηριστικά γνωρίσματα της γλώσσας λεπτομερώς αλλά μερικά σημεία αξίζουν να σημειωθούν:

- Το βασικό σύνολο των μεθόδων HTTP (GET, POST, PUT, DELETE, HEAD) υποστηρίζεται από το σχήμα WADL αλλά η απαρίθμηση μεθόδων είναι ανοικτή στη χρήση και άλλων μεθόδων όπως εκείνες που διευκρινίζονται από το WebDAV.
- Οι παράμετροι αντιπροσώπευσης είναι hints στους επεξεργαστές που επισημαίνουν τα ενδιαφέροντα μέρη μιας αντιπροσώπευσης των πόρων. Μπορούν να χρησιμοποιηθούν ή να αγνοηθούν όπως επιθυμείται. Θα είναι χρήσιμοι για τις RPC-ομοειδείς αλληλεπιδράσεις αλλά λιγότερο στην προσανατολισμένη ως προς το έγγραφο εργασία.
- Τα στοιχεία αντιπροσώπευσης sibling αντιπροσωπεύουν τις εναλλακτικές αντιπροσωπεύσεις του ίδιου πόρου. Αυτό σημαίνει ότι μπορεί να περιγραφεί ένας πόρο που προσφέρεται με εναλλασσόμενα σχήματα, π.χ. XML ή HTML.
- Το σχεδιασμένο κέντρο είναι XML/HTTP αλλά αυτό δεν αποκλείει τη χρήση εναλλασσόμενων σχημάτων αντιπροσώπευσης: η ιδιότητα representation/@mediaType παρέχει το απαραίτητο στοιχείο.

Κεφάλαιο 3: WCF εφαρμογές

3.1 Εισαγωγή

Υπάρχουν διάφορες υπάρχουσες προσεγγίσεις στην δημιουργία των διανεμημένων εφαρμογών. Αυτές περιλαμβάνουν τις υπηρεσίες Ιστού, το . Net Remoting, το , Message Queuing (MSMQ) and COM+/Enterprise Services. Το Windows Communication Foundation (WCF) ενοποιεί αυτές τις τεχνολογίες σε ένα ενιαίο πλαίσιο για τη δημιουργία και την κατανάλωση υπηρεσιών. Η Microsoft εισήγαγε αρχικά WCF ως τμήμα του .NET Framework 3.0 και έχει συνεχίσει να το ενισχύει για το .NET Framework 3.5 και το Visual Studio 2008 όπως και για το Visual Studio 2010.

Στο κεφάλαιο που ακολουθεί θα μιλήσουμε εκτενώς για το Windows Communication Foundation (WCF). Αρχικά θα μιλήσουμε για το τι ακριβώς είναι και για τη διαδικασία που ακολουθεί. Θα αναφερθούμε στους λόγους για τους οποίους διαφέρουν οι βασισμένες σε WCF υπηρεσίες με τις υπόλοιπες υπηρεσίες ιστού. Στη συνέχεια θα επεκταθούμε στη διαλειτουργικότητα αυτών των υπηρεσιών με προγενέστερες τεχνολογίες και υπηρεσίες που είναι βασισμένες σε άλλες πλατφόρμες όπως και στην αρχιτεκτονική που είναι βασισμένες. Επιπρόσθετα θα εστιάσουμε στη διάκριση μιας WCF υπηρεσίας και μίας WCF εφαρμογής πελάτη και στα αρχεία από τα οποία αποτελούνται όπως και στη μεταξύ τους επικοινωνία. Η διαδικασία αποστολής ενός αιτήματος του πελάτη περιγράφεται επισης. Τέλος θα δούμε άλλες πλευρές του WCF όπως τις επιλογές μηνύματος, τον έλεγχο της τοπικής συμπεριφοράς, την ασφάλεια και τις συναλλαγές.

3.2 Τι είναι το WCF;

Το Windows Communication Foundation η αλλιώς WCF είναι ένα framework που χρησιμοποιείται για τη δημιουργία υπηρεσιοστρεφών εφαρμογών. Το χρησιμοποιούμε για να στέλνει δεδομένα ως ασυγχρόνιστα μηνύματα από ένα service endpoint σε ένα άλλο. Ένα service endpoint μπορεί να είναι μέρος μιας συνεχούς διαθέσιμης υπηρεσίας που έχει ως host τον IIS (*Internet Information Services*) η μια άλλη εφαρμογή που μπορεί να επιτελέσει το ίδιο έργο. Ένα endpoint μπορεί να είναι ένας πελάτης της υπηρεσίας που στέλνει ένα request για δεδομένα από ένα service endpoint.

Η λειτουργία μίας WCF υπηρεσίας είναι όχι πολύπλοκη διαδικασία. Μία WCF υπηρεσία είναι απλώς ένα αντικείμενο που αποκαλύπτει ένα set από μεθόδους όπου η εφαρμογή του πελάτη μπορεί να τις δει. Όταν χτίζεται μια υπηρεσία που βασίζεται σε ένα contract δημιουργείται μία κλάση που υπερκαλύπτει το contract. Για να εκτελεστεί μια υπηρεσία πρέπει να της παρέχουμε ένα runtime environment για το αντικείμενό της το οποίο πρέπει να γίνει διαθέσιμο στις εφαρμογές των πελατών. Το runtime environment για το αντικείμενο που υπερκαλύπτει την υπηρεσία παρέχεται από την host εφαρμογή.

3.3 Οι διαφορές των υπηρεσιών ιστού σε σχέση με τις WCF υπηρεσίες ιστού

Έχοντας έρθει σε μία πρώτη επαφή με την έννοια του WCF δημιουργούνται εύλογες απορίες για τις διαφορές που έχουν οι άλλες υπηρεσίες ιστού με τις υπηρεσίες που είναι βασισμένες σε WCF. Υπάρχουν πολλές και άξιες αναφοράς. Οι κυριότερες είναι οι ακόλουθες:

- Αρχικά το WCF έχει μια επιλογή ASP.NET συμβατότητας για να επιτρέψει στις εφαρμογές WCF να προγραμματιστούν και να διαμορφωθούν όπως οι ASP.NET υπηρεσίες Ιστού, και καταφέρουν να μιμηθούν τη συμπεριφορά τους.
- οι υπηρεσίες Ιστού χρησιμοποιούν XmlSerializer αλλά το WCF DataContractSerializer που είναι καλύτερο σε απόδοση σε σύγκριση με τον XmlSerializer.
- Οι υπηρεσίες Ιστού μπορούν να φιλοξενηθούν σε IIS καθώς επίσης και έξω από το IIS. Ενώ η υπηρεσία WCF μπορεί να φιλοξενηθεί σε IIS, σε Windows activation service(WAS), μπορεί ακόμα να χρησιμοποιηθεί και Self Hosting. Ακόμα μπορεί να γίνει χρήση πολλών και διάφορων πρωτόκολλων όπως Named Pipe, TCP και άλλων.
- Οι ιδιότητες των υπηρεσιών Ιστού προστίθενται στην κλάση. Σε ένα WCF θα υπάρξουν service contract attributes. Με τον ίδιο τρόπο Web Method attribute προστίθενται στην μέθοδο της υπηρεσίας Ιστού ενώ στο WCF ένα Service Operation Contract θα προστεθεί στην μέθοδο.
- Στην υπηρεσία Ιστού υποστηρίζεται το System.XML.Serialization ενώ στην υπηρεσία WCF υποστηρίζεται το System.RunTime.Serialization.
- Οι υπηρεσίες WCF μπορούν να είναι πολύπλοκες μέσω της κλάσης ServiceBehavior ενώ η υπηρεσία Ιστού δεν μπορεί να είναι.
- Οι υπηρεσίες WCF υποστηρίζουν διαφορετικούς τύπους συνδέσεων όπως BasicHttpBinding, WSHttpBinding, WSDualHttpBinding και τα λοιπά ενώ οι υπηρεσίες ιστού χρησιμοποιούν μόνο SOAP η XML για αυτό.

Κάποιες επιπλέον διαφορές διαφαίνονται στο ακόλουθο πινακάκι:

Χαρακτηριστικά	Υπηρεσίες ιστού	WCF
Hosting	Μπορεί να φιλοξενηθούν σε IIS	Μπορεί να φιλοξενηθεί σε IIS, windows activation service, Self-hosting, Windows service
Programming	[WebService] η ιδιότητα πρέπει να προστεθεί στην κλάση	[ServiceContract] η ιδιότητα πρέπει να προστεθεί στην κλάση
Model	[WebMethod] αντιπροσωπεύει τη μέθοδο που εκτίθεται στον πελάτη	[OperationContract] αντιπροσωπεύει τη μέθοδο που εκτίθεται στον πελάτη
Operation	Η μονόδρομη, απάντηση-αιτήματος είναι οι διαφορετικές διαδικασίες που υποστηρίζονται στην	Η μονόδρομη, αίτημα-απάντηση, διπλής κατεύθυνσης είναι διαφορετικοί τύποι

	υπηρεσία Ιστού	διαδικασιών που υποστηρίζονται σε ένα WCF
XML	Το System.Xml.serialization name space χρησιμοποιείται serialization	Το System.Runtime.Serialization namespace χρησιμοποιείται για serialization
Encoding	XML 1.0, MTOM(Message Transmission Optimization Mechanism), DIME, Custom	XML 1.0, MTOM, Binary, Custom
Transports	Μπορούμε να έχουμε πρόσβαση χρησιμοποιώντας HTTP, TCP, Custom	Μπορούμε να έχουμε πρόσβαση χρησιμοποιώντας HTTP, TCP, Named pipes, MSMQ,P2P, Custom
Protocols	Ασφάλεια	Ασφάλεια, αξιόπιστα μηνύματα, συναλλαγές

Γενικότερα η υπηρεσία Ιστού είναι ένα μέρος του WCF. Το WCF παρέχει περισσότερη ευελιξία και φορητότητα για να αναπτύξει μια υπηρεσία σε σύγκριση με την υπηρεσία Ιστού

3.4 Διαλειτουργικότητα με τις εφαρμογές που στηρίζονται σε άλλες τεχνολογίες

Πριν αναφερθούμε εκτενέστερα στην δομή μίας WCF υπηρεσίας όπως και μίας WCF εφαρμογής πελάτη και επικεντρωθούμε στα αρχεία τα οποία τις απαρτίζουν και εξηγήσουμε τη μεταξύ τους επικοινωνία θα ήταν σημαντικό να παρατάξουμε μερικά στοιχεία για την διαλειτουργικότητα του WCF σε σχέση με άλλες εφαρμογές που βασίζονται σε άλλες τεχνολογίες .

Απεικονίζοντας την ετερογένεια των περισσότερων επιχειρήσεων, το WCF είναι σχεδιασμένο για να επικοινωνεί καλά και με υπηρεσίες που δεν υποστηρίζουν αυτή την τεχνολογία. Υπάρχουν δύο σημαντικές πτυχές αυτού: αρχικά η διαλειτουργικότητα με τις πλατφόρμες που δημιουργούνται από άλλες εταιρίες, και έπειτα η διαλειτουργικότητα με τις τεχνολογίες της Microsoft που προηγήθηκαν από το WCF. Τα επόμενα τμήματα περιγράφουν και τις δύο περιπτώσεις.

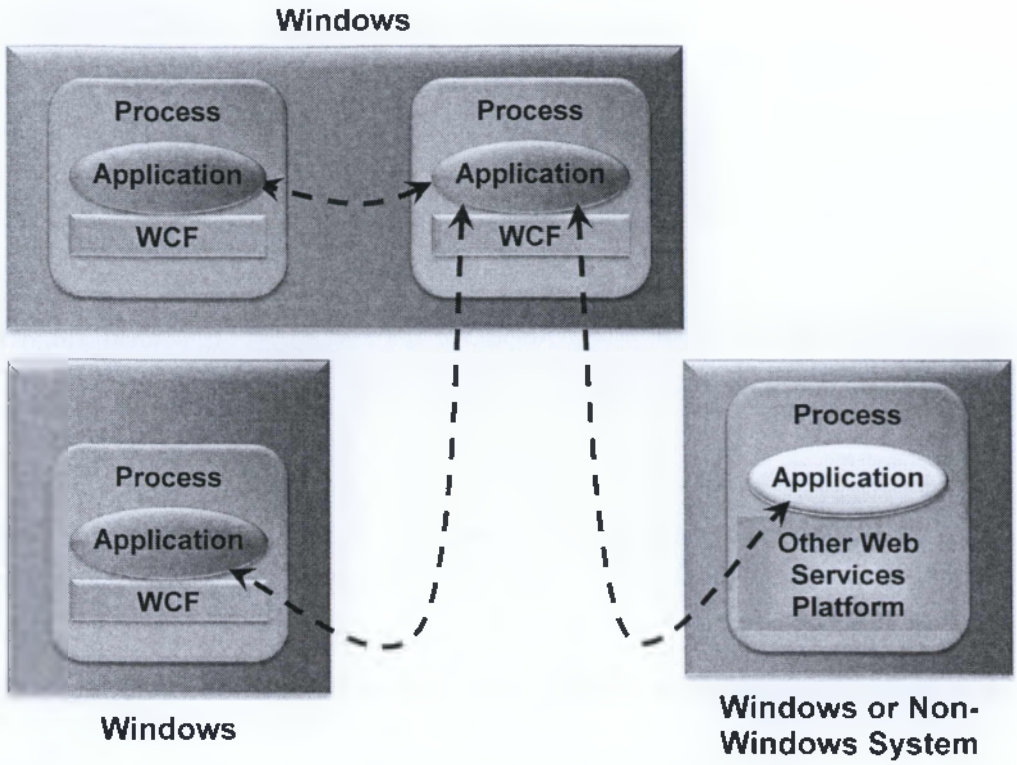
1. *Διαλειτουργικότητα με άλλες πλατφόρμες υπηρεσιών Ιστού*

   Οι επιχειρήσεις σήμερα συνήθως έχουν συστήματα και εφαρμογές που αγοράστηκαν από μια σειρά προμηθευτών. Παραδείγματος χάριν, η επικοινωνία μιας εταιρίας με άλλες απαιτεί διάφορες εφαρμογές λογισμικού που γράφονται σε διάφορες γλώσσες και τρέχουν σε διάφορα λειτουργικά συστήματα. Αυτό το είδος ποικιλομορφίας είναι μια πραγματικότητα σε πολλές οργανώσεις, και θα παραμείνει έτσι για το εγγύς μέλλον. Ομοίως, οι εφαρμογές που παρέχουν τις υπηρεσίες στο διαδίκτυο μπορούν να στηριχτούν σε οποιαδήποτε πλατφόρμα. Οι πελάτες που αλληλεπιδρούν με αυτές, πρέπει να είναι σε θέση να επικοινωνούν σε ό, τι στυλ είναι απαραίτητο.

   Οι βασισμένες σε WCF εφαρμογές μπορούν να επικοινωνήσουν με άλλο λογισμικό που τρέχει σε διάφορα πλαίσια. Όπως φαίνεται στο σχήμα, μια εφαρμογή που στηρίζεται σε WCF μπορεί να αλληλεπιδράσει με όλα τα παρακατω :

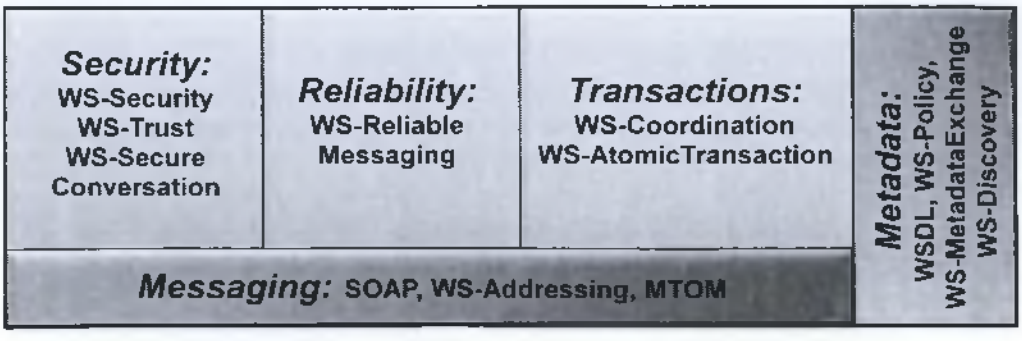
   a. Εφαρμογές βασισμένες σε WCF που τρέχουν σε μια διαφορετική διαδικασία στην ίδια μηχανή Windows.
   b. Εφαρμογές βασισμένες σε WCF που τρέχουν σε μια άλλη μηχανή Windows.

c. Εφαρμογές που στηρίζονται σε άλλες τεχνολογίες, όπως οι κεντρικοί υπολογιστές της Java EE, οι οποίοι υποστηρίζουν τις τυποποιημένες υπηρεσίες Ιστού. Αυτές οι εφαρμογές μπορούν να τρέχουν σε μηχανές Windows ή σε μηχανές άλλων λειτουργικών συστημάτων, όπως ο SUN, Solaris, IBM z/OS, ή Linux.



Σχήμα 3.4. 1 Applications built on WCF can communicate with other WCF-based applications or with applications built on other Web services platforms

Για να επιτρέψει κάτι παραπάνω από τη βασική επικοινωνία, το WCF υλοποιεί τις τεχνολογίες υπηρεσιών Ιστού που καθορίζονται από τις WS - * προδιαγραφές. Όλες αυτές οι προδιαγραφές καθορίστηκαν αρχικά από τη Microsoft, την IBM, και άλλες εταιρίες που εργάζονται από κοινού. Δεδομένου ότι οι προδιαγραφές έχουν γίνει σταθερές, η ιδιοκτησία έχει περάσει χαρακτηριστικά στους οργανισμούς προτύπων όπως Organization for the Advancement of Structured Information Standards(OASIS). Όπως το σχήμα 2 παρουσιάζει, αυτές οι προδιαγραφές εξετάζουν διαφορετικές περιοχές, συμπεριλαμβανομένου του βασικού μηνύματος, της ασφάλειας, της αξιοπιστίας, των συναλλαγών, και της εργασίας με τα μεταδεδομένα των υπηρεσιών.



Σχήμα 3.4. 2 WCF implements a range of Web Services standards

Το WCF υποστηρίζει όλες τις προδιαγραφές που παρουσιάζονται σε αυτό το σχήμα. Ομαδοποιημένα από τη λειτουργία, αυτά τα specs είναι:

a. *μήνυμα*: Το SOAP είναι το θεμελιώδες πρωτόκολλο για τις υπηρεσίες Ιστού, που καθορίζουν έναν βασικό φάκελο που περιέχει μια επικεφαλίδα και ένα σώμα. Η WS-Addressing καθορίζει τις προσθήκες στην επικεφαλίδα SOAP για την εξέταση των μηνυμάτων SOAP, η οποία ελευθερώνει το SOAP από τη στήριξη στο πρωτόκολλο μεταφορών, όπως το HTTP, για να φέρει τις πληροφορίες. Ο μηχανισμός βελτιστοποίησης μετάδοσης μηνυμάτων (Message Transmission Optimization Mechanism- MTOM) καθορίζει ένα βελτιστοποιημένο σχήμα μετάδοσης για τα μηνύματα SOAP βασισμένα στη βελτιστοποιημένη XML-binary Optimized Packaging (XOP).
b. *μεταδεδομένα*: Η Web Services Description Language (WSDL) καθορίζει μια τυποποιημένη γλώσσα για τον ορισμό των υπηρεσιών και των διάφορων πτυχών για το πώς εκείνες οι υπηρεσίες μπορούν να χρησιμοποιηθούν. Η WS-Policy επιτρέπει την προδιαγραφή των δυναμικότερων πτυχών της συμπεριφοράς υπηρεσιών που δεν μπορούν να εκφραστούν σε WSDL, όπως μια προτιμημένη επιλογή ασφάλειας. Το WS-MetadataExchange επιτρέπει σε έναν πελάτη να ζητήσει τις περιγραφικές πληροφορίες για μια υπηρεσία, όπως το WSDL του και οι πολιτικές του, μέσω του SOAP. Αρχίζοντας με το .NET Framework 4, το WCF υποστηρίζει επίσης την WS-Discovery. Αυτό το βασισμένο στη μετάδοση πρωτόκολλο αφήνει μια εφαρμογή να βρεί τις διαθέσιμες υπηρεσίες αλλού στο τοπικό δίκτυό.
c. *ασφάλεια*: Η WS-Security, η WS-Trust και η WS-SecureConversation καθορίζουν τις προσθήκες στα μηνύματα SOAP για την παροχή της επικύρωσης, της ακεραιότητας, της ιδιωτικότητας στοιχείων και άλλων χαρακτηριστικών γνωρισμάτων ασφάλειας.
d. *αξιοπιστία*: Το WS-ReliableMessaging καθορίζει τις προσθήκες στην επικεφαλίδα SOAP που επιτρέπουν την αξιόπιστη end to end επικοινωνία, ακόμα και όταν πρέπει να προσπελαστούν ένας ή περισσότεροι μεσάζοντες SOAP.
e. *συναλλαγές*: Στηριγμένο στο WS-Coordination, και το WS-AtomicTransaction επιτρέπει τη χρήση διφασικής δέσμευσης των συναλλαγών με τις βασισμένες στο SOAP ανταλλαγές.

Μία εφαρμογή θα χρησιμοποιούσε πιθανώς αρκετές από αυτές τις πιό προηγμένες τεχνολογίες. Παραδείγματος χάριν, η WS-Addressing είναι σημαντική οποτεδήποτε το SOAP τρέχει μέσω ενός πρωτόκολλου εκτός από το HTTP, η οποία μπορεί να είναι η περίπτωση της επικοινωνίας με τη βασισμένη στη .NET Framework εφαρμογή κλήσης κεντρικών πελατών. Το WCF στηρίζεται στην WS-Policy και τη WS-MetadataExchange για να ανακαλύψει εάν το σύστημα με το οποίο επικοινωνεί χρησιμοποιεί επίσης WCF και για άλλα πράγματα. Η αξιόπιστη επικοινωνία είναι απαραίτητη για τις περισσότερες καταστάσεις, έτσι είναι πιθανό το WS-ReliableMessaging να χρησιμοποιηθεί για να αλληλεπιδράσει με πολλές από τις άλλες εφαρμογές αυτού του σεναρίου. Ομοίως, η WS-Security και οι σχετικές προδιαγραφές μπορεί να χρησιμοποιηθούν για την επικοινωνία με μια ή περισσότερες από τις εφαρμογές, δεδομένου ότι όλοι θα απαιτούσαν κάποια ασφάλεια. Για τις εφαρμογές που επιτρέπεται να χρησιμοποιούν τις συναλλαγές με τη δική μας εφαρμογή, η WS-AtomicTransaction θα ήταν αναγκαια. Τέλος, η MTOM θα μπορούσε να χρησιμοποιηθεί όποτε ένα βελτιστοποιημένο σχήμα καλωδίων θα είχε νόημα και οι δύο πλευρές της επικοινωνίας θα όφειλαν να υποστηρίξουν αυτήν την επιλογή.

Το βασικό σημείο είναι ότι το WCF εφαρμόζει τις διαλειτουργικές βασισμένες στο SOAP υπηρεσίες Ιστού, πληρώντας την ασφάλεια , την αξιοπιστία, τις συναλλαγές, και άλλα. Για να αποφύγει μια περιττή ποινική ρήτρα απόδοσης, η επικοινωνία WCF-to-WCF μπορεί επίσης να είναι βελτιστοποιηθεί (το τυποποιημένο βασισμένο σε XML, SOAP δεν απαιτείται πάντα). Στην

πραγματικότητα, είναι εύκολο ακόμα και για μια ενιαία εφαρμογή να εκθέσει τις υπηρεσίες της και στα δύο είδη πελατών.

Επίσης, το WCF υποστηρίζει το Rest. Παρά την αποστολή των μηνυμάτων SOAP άνω του HTTP, η RESTful επικοινωνία στηρίζεται άμεσα στα ενσωματωμένα ρήματα του HTTP: GET, POST, και άλλα. Ενώ το SOAP είναι η σωστή προσέγγιση για μερικά είδη διαλειτουργικότητας, όπως καταστάσεις που απαιτούν την πιο προηγμένη ασφάλεια ή τις συναλλαγές μιας υπηρεσίας. Η RESTful επικοινωνία είναι μια καλύτερη επιλογή σε άλλες περιπτώσεις. Συνεπώς, το WCF υποστηρίζει και τις δύο προσεγγίσεις.

2. *Διαλειτουργικότητα με τις τεχνολογίες πριν από το Wcf της Microsoft*

Πολλοί από τους πελάτες της Microsoft έχουν κάνει σημαντικές επενδύσεις στις τεχνολογίες .NET Framework που το WCF εντάσσει. Η προστασία εκείνων των επενδύσεων ήταν ένας θεμελιώδης στόχος των σχεδιαστών του WCF. Η εγκατάσταση του WCF δεν σπάζει την υπάρχουσα τεχνολογία, έτσι δεν υπάρχει καμία απαίτηση οι οργανώσεις να αλλάξουν τις υπάρχουσες εφαρμογές για να το χρησιμοποιήσουν. Μια πορεία βελτίωσης παρέχεται, εντούτοις, και οπουδήποτε είναι δυνατόν, το σWCF επικοινωνεί με εκείνες τις προηγούμενες τεχνολογίες.

Για παράδειγμα, και το WCF και το ASMX χρησιμοποιούν SOAP, έτσι οι βασισμένες στο WCF εφαρμογές μπορεί άμεσα να επικοινωνήσουν με εκείνες που στηρίζονται σε ASMX. Οι υπάρχουσες εφαρμογές επιχειρηματικών υπηρεσιών μπορούν επίσης να συνδυαστούν με τις WCF διεπαφές, επιτρέποντας σε αυτές να επικοινωνήσουν με τις εφαρμογές που στηρίζονται σε WCF. και επειδή η αναμονή του WCF στηρίζεται σε MSMQ, οι βασισμένες σε WCF εφαρμογές μπορούν να επικοινωνήσουν άμεσα με εφαρμογές που δεν βασίζονται σε WCF και χτίζονται χρησιμοποιώντας τις εγγενείς διεπαφές MSMQ όπως το System.Messaging. Στην εφαρμογή μας, για παράδειγμα, το λογισμικό που χτίζεται χρησιμοποιώντας οποιεσδήποτε από αυτές τις προηγούμενες τεχνολογίες που θα μπορούσαν άμεσα να συνδεθούν και να χρησιμοποιήσουν τις βασισμένες σε WCF υπηρεσίες του νέου συστήματος.

Η διαλειτουργικότητα δεν είναι πάντα δυνατή. Για παράδειγμα, ακόμα κι αν το WSE 1,0 και το WSE 2,0 εφάρμοζαν μερικές από τις ίδιες WS - * προδιαγραφές όπως το WCF, αυτές οι προηγούμενες τεχνολογίες στηρίζονται στις προηγούμενες εκδόσεις των specs. Λόγω αυτού, μόνο το WSE 3,0 μπορεί να επικοινωνήσει με το WCF(πιό πρόωρες εκδόσεις δεν μπορούν).

3.5 Υπηρεσιοστραφής αρχιτεκτονική (Service oriented architecture - SOA)

Η υπηρεσιοστραφής αρχιτεκτονική βασίζεται στη λογική "Software as a Service" και έχει ως κύριο στόχο την ανταγωνιστικότητα και επιπρόσθετα να είναι δυνατή η επαναχρησιμοποίηση του υπάρχοντος λογισμικού κάνοντάς το ενεργό με οποιοδήποτε τρόπο λειτουργίας.

Το SOA αποτελείται από πηγές στο δίκτυο που είναι διαθέσιμες ως ανεξάρτητες υπηρεσίες και μπορούμε να έχουμε πρόσβαση σε αυτές χωρίς να απαιτείται γνώση για το πώς αυτές υπερκαλύφθηκαν. Μπορεί να συνδυαστούν υπηρεσίες σε ένα SOA για να δημιουργήσουν μια νέα εφαρμογή. Με βάση το SOA μπορούν να δημιουργηθούν solutions τα οποία είναι ανεξάρτητα από την πλατφόρμα και την τοποθεσία.

Το WCF είναι ιδανική πλατφόρμα για την υπερκάλυψη του SOA(γι αυτό κρίνεται και αναγκαία η αναφορά της συγκεκριμένης αρχιτεκτονικής σε αυτό το κεφάλαιο) αφού είναι δυνατός ο συνδυασμός εφαρμογών που προϋπήρχαν και να συνδεθούν με τρόπους που πριν δεν ήταν εύκολο να επιτύχουμε. Ακόμα μέσω του WCF μπορούν να συνδυαστούν εφαρμογές με συστατικά. Τέλος υπάρχει η δυνατότητα να χρησιμοποιηθούν :

1. Standard πρωτόκολλα
2. Μορφές δεδομένων
3. Μηχανισμούς επικοινωνίας

Ετσι ώστε οι υπηρεσίες να έχουν διαλειτουργικότητα με υπηρεσίες που χρησιμοποιούν άλλες τεχνολογίες.

Βασικές αρχές υπηρεσιοστραφούς αρχιτεκτονικής

1. Τα όρια πρέπει να είναι σαφής: Ακολουθώντας αυτή την αρχή θα αποφευχθούν οι εξαρτήσεις μεταξύ των υπηρεσιών και των εφαρμογών των πελατών.
2. Οι υπηρεσίες να είναι αυτόνομες: Η τοποθεσία μιας υπηρεσίας ιστού μπορεί να αλλάξει η μια υπηρεσία μπορεί προσωρινά να είναι offline για κάποιους λόγους. Εάν δεν υπάρχουν εξαρτήσεις η υπηρεσία μας θα εξακολουθήσει να τρέχει ακόμα και αν οι άλλες υπηρεσίες δεν είναι διαθέσιμες.
3. Οι υπηρεσίες να μοιράζονται schemas και contracts και όχι κλάσεις ή τύπους: Σχεδιάζοντας schemas και contracts μειώνονται οι εξαρτήσεις οπότε σε περίπτωση που μία υπηρεσία ενημερωθεί θα διατηρήσει τη συμβατότητα με τους υπάρχοντες πελάτες επειδή οι πελάτες θα συνεχίζουν να υπερκαλύπτουν τα υπάρχοντα contracts και στέλνουν μηνύματα που να είναι συμβατά με το schema.
4. Η συμβατότητα βασίζεται στην πολιτική: Τα schemas και τα contracts μιας υπηρεσίας ορίζουν το σχήμα της αλλά όχι και τις μη λειτουργικές της απαιτήσεις. Παράδειγμα μη λειτουργικής απαίτησης είναι η policy η οποία δεν πρέπει να ορίζεται χρησιμοποιώντας contracts. Δεν απαιτεί πρόσθετο κώδικα στον πελάτη η στην υπηρεσία. Συνήθως βρίσκεται σε κάποιο configuration file αλλά είναι απαραίτητο οι υπηρεσίες και οι εφαρμογές των πελατών να είναι σύμφωνες με την policy information.

3.6 Τα αρχεία που περιέχονται μέσα στις WCF υπηρεσίες τις εφαρμογές πελατών

Μία WCF υπηρεσία απαρτίζεται από τρία μέρη. Το πρώτο μέρος είναι το interface το οποίο περιέχει όλα τα contracts που πρέπει να υλοποιηθούν. Ουσιαστικά καθορίζει το σχήμα της υπηρεσίας. Ορίζει ποια δεδομένα θα γίνουν serialized από το WCF. Όσα δεδομένα δεν χαρακτηριστούν ως DataMember δεν μπορούν να γίνουν serilized. Από το συγκεκριμένο μέρος παράγεται και το WSDL.

Το δεύτερο μέρος είναι η υλοποίηση του πρώτου. Είναι σημαντικό ν' αναφερθούμε στο γεγονός ότι πρέπει να υλοποιηθούν όλες οι λειτουργίες που βρίσκονται σε operation contracts. Τέλος η εφαρμογή δημιουργεί αυτόματα την proxy class η οποία είναι ο συνδετικός κρίκος μεταξύ της υπηρεσίας και του client.

Πιο αναλυτικά στο αρχείο που ορίζεται το Interface βρίσκονται τα contracts που θα υπερκαλύψει η υπηρεσία και μετά η υπηρεσία χτίζεται πάνω σε αυτά. Σκοπός είναι να μπορούμε να συγκεντρωθούμε από την αρχή στη δομή της υπηρεσίας. Το πλεονέκτημα της χρήσης τους είναι ότι επανεξετάζονται γρήγορα για να δούμε εάν υπάρχουν εξαρτήσεις από κάποιο software ή hardware πριν αναπτυχθεί πλήρως. Τα contracts επίσης ορίζουν ποιες λειτουργίες το WCF θα

υπερκαλύπτει. Ουσιαστικά μέσα σε αυτό το αρχείο ορίζονται τα δεδομένα που θα στέλνονται στην εφαρμογή του πελάτη μέσω της WCF υπηρεσίας.Τα contracts χωρίζονται σε:

1. Data contracts
2. Service contracts
3. Operation contracts
4. Fault contracts
5. Message contracts

Data contracts: Ορίζουν τους τύπους που πρέπει να γίνουν serialized και deserialized ως XML ροή από το WCF. (Όσοι τύποι μεταβλητών δεν χαρακτηριστούν με κάποιο τύπο data member δεν μπορούν να γίνουν serialized από το WCF.)
Service contracts: Ορίζοντας ένα service contract σαν interface υπάρχει η δυνατότητα να διαχωρίζουμε τον ορισμό του contract από την υλοποίηση του.
Operation contracts: ορίζουμε όποια μέθοδο θέλουμε να χρησιμοποιηθεί για να βασιστεί το WCF κατά το rumtime για να παράγει metadata για την εφαρμογή του πελάτη που επιθυμεί να χρησιμοποιήσει η υπηρεσία.

Ουσιαστικά τα data contracts ορίζουν τη δομή των δεδομένων που θα εισέρχονται στο WCF ενώ τα service contracts το σχήμα που θα έχει.
Fault contracts: Καθορίζονται ποια λάθη αυξάνονται από την υπηρεσία, και πώς η υπηρεσία χειρίζεται και διαδίδει τα λάθη στους πελάτες της.
Message contracts: Επιτρέπουν στην υπηρεσία να αλληλεπιδράσει άμεσα με μηνύματα. Οι συμβάσεις μηνυμάτων μπορούν να είναι typed ή untyped, και είναι χρήσιμες σε περιπτώσεις διαλειτουργικότητας και όταν υπάρχει ένα υπάρχον σχήμα μηνυμάτων και εμείς πρέπει να συμμορφωθούν με αυτό.

3.7 Πως επικοινωνούν μία WCF υπηρεσία και μια WCF εφαρμογή πελάτη.

Εκτός από τα αρχεία που απαρτίζουν το WCF είναι ενδιαφέρον και ο τρόπος με τον οποίο επικοινωνούν οι δύο εφαρμογές. Η επικοινωνία μεταξύ της WCF εφαρμογής του πελάτη και της WCF υπηρεσίας βασίζεται στα contracts, την proxy class και τα endpoints. Οι δύο εφαρμογές πρέπει να έχουν τα ίδια contracts. Όσο αφορά την proxy class είναι μία κλάση η οποία παράγεται από το metadata endpoint και παρέχει τις ίδιες operations και μηνύματα με το WCF αλλά τις κάνει διαθέσιμες σαν μεθόδους, παραμέτρους και επιστρέφει τιμές . Ο κώδικας της proxy class μετατρέπει τα response messages και τις κλήσεις μεθόδων σε request messages και μετατρέπει τα response messages που δέχεται από το WCF σε κλήση μεθόδων.

Endpoints:

Το endpoint παρέχει πληροφορίες για την υπηρεσία που χρειάζεται η εφαρμογή το πελάτη για να επικοινωνήσει με το WCF. Είναι το σημείο που μπορούν να στέλνουν τα request τους οι πελάτες. Το endpoint μπορεί να έχει ένα προαιρετικό όνομα όπου μπορεί να αναφέρεται στον κώδικα του πελάτη. Αυτό είναι χρήσιμο εάν η πελατειακή εφαρμογή συνδέεται με πολλαπλές υπηρεσίες γιατί κάθε υπηρεσία έχει το δικό της endpoint με όνομα στο configuration file του πελάτη.Το endpoint αποτελείται από τρία μέρη: το address, το binding και το contract.

Address:

Η μορφή της διεύθυνσης της υπηρεσίας εξαρτάται από διάφορους παράγοντες συμπεριλαμβανομένου και του πρωτόκολλου μεταφοράς που χρησιμοποιείται. Διαφορετικοί μηχανισμοί μεταφοράς χρησιμοποιούν διαφορετικά address spaces. Για παράδειγμα χρησιμοποιώντας τον IIS (*Internet Information Services*) δημιουργείται μια διεύθυνση: http://localhost/ProductsServices/ProductsService.svc.
Η διεύθυνση αυτή υποδεικνύει ένα virtual directory και το service definition file(svc). Εάν φτιάξουμε τη δική μας host applicationμπορεί να χρησιμοποιηθεί διαφορετικός μηχανισμός μεταφοράς και πρέπει να ορισθεί μια διεύθυνση που να είναι κατάλληλη για τον επιλεγμένο μηχανισμό.

Binding:

Το binding περιγράφει πως ο πελάτης μπορεί να συνδεθεί στην υπηρεσία και τη μορφή των δεδομένων που αναμένεται από την υπηρεσία. Ένα binding περιέχει:

I. Πρωτόκολλο μεταφοράς: Πρέπει να συμβαδίζει με τις απαιτήσεις της διεύθυνσης της υπηρεσίας. Για παράδειγμα εάν χρησιμοποιούμε τον IIS πρέπει να ορίσουμε το HTTP ή το HTTPs ως πρωτόκολλο μεταφοράς. Το WCF είναι επίσης φτιαγμένο να υποστηρίζει TCP protocol, named-pipes και message queues. Μπορούν να ορισθούν χρησιμοποιώντας transport schemas.
II. Κωδικοποιημένη μορφή μηνυμάτων: Σε πολλές περιπτώσεις τα request και τα response μηνύματα μεταδίδονται σε XML μορφή κωδικοποιημένα σαν κοινό κείμενο. Παρόλα αυτά σε μερικές περιπτώσεις ίσως χρειαστεί να μεταφερθούν δεδομένα χρησιμοποιώντας δυαδική κωδικοποίηση ιδιαιτέρως εάν μεταφέρονται εικόνες η ελεγχόμενη ροή δεδομένων.
III. Απαιτήσεις ασφαλείας μιας υπηρεσίας: Μπορεί να υλοποιηθεί ασφάλεια στο επίπεδο ασφαλείας και στο επίπεδο μηνυμάτων παρόλο που διαφορετικά πρωτόκολλα μεταφοράς έχουν δικούς τους περιορισμούς και απαιτήσεις.
IV. Απαιτήσεις συναλλαγής μιας υπηρεσίας : Μια υπηρεσία προσφέρει πρόσβαση σε μία η παραπάνω πηγές. Οι εφαρμογές των πελατών ανανεώνουν τις πηγές στέλνοντας αιτήματα στην υπηρεσία. Εάν ο πελάτης στείλει πολλαπλά αιτήματα στην υπηρεσία έχει ως αποτέλεσμα σε πολλαπλές ενημερώσεις. Είναι σημαντικό να διασφαλίσουμε ότι αυτές οι ενημερώσεις θα γίνουν μόνιμες. Σε περίπτωση αποτυχίας η υπηρεσία θα αναιρέσει όλες τις ενημερώσεις. Αυτός είναι ο ορισμός της συναλλαγής (transaction).
V. Η αξιοπιστία της επικοινωνίας της υπηρεσίας: Οι πελάτες συνήθως συνδέονται με τις υπηρεσίες μέσω ενός δικτύου. Τα δίκτυα είναι αβέβαια και μπορεί να αποτύχουν ανά πάσα στιγμή. Εάν ένας πελάτης ανταλλάσει μηνύματα με μία υπηρεσία πληροφορίες για την

αξιοπιστία της είναι απαραίτητες. Για παράδειγμα μία υπηρεσία θα πρέπει να μπορεί να διασφαλίσει ότι έχει δεχθεί όλα τα μηνύματα που έστειλε ο πελάτης και με τη σειρά που τα έστειλε. Μία υπηρεσία μπορεί να διασφαλίσει την integrity των συζητήσεων υπερκαλύπτωντας ένα αξιόπιστο πρωτόκολλο μηνυμάτων.

Contract:

Ένα WCF service contact είναι ένα interface store σε ένα .NET framework assembly tan notated με το Service Contract attribute. Το Service contract περιγράφει τις operations που υπερκαλύπτονται από την υπηρεσία κάνοντας tag αυτά με το Operation Contract attribute. Όποια πληροφορία περάσει προς και από τα operations πρέπει να γίνει serialized. Μια υπηρεσία μπορεί να ορίσει data contracts τα οποία περιγράφουν τη δομή από complex data και πως τα δεδομένα πρέπει να γίνουν serialized . Η υπηρεσία μπορεί να δημοσιοποιεί την περιγραφή του service contract , το οποίο μπορεί να χρησιμοποιήσει μια client application για ascertain τις operations που υλοποιεί η service και να στέλνει μηνύματα που είναι σωστά μορφοποιημένα.

3.8 Processing a Client Request.

Μία service μπορεί να ανταποκριθεί στα requests από πολλαπλές client applications ταυτόχρονα. Για να επιτευχθεί αυτό η application που κάνει hosting στην service πρέπει να μπορεί να δέχεται πολλαπλά εισερχόμενα request και να κατευθύνει service responses πίσω στον κατάλληλο client. Επιπλέον η host application πρέπει να διασφαλίζει τα μηνύματα που στέλνονται μεταξύ clients και services να ακολουθούν την πολιτική ασφαλείας, τη reliability και τις transactional απαιτήσεις που έχει το binding που χρησιμοποιείται. Αυτή η διαδικασία παρέχεται από το WCF runtime environment για την client application→ μία collection of channel objects.

Ένα κανάλι είναι υπεύθυνο να χειρίζεται μια πλευρά του message processing όπως ορίζεται από το binding μιας υπηρεσίας. πχ. Ένα transport channel διαχειρίζεται τις επικοινωνίες χρησιμοποιώντας ένα συγκεκριμένο transport protocol και ένα transaction channel ελέγχει το transactional integrity (ακεραιότητα) μιας συζήτησης. Το WCF runtime παρέχει ένα built-in channel για κάθε ένα από τα υποστηριζόμενα transport protocols.

Το WCF runtime επίσης παρέχει channels τα οποία χειρίζονται διαφορετικούς τρόπους που το WCF μπορεί να κωδικοποιήσει δεδομένα, χειρίζεται ασφάλεια, ολοκληρώνει reliability και perform transactions. Το WCF runtime συνθέτει channels σε μια channel stack. Όλα τα μηνύματα που περνάνε ανάμεσα από το client και τη υπηρεσία περνάνε από το κάθε channel στην channel stack και μεταφέρει το μήνυμα με κάποιο τρόπο ώστε το output από το ένα channel να περνάει σαν input στο επόμενο. Η channel stack λειτουργεί σε δυο κατευθύνσεις μηνύματα received από clients across the network proceed up στην channel stack προς την υπηρεσία και response μηνύματα που στέλνονται από την υπηρεσία διασχίζουν το channel stack προς την αντίθετη κατεύθυνση πίσω προς το δίκτυο και μετά στον client. Εάν το channel δεν μπορεί να στέλνει το μήνυμα εμφανίζει ένα error και το στέλνει στον client και το μήνυμα δεν επεξεργάζεται περεταίρω.

Υπάρχει σειρά στα channels στην channel stack. Το transport channel είναι το πρώτο που δέχεται τα δεδομένα και βρίσκεται στο κάτω μέρος της στοίβας . Στην κορυφή της stack βρίσκεται το encoding channel. Αυτά τα 2 κανάλια είναι υποχρεωτικά. Τα υπόλοιπα δεν είναι υποχρεωτικά.

Όταν ξεκινά να τρέχει μια εφαρμογή το WCFruntime χρησιμοποιεί το endpoint και της πληροφορίες που ορίζονται μέσα σ' αυτό σαν ένα μέρος της service configuration για να δημιουργήσει ένα listener object για κάθε address που ορίζεται για την service. Όταν δέχεται ένα

εισερχόμενο request το WCF runtime δημιουργεί μια channel stack για να χρησιμοποιήσει το binding information που ορίζεται για την address και τις διαδρομές της εισερχόμενης πληροφορίας από τον client προς την στοίβα. Εάν ένα μήνυμα διασχίσει με επιτυχία την channel stack το transformed request μεταφέρεται σαν στιγμιότυπο στην υπηρεσία για επεξεργασία.

Το WCF πρέπει να έχει τη δυνατότητα να χειρίζεται request από πολλαπλούς client application ταυτόχρονα. Για να επιτευχθεί αυτό : Πρέπει να δημιουργεί πολλαπλά concurrent instances της υπηρεσίας. Το WCF runtime δημιουργεί ένα Instance Context Object για να ελέγχει την αλληλεπίδραση μεταξύ της channel stack και του service instance. Μπορεί να τροποποιηθεί ο τρόπος με τον οποίο το WCF runtime instantiates ένα service instance μέσω τον Instance Context Object ορίζοντας το Service Behavior attribute της class υπερκαλύπτοντας το service contract. Το Service Behavior attribute => Instance Context Mode.

3.9 Άλλες πλευρές του WCF.

Τα βασικά στοιχεία των υπηρεσιών και των πελατών είναι θεμελιώδη σε κάθε εφαρμογή WCF. Οι περισσότερες εφαρμογές θα χρησιμοποιήσουν άλλες πτυχές αυτής της τεχνολογίας. Αυτό το τμήμα περιγράφει μερικές από αυτές τις πρόσθετες ικανότητες.

3.9.1 Επιλογές μηνύματος

Ο συνηθέστερος τρόπος που υποστηρίζεται από το WCF είναι η απομακρυσμένη κλήση διαδικασίας (RPC) στην αλληλεπίδραση πελατών/υπηρεσιών. Το WCF υποστηρίζει αυτήν την επιλογή, αλλά δεν είναι η μόνη. Το WCF υποστηρίζει διάφορες δυνατότητες μαζί με το παραδοσιακό RPC, συμπεριλαμβανομένων των εξής:

- Ασύγχρονο RPC, με τις non-blocking ταξινομημένες κατά ζεύγος κλήσεις που επιστρέφουν τους καταλόγους των δακτυλογραφημένων παραμέτρων.
- Παραδοσιακό μήνυμα, με τις non-blocking κλήσεις που επιστρέφουν μια ενιαία παράμετρο μηνυμάτων. Χρησιμοποιώντας άμεσα τα κανάλια εκθέτει τις μεθόδους για να στείλει και να λάβει τα μηνύματα, και WCF καθορίζει μια κλάση πρότυπων μηνυμάτων που μπορεί να χρησιμοποιηθεί για να επικοινωνεί άμεσα με τα μηνύματα XML.
- Άμεσος χειρισμός μηνυμάτων SOAP που χρησιμοποιούν τις WCF-καθορισμένες ιδιότητες του MessageContract. Αυτό γίνεται χρησιμοποιώντας δύο σχετικές ιδιότητες, το MessageHeader και το MessageBodyMember. Μια εφαρμογή μπορεί ρητά να έχει πρόσβαση στο περιεχόμενο της header και του body ενός SOAP μηνύματος.
  Για να επιτραπεί σε μια μέθοδο που στέλνει πληροφορίες αλλά δεν εμποδίζει την αναμονή μιας απάντησης, υπάρχει η ιδιότητα OperationContract που καλεί μια ιδιότητα την IsOneWay. Οι μέθοδοι που μαρκάρονται με αυτές τις ιδιότητες μπορούν να έχουν μόνο παράμετρους εισαγωγής και πρέπει να επιστρέψουν void. Για παράδειγμα:

```
[OperationContract(IsOneWay=true)]
void NewCarAvailable(int vehicleClass, int location);
```

Με αυτήν την μέθοδο, ο επισκέπτης δεν παίρνει τίποτα ως επιστροφή, είναι μονόδρομη επικοινωνία. Μια χαρακτηριστική χρήση των μονόδρομων μεθόδων όπως αυτή είναι για να σταλούν τα unsolicited events (εκούσια γεγονότα). Υποθέστε, παραδείγματος χάριν, ότι η εφαρμογή κεντρικών πελατών κλήσης ενημερώνεται κάθε φορά που ένα αυτοκίνητο είναι διαθέσιμο σε μια προηγούμενη

θέση ως sold-out. Εκείνος ο client θα έπρεπε να εφαρμόσει μια WCF σύμβαση υπηρεσιών που να περιέχει τη μέθοδο NewCarAvailable, και να αφήσει έπειτα την εφαρμογή κράτησης ενοικίου αυτοκινήτων να την καλέσει καθώς τα νέα αυτοκίνητα γίνονται διαθέσιμα.

Είναι επίσης δυνατό να συνδεθούν οι κλήσεις με τις μεθόδους, είτε μονόδρομες είτε κανονικές δυπλής κατεύθυνσης μέθοδοι, που επιτρέπουν και τις δύο πλευρές της επικοινωνίας για να ενεργήσουν ως πελάτης και υπηρεσία. Για να το κάνει αυτό, κάθε πλευρά εφαρμόζει μια σύμβαση που συνδέεται με το συνεργάτη της, με συνέπεια αυτά που καλούνται *δυπλές* συμβάσεις. Το WCF παρέχει τις ειδικές συνδέσεις για το χειρισμό αυτής της περίπτωσης, όπως το WsDualHttpBinding για δυπλές συμβάσεις που χρησιμοποιούν το τυποποιημένο διαλειτουργικό SOAP.

3.9.2Ελέγχοντας την τοπική συμπεριφορά

Πολλές πτυχές του WCF, όπως τα contracts, τα bindings και άλλα, συσχετίζονται με την επικοινωνία μεταξύ μιας υπηρεσίας και των πελατών της. Ακόμα υπάρχουν μέρη της συμπεριφοράς των υπηρεσιών που είναι τοπικά. Παραδείγματος χάριν, πώς η ταυτόχρονη πρόσβαση σε μια περίπτωση υπηρεσιών ρυθμίζεται και πώς η διάρκεια ζωής της περίπτωσης ελέγχεται; Για να επιτραπεί στους υπεύθυνους για την ανάπτυξη να θέσουν τοπικές συμπεριφορές όπως αυτή, το WCF καθορίζει δύο αρχικές ιδιότητες, οι οποίες με τη σειρά τους έχουν διάφορες ιδιότητες. Μια από αυτές τις ιδιότητες, η ServiceBehavior, μπορεί να εφαρμοστεί σε οποιαδήποτε κατηγορία υπηρεσιών. Άλλη όπως η OperationBehavior, μπορεί να εφαρμοστεί σε οποιαδήποτε μέθοδο σε μια κατηγορία υπηρεσιών που είναι μαρκαρισμένη με τις ιδιότητες της OperationContract.

Η ιδιότητα ServiceBehavior έχει τις ιδιότητες που έχουν επιπτώσεις στη συμπεριφορά της υπηρεσίας συνολικά. Παραδείγματος χάριν, μια ιδιοκτησία αποκαλούμενη ConcurrencyMode μπορεί να χρησιμοποιηθεί για να ελέγξει την ταυτόχρονη πρόσβαση στην υπηρεσία. Εάν θέστε single, το WCF θα παραδώσει μόνο ένα αίτημα πελατών τη φορά σε αυτήν την υπηρεσία, δηλαδή, η υπηρεσία θα είναι ενός νήματος. Εάν αυτή η ιδιότητα τίθεται στο Multiple, το WCF θα παραδώσει περισσότερα από ένα αιτήματα πελατών τη φορά στην υπηρεσία, κάθε μια θα τρέχει σε ένα διαφορετικό νήμα. Ομοίως, η ιδιότητα InstanceContextMode της ServiceBehavior μπορεί να χρησιμοποιηθεί για να ελέγξει πώς τα στιγμιότυπα μιας υπηρεσίας δημιουργούνται και καταστρέφονται. Εάν το InstanceMode τίθεται PerCall, μια νέα περίπτωση της υπηρεσίας θα δημιουργηθεί για να χειριστεί κάθε αίτημα πελατών, και θα καταστραφεί όταν το αίτημα ολοκληρώνεται. Παρόλα αυτά εάν τίθεται ως PerSession, το ίδιο στιγμιότυπο της υπηρεσίας θα χρησιμοποιηθεί για να χειριστεί όλα τα αιτήματα από έναν συγκεκριμένο πελάτη. (Για να γίνει αυτό απαιτεί επίσης τις ιδιότητες συνόδου ServiceContract και μια σύνδεση που υποστηρίζει τις συνόδους.) Τα InstanceContextMode μπορούν επίσης να τεθούν σε single, τα οποία αναγκάζουν μια μόνο περίπτωση της υπηρεσίας για να χειριστούν όλα τα αιτήματα από όλους τους πελάτες.

Υποθέστε, παραδείγματος χάριν, ότι ο υπεύθυνος για την ανάπτυξη αποφάσισε να καταστήσει την κατηγορία RentalReservations πολυνηματική και να την έχει για να τη χρησιμοποιήσει για όλες τις κλήσεις από όλους τους πελάτες. Ο καθορισμός της κατηγορίας θα έμοιαζε έπειτα με αυτό:

```
using System.ServiceModel;

[ServiceContract]
[ServiceBehavior(
  ConcurrencyMode=Multiple,
  InstanceContextMode=Single)]
        class RentalReservations { ... }
```

Ομοίως, οι ιδιότητες της OperationBehavior επιτρέπουν τη συμπεριφορά impersonation της μεθόδου που υλοποιεί μια συγκεκριμένη λειτουργία, οι απαιτήσεις των συναλλαγών της και άλλα πράγματα.

Μια άλλη πτυχή της τοπικής συμπεριφοράς είναι πώς ένα αίτημα από έναν πελάτη καθοδηγείται μέσα σε μια υπηρεσία WCF. Το WCF στο .NET Framework 4 προσθέτει μια τυποποίηση βασισμένη στο περιεχόμενο routing υπηρεσία που ο υπεύθυνος για την ανάπτυξη μπορεί να χρησιμοποιήσει τις βασισμένες στο SOAP υπηρεσίες. Η routing υπηρεσία βρίσκεται μπροστά από τις άλλες υπηρεσίες WCF, καθοδηγώντας κάθε εισερχόμενο μήνυμα σε μια από εκείνες τις υπηρεσίες που εδρεύουν στο περιεχόμενο του μηνύματος. Παραδείγματος χάριν, ένας υπεύθυνος για την ανάπτυξη μπορεί να χρησιμοποιήσει την routing υπηρεσία για να κρύψει τις διαφορετικές εκδόσεις μιας υπηρεσίας. Αυτό επιτρέπει σε έναν πελάτη να στέλνει πάντα στο ίδιο endpoint, ανεξάρτητα από την έκδοση της υπηρεσίας που στοχεύει, ενώ η routing υπηρεσία καθοδηγεί κάθε μήνυμα στη σωστή έκδοση της υπηρεσίας που εδρεύει στο περιεχόμενο του μηνύματος.

3.9.3 Ασφάλεια

Εκθέτοντας τις υπηρεσίες σε ένα δίκτυο, ακόμη και ένα εσωτερικό δίκτυο, απαιτεί συνήθως κάποιο είδος ασφάλειας. Πώς μπορεί η υπηρεσία να είναι σίγουρη για την ταυτότητα του πελάτη της; Πώς μπορούν τα μηνύματα που στέλνονται σε μια υπηρεσία και από μια υπηρεσία να κρατηθούν ασφαλή από κακόβουλους χρήστες; Και πώς μπορεί η πρόσβαση σε μια υπηρεσία να περιοριστεί μόνο σε εκείνους που εγκρίνονται για να το χρησιμοποιήσουν; Χωρίς κάποια λύση σε αυτά τα προβλήματα, είναι πάρα πολύ επικίνδυνο να εκτεθούν πολλά είδη υπηρεσιών. Είναι ακόμα δύσκολο να δημιουργηθούν ασφαλείς διανεμημένες εφαρμογές . Ιδανικά, πρέπει να υπάρξουν απλοί τρόποι να εξεταστούν τα κοινά σενάρια ασφάλειας, μαζί με fine-grained έλεγχο για τις εφαρμογές που το χρειάζονται.

Για να επιτευχθεί αυτό, το WCF παρέχει τις λειτουργίες ασφάλειας πυρήνων, της επικύρωσης, της ακεραιότητας μηνυμάτων, της εμπιστευτικότητας μηνυμάτων, και της έγκρισης. Όλοι αυτοί εξαρτώνται πλήρως από την έννοια της ταυτότητας: ποιος είναι αυτός ο χρήστης; Η καθιέρωση μιας ταυτότητας για έναν πελάτη ή μια υπηρεσία απαιτεί πληροφορίες όπως ένα όνομα χρήστη και ένας κωδικός πρόσβασης, ένα πιστοποιητικό X.509, ή κάτι άλλο. Ένας πελάτης ή μια υπηρεσία μπορεί να κάνει αυτό άμεσα με το να επικαλεσθεί μια λειτουργία WCF, ή με τη χρησιμοποίηση ενός αρχείου config, ή με άλλους τρόπους, όπως με την παραπομπή σε ένα κατάστημα πιστοποιητικών. Η καθιέρωση μιας ταυτότητας είναι ένα ουσιαστικό μέρος της χρήσης της ασφάλειας WCF.

Η διανεμημένη ασφάλεια μπορεί να γίνει πολύ περίπλοκη πολύ γρήγορα. Ένας αρχικός στόχος των σχεδιαστών του WCF ήταν να καταστήσουν τις ασφαλείς εφαρμογές ευκολότερες. Για να επιτραπεί αυτό, η προσέγγιση του WCF στην επικύρωση, την ακεραιότητα στοιχείων και την ιδιωτικότητα στοιχείων στηρίζονται στις συνδέσεις. Αντί να απαιτηθεί οι υπεύθυνοι για την ανάπτυξη να παρεμβάλουν τις σωστές ιδιότητες για να εξασφαλίσουν κάθε κατηγορία και μέθοδο, μπορούν αντ' αυτού να χρησιμοποιήσουν ένα binding που παρέχει το σωστό σύνολο υπηρεσιών ασφάλειας. Οι επιλογές ενός υπεύθυνου για την ανάπτυξη μπορούν να είναι οι εξής:

1 Η χρήση ενός τυποποιημένου binding που υποστηρίζει άμεσα την ασφάλεια. Για παραδειγμα, οι εφαρμογές που απαιτούν την end to end ασφάλεια για τα μηνύματα που περνούν από τους πολλαπλούς μεσάζοντες SOAP μπορούν να χρησιμοποιήσουν μια σύνδεση που υποστηρίζει την WS-ασφάλειας, όπως WsHttpBinding.

2 Η χρήση ενός τυποποιημένου binding που υποστηρίζει προαιρετικά την ασφάλεια, και κατόπιν διαμορφώνεται όπως απαιτείται. Παραδείγματος χάριν, οι εφαρμογές που χρειάζονται μόνο την βασισμένη στη μεταφορά ασφάλεια μπορούν να επιλέξουν BasicHttpBinding, διαμορφώνοντας το για να χρησιμοποιήσουν το HTTPS αντί του HTTP. Είναι επίσης δυνατό να προσαρμοστούν άλλες περισσότερο προηγμένες συμπεριφορές ασφάλειας. Για παράδειγμα, τον μηχανισμό επικύρωσης που χρησιμοποιείται από ένα binding όπως το WsHttpBinding μπορεί να αλλάξει εάν είναι επιθυμητό.

3 Η δημιουργία ενός binding που παρέχει ακριβώς τη συμπεριφορά ασφάλειας που ένας υπεύθυνος για την ανάπτυξη χρειάζεται. αυτός ο τρόπος δεν είναι πανάκεια , αλλά μπορεί να είναι η σωστή λύση για μερικά προβλήματα.

4 Η χρήση ενός τυποποιημένου binding που δεν παρέχει καμία υποστήριξη για την ασφάλεια, όπως το BasicHttpBinding. Χωρίς τη χρήση καμίας ασφάλειας είναι συχνά επικίνδυνο πράγμα, αλλά μπορεί μερικές φορές να είναι η μόνη επιλογή.

Για παράδειγμα, υποθέστε ότι η υπηρεσία RentalReservations επιθυμούν να εκθέσει ένα endpoint χρησιμοποίησε το BasicHttpBinding με HTTPS αντί για το HTTP. Το αρχείο διαμόρφωσης που καθόρισε αυτό το endpoint θα έμοιαζε με αυτό:

```
<configuration>
 <system.serviceModel>
  <services>
   <service
    type="RentalReservations,RentalApp"
    <endpoint
     contract="IReservations"
     binding="basicHttpBinding"
     bindingConfiguration="UsingHttps"
     address=
      "http://www.fabrikam.com/reservation/reserve.svc"/>
   </service>
  </services>
  <bindings>
        <basicHttpBinding>
    <binding
     configurationName="UsingHttps"
     securityMode="Https"/>
   </basicHttpBinding>
  </bindings>
 </system.serviceModel>
</configuration>
```

Αντίθετα από το αρχείο διαμόρφωσης που παρουσιάζεται νωρίτερα, που χρησιμοποιείται μόνο το τυποποιημένο basicHttpBinding, αυτή η έκδοση περιλαμβάνει τις ιδιότητες του bindingConfiguration στο στοιχείο endpoint. Η διαμόρφωση στις αναφορές αυτής της ιδιότητας, λαμβάνοντας υπόψη το όνομα UsingHttps, καθορίζεται αργότερα στο αρχείο config μέσα στο στοιχείο του binding. Αυτή η διαμόρφωση θέτει την ιδιότητα του basicHttpBinding, securityMode σε Https, αναγκάζοντας όλη την επικοινωνία με αυτό το endpoint να χρησιμοποιήσει HTTPS αντί του τυποποιημένου HTTP.

Μια υπηρεσία WCF μπορεί επίσης να ελέγξει ποιοι πελάτες εξουσιοδοτούνται για να χρησιμοποιήσουν τις υπηρεσίες της. Για να γίνει αυτό, το WCF στηρίζεται στους υπάρχοντες μηχανισμούς έγκρισης του .NET Framework. Μια υπηρεσία μπορεί να χρησιμοποιήσει τις τυποποιημένες ιδιότητες του PrincipalPermission, παραδείγματος χάριν, για να καθορίσει ποιος έχει την άδεια για να έχει πρόσβαση σε αυτό. Εναλλακτικά, μια εφαρμογή WCF που χρειάστηκε την βασισμένη στο ρόλο έγκριση θα μπορούσε να χρησιμοποιήσει το Windows Authorization Manager για να το εφαρμόσει αυτό.

Η βοήθεια των υπεύθυνων για την ανάπτυξη για να χτιστούν ασφαλέστερες εφαρμογές χωρίς την έκθεση τους σε συντριπτική πολυπλοκότητα είναι ένα τεράστιο πρόβλημα. Με την παροχή μιας απλής προσέγγισης για τις πιό κοινές περιπτώσεις και του fine-grained ελέγχου για τις πιό σύνθετες καταστάσεις, το WCF στοχεύει στην επίτευξη αυτού του στόχου με έναν χρησιμοποιήσιμο και αποτελεσματικό τρόπο.

3.9.4 Συναλλαγές

Ο χειρισμός των συναλλαγών είναι μια σημαντική πτυχή της δημιουργίας πολλών ειδών επιχειρησιακής λογικής. Ακόμα η χρήση των συναλλαγών σε έναν προσανατολισμένο προς τις υπηρεσίες κόσμο μπορεί να είναι προβληματική. Οι διανεμημένες συναλλαγές υποθέτουν ένα υψηλό επίπεδο εμπιστοσύνης μεταξύ των συμμετεχόντων, έτσι δεν είναι συχνά αρμόζον για μια συναλλαγή να επεκταθεί πάνω από ένα όριο υπηρεσιών. Ακόμα, δεδομένου ότι υπάρχουν καταστάσεις όπου ο συνδυασμός των συναλλαγών και των υπηρεσιών έχει νόημα, το WCF περιλαμβάνει υποστήριξη για αυτήν την σημαντική πτυχή του σχεδίου εφαρμογής.

Συναλλαγές στο *.NET Framework*: Συναλλαγές συστημάτων <>

Η υποστήριξη της συναλλαγής στο WCF στηρίζεται στις System.Transactions, ένα namespace του .NET Framework που επικεντρώνεται στον έλεγχο των συμπεριφορών των συναλλαγών. Αν και δεν απαιτείται, οι υπεύθυνοι για την ανάπτυξη χρησιμοποιούν συνήθως τις υπηρεσίες των System.Transactions σε συντονισμό με ένα *πλαίσιο εκτέλεσης*. Ένα πλαίσιο εκτέλεσης επιτρέπει την προδιαγραφή των κοινών πληροφοριών, όπως μια συναλλαγή, η οποία ισχύει για όλο τον κώδικα να περιλαμβάνεται σε ένα καθορισμένο πλαίσιο. Εδώ παραθέτεται ένα παράδειγμα για το πώς μια εφαρμογή μπορεί να χρησιμοποιήσει αυτήν την προσέγγιση για να ομαδοποιήσει ένα σύνολο διαδικασιών σε μια ενιαία συναλλαγή:

```
using System.Transactions;

using (TransactionScope ts =
  new TransactionScope(TransactionScopeOption.Required)) {
  // Do work, e.g., update different DBMSs
  ts.Complete();
}
```

Όλες οι διαδικασίες μέσα στο using φραγμό θα γίνουν μέρος μιας ενιαίας συναλλαγής, δεδομένου ότι μοιράζονται το πλαίσιο εκτέλεσης των συναλλαγών που καθορίζεται. Η τελευταία γραμμή σε αυτό το παράδειγμα, που καλεί την Complete μέθοδο του TransactionScope, θα οδηγήσει σε ένα αίτημα που θα δεσμεύσει τη συναλλαγή όταν ο φραγμός θα βγει. Αυτή η προσέγγιση παρέχει επίσης τον ενσωματωμένο χειρισμό λάθους, αποβάλλοντας τη συναλλαγή εάν μια εξαίρεση εμφανιστεί.

Ορίζοντας το TransactionScopeOption.Required για το νέο TransactionScope, όπως αυτό το παράδειγμα, σημαίνει ότι αυτός ο κώδικας θα τρέξει πάντα ως τμήμα μιας συναλλαγής: προσαρτεί τη συναλλαγή του επισκέπτη εάν υπάρχει, αλλιώς δημιουργεί μια νέα. Σαν τις επιχειρηματικές υπηρεσίες, άλλες επιλογές όπως οι RequiresNew, μπορούν επίσης να οριστούν.

Αντίθετα από τις επιχειρηματικές υπηρεσίες και των προκατόχων Microsoft Transaction Server (MTS) και οι COM +, οι συναλλαγές συστημάτων στρέφονται εξ ολοκλήρου στον έλεγχο της συμπεριφοράς των συναλλαγών. Για παράδειγμα δεν υπάρχει καμία απαραίτητη σύνδεση μεταξύ μιας συναλλαγής και της εσωτερικής κατάστασης ενός αντικειμένου. Ενώ οι επιχειρηματικές υπηρεσίες απαιτούν ένα αντικείμενο για να απενεργοποιηθούν όταν τελειώνει μια συναλλαγή, οι συναλλαγές συστημάτων δεν προβάλλουν καμία τέτοια απαίτηση. Δεδομένου ότι το WCF στηρίζεται στη συναλλαγή συστημάτων, οι βασισμένες σε WCF εφαρμογές είναι επίσης ελεύθερες να διαχειριστούν τις συναλλαγές και την κατάσταση του αντικειμένου ανεξάρτητα.

Οι βασισμένες σε WCF εφαρμογές μπορούν να χρησιμοποιήσουν τους τύπους στις System.Transactions άμεσα, ή μπορούν να ελέγξουν τις συναλλαγές χρησιμοποιώντας τις WCF-καθορισμένες ιδιότητες που στηρίζονται στις System.Transactions. Στην πρώτη επιλογή, μια μέθοδος που χαρακτηρίστηκε με τις ιδιότητες της OperationContract να οργανώσει την εργασία της σε μια συναλλαγή χρησιμοποιώντας το TransactionScope, όπως ακριβώς περιγράφεται. Για παράδειγμα αυτή η μέθοδος θα περιλάβει μια using δήλωση που καθιερώνει ένα πεδίο συναλλαγής, και έπειτα ενημερώνει δύο ανεξάρτητες βάσεις δεδομένων μέσα σε εκείνη την συναλλαγή.

Εναλλακτικά, οι μέθοδοι των υπηρεσιών μπορούν να ελέγξουν τη συμπεριφορά των συναλλαγών μέσω μιας ιδιότητας. Παρά την ρητή εξάρτηση στις System.Transactions, μια υπηρεσία μπορεί να χρησιμοποιήσει τις ιδιότητες της OperationBehavior που περιγράφονται νωρίτερα. Υποθέστε, για παράδειγμα, ότι η εργασία που έγινε με τη μέθοδο Reserve της κλάσης RentalReservations ήταν πάντα συναλλαγών. Αυτό είναι βεβαίως εύλογο, αφού δημιουργώντας μια νέα reservation είναι πιθανόν να απαιτούνται περισσότερα από ένα συστήματα βάσεων δεδομένων. Σε αυτήν την περίπτωση, η reservation θα μπορούσε να καθοριστεί όπως αυτό:

```
[OperationContract]
[OperationBehavior(TransactionScopeRequired=true,
        TransactionAutoComplete=true)]
private int Reserve(int vehicleClass, int location, string dates)
{
 int confirmationNumber;
 // logic to reserve rental car goes here
 return confirmationNumber;
}
```

Σε αυτό το παράδειγμα, η Reserve προηγείται τώρα από τις ιδιότητες της OperationBehavior με την TransactionScopeRequired καθορισμένη ως true. Λόγω αυτού, όλη η εργασία που γίνεται μέσα σε αυτήν την μέθοδο θα συμβεί μέσα σε μια συναλλαγή, ακριβώς σαν ήταν μέσα στο πεδίο συναλλαγής του using φραγμού που παρουσιάστηκε νωρίτερα. Και επειδή η TransactionAutoComplete είναι ορισμένη επίσης true, η μέθοδος αυτόματα θα δεσμεύσει τη συναλλαγή εάν καμία εξαίρεση δεν εμφανιστεί.

Εάν ο πελάτης που επικαλείται αυτήν την μέθοδο δεν τρέχει μέσα σε μια συναλλαγή, η μέθοδος Reserve θα τρέξει στη δική της συναλλαγή-δεν έχει καμία άλλη επιλογή. Αλλά υποθέστε ότι ο πελάτης είναι ήδη μέρος μιας υπάρχουσας συναλλαγής όταν καλεί την Reserve. Η εργασία που γίνεται με τη μέθοδο Reserve θα ενώσει τη συναλλαγή του πελάτη, ή θα τρέξει ακόμα μέσα στην ανεξάρτητη συναλλαγή της; Η απάντηση εξαρτάται από εάν αυτή η υπηρεσία μπορεί να δεχτεί ένα *πλαίσιο συναλλαγής* σταλμένο από τον πελάτη, μια επιλογή ελεγχόμενη χρησιμοποιώντας bindings. Μερικά bindings, όπως το WsHttpBinding και το NetTcpBinding, μπορούν να διαμορφωθούν για να περιλάβουν ένα στοιχείο transactionFlow που θα καθορίζει εάν μια υπηρεσία μπορεί να δεχτεί ένα πλαίσιο συναλλαγής από τον πελάτη. Εάν το binding που χρησιμοποιείται από την υπηρεσία είναι σαν αυτό του παραδείγματος και ο πελάτης έχει ένα πλαίσιο συναλλαγής, η εργασία που γίνεται από την Reserve θα γίνει μέρος της συναλλαγής του πελάτη. Αν όχι, η εργασία αυτής της μεθόδου θα παραμείνει στη συναλλαγή της μεθόδου.

Για να φανεί πώς αυτό να χρησιμοποιηθεί, σκεφτείτε άλλη μια φορά για την εφαρμογή που χρησιμοποιήσαμε πριν ως παράδειγμα. Το endpoint που θα έχει πρόσβαση η κλήση της εφαρμογής κεντρικών πελατών θα διαμορφώνονταν πιθανώς για να δεχτούν ένα πλαίσιο συναλλαγής που στέλνεται από τον πελάτη, δεδομένου ότι αυτή η εφαρμογή είναι αρκετά αξιόπιστη για να συμπεριλαμβάνει τις μεθόδους του συστήματος επιφύλαξης σε μια συναλλαγή. Στην πραγματικότητα, δεδομένου ότι αυτός ο πελάτης δημιουργείται πιθανώς από την ίδια ομάδα που παρήγαγε την εφαρμογή reservation, είναι

ασφαλές να υποτεθεί ότι δεν θα κρατήσει μια συναλλαγή ανοικτή - και να κλειδώσει έτσι τα στοιχεία που η συναλλαγή χρησιμοποιεί - για υπερβολικά μεγάλο χρονικό διάστημα. Ομοίως, η βασισμένη στην java EE υπάρχουσα εφαρμογή reservation θα έχει πρόσβαση στο σύστημα reservation χρησιμοποιώντας ένα endpoint του οποίου η σύνδεση θα έχει διαμορφωθεί για να δεχτεί ένα πλαίσιο συναλλαγής που στάλθηκε από τον πελάτη. Αυτή η εφαρμογή είναι αξιόπιστη επίσης για να περιλάβει τις μεθόδους του συστήματος reservation στις συναλλαγές, δεδομένου ότι ελέγχεται από την ίδια επιχείρηση. Αλλά όλα τα endpoints που προσεγγίζονται από τις εφαρμογές συνεργατών είναι πιθανό να έχουν bindings που δεν θα δεχτούν ότι ένα πλαίσιο συναλλαγής στάλθηκε από τον πελάτη. Δεδομένου ότι αυτές οι εφαρμογές συνεργατών κατέχονται από άλλες οργανώσεις, είναι απίθανο να είναι αρκετά αξιόπιστες για να συμπεριλαμβάνουν τις μεθόδους της εφαρμογής reservation μέσα στις συναλλαγές.

Τέλος, αξίζει να σημειωθεί ότι οι εφαρμογές που στηρίζονται σε WCF μπορούν να συμμετέχουν στις συναλλαγές που περιλαμβάνουν εφαρμογές που τρέχουν σε μη wcf πλατφόρμες. Για παράδειγμα, μια βασισμένη σε WCF εφαρμογή μπορεί να αρχίσει μια συναλλαγή, να ενημερώσει ένα αρχείο σε μια τοπική βάση δεδομένων κεντρικών υπολογιστών SQL, κατόπιν να επικαλεσθεί μια υπηρεσία Ιστού που εφαρμόστηκε σε μια βασισμένη σε java EE εφαρμογή που ενημερώνει ένα αρχείο σε μια άλλη βάση δεδομένων. Εάν αυτή η υπηρεσία είναι συναλλαγών και η πλατφόρμα που τρέχει υποστηρίζει την προδιαγραφή WS- AtomicTransaction, και οι δύο αναπροσαρμογές βάσεων δεδομένων μπορούν να είναι μέρος της ίδιας συναλλαγής. Όπως με την ασφάλεια και με το αξιόπιστο μήνυμα, ετσι και οι WCF συναλλαγές μπορούν να λειτουργήσουν στον ετερογενή κόσμο που οι υπηρεσίες Ιστού καθιστούν πιθανό.

Κεφάλαιο 4: Ανάλυση πρακτικού μέρους

4.1 Εισαγωγή

Αναφερθήκαμε είδη στον ορισμό των υπηρεσιών ιστού, τον τρόπο λειτουργίας τους, τα πρωτόκολλα που χρησιμοποιούν καθώς και το WCF framework. Συγκεκριμένα στο κεφάλαιο 2 αναλύθηκαν τα πρωτόκολλα SOAP, REST, UDDI, WSDL , WADL ενώ στο κεφάλαιο 3 παρουσιάστηκε εκτενέστερα το WCF framework καθώς και οι υπηρεσιοστραφείς εφαρμογές που μπορούν να δημιουργηθούν μέσω αυτού. Επίσης διαχωρίστηκε η έννοια μίας WCF service και ενός WCF client application, οι οποίοι επικοινωνούν μέσω μιας proxy class και ενός endpoint.

Όσον αφορά το κεφάλαιο 4, σκοπός του είναι να συνδέσει τη θεωρεία που αναπτύχθηκε στα προηγούμενα κεφάλαια με μία πρακτική εφαρμογή η οποία απαρτίζεται από έναν client application και μία wcf service. Αναλυτικότερα, θα αναφερθούν τα πρωτόκολλα που χρησιμοποιεί η εφαρμογή όπως επίσης και παραδείγματα χρήσης τους. Στη συνέχεια θα γίνει συσχετισμός της WCF λογικής και της εφαρμογής καθώς και αντιστοίχιση των θεωρητικών εννοιών υπηρεσίας και πελάτη με τα αυτά της πρακτικής εφαρμογής.

Τέλος θα εξεταστεί η υπηρεσία με βάση τις λειτουργίες της, παραθέτοντάς τες καθώς και τα κριτήρια λειτουργίας τους.

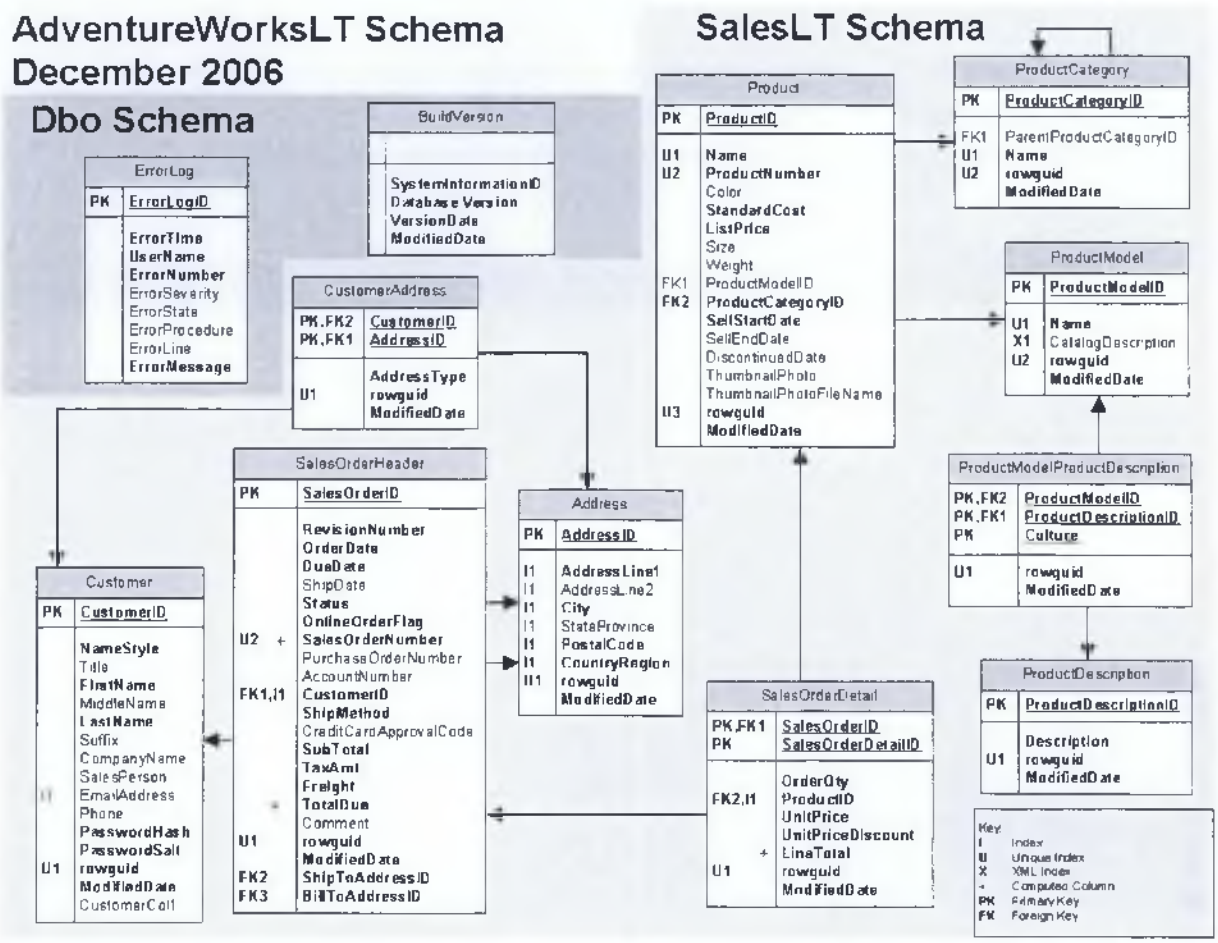
4.2 Η Βάση Δεδομένων AdventureWorksLT

Στο πρακτικό μέρος, που θα επεξηγήσουμε τη λειτουργία του στη συνέχεια του κεφαλαίου, υλοποιήσαμε την εφαρμογή σε δύο μέρη την ProductsService και την ProductsClient. Η ProductsClient είναι το μέσο διεπαφής με τον χρήστη και χρησιμοποιεί την proxy class για να στείλει αιτήματα και να επικοινωνήσει με την ProductsService. Η ProductsService χρησιμοποιεί το connection string για να τραβήξει δεδομένα από την βάση AdventureWorksLT. Αυτή η βάση μας παρέχει έτοιμα δεδομένα για να τα αξιοποιήσουμε στην εφαρμογή μας. Σκοπός της είναι να παρέχει ένα σχετικά απλό σχήμα βάσης με δεδομένα που να υποστηρίζουν ένα σενάριο πωλήσεων προϊόντων. Μπορείτε να την βρείτε στο
http://msftdbprodsamples.codeplex.com/wikipage?title=AWLTDocs

Η AdventureWorksLT αναλύεται στο σχήμα 4.4.1 που ακολουθεί. Η εφαρμογή μας χρησιμοποιεί τους εξής πίνακες: τον Customer, ο οποίος περιέχει στοιχεία για τους πελάτες, τον SalesOrderHeader και SalesOrderDetail, σκοπός των οποίων είναι η κράτηση στοιχείων για τις παραγγελίες και τις λεπτομέρειες τους και τον Product, ο οποίος κρατά πληροφορίες σχετικά με τις λεπτομέρειες των προϊόντων.

Πιο συγκεκριμένα ο πίνακας Customer περιλαμβάνει στοιχεία για τον πελάτη όπως τον κωδικό του, το πλήρες ονοματεπώνυμό του και διάφορα στοιχεία επικοινωνίας ,για παράδειγμα τον αριθμό του τηλεφώνου του και το e-mail του. Ο πίνακας Product όπως είναι αναμενόμενο περιέχει πληροφορίες όπως τον κωδικό του προϊόντος ,το όνομά του, το κόστος του, το χρώμα του και άλλες λεπτομέρειες περιγραφής. Ο Sales Order Header παρέχει δεδομένα για τις παραγγελίες που αφορούν τον κωδικό της και το όνομα της, τον κωδικό του πελάτη, την ημέρα που έγινε η παραγγελία. Περιλαμβάνει ακόμα και τον τρόπο που θα μεταφερθεί το προϊόν. Τέλος ο πίνακας

Sales Order Detail συμπληρώνει τον προηγούμενο παρέχοντας τα εξής στοιχειά: τον κωδικό του προϊόντος και αλλους κωδικούς που διευκολίνουν την αναζήτηση δεδομένων από τους πίνακες.



Σχήμα 4.2. 1 AdventureWorksLT

4.3 Τα πρωτόκολλα της υπηρεσίας

Η εφαρμογή χρησιμοποιεί το πρωτόκολλο SOAP. Δεδομένου ότι μιλάμε για μια υπηρεσία ιστού η χρήση του συγκεκριμένου προσφέρει την απλότητα ως πλεονέκτημα. Αυτό συμβαίνει διότι το SOAP αποτελείται απλώς από XML και HTTP που συνδυάζονται για μηνύματα διαδικτύου. Μιας και τα αιτήματα HTTP επιτρέπονται μέσω των firewalls το πρόγραμμα δεν θα αντιμετωπίσει πρόβλημα από τα firewalls των servers που διώχνουν εκτός τα αιτήματα από άγνωστες εφαρμογές. Επιπρόσθετα η χρήση του SOAP δεν απαιτεί συγκεκριμένη γλώσσα προγραμματισμού η πλατφόρμα οπότε καθίσταται πιο ευπροσάρμοστο.

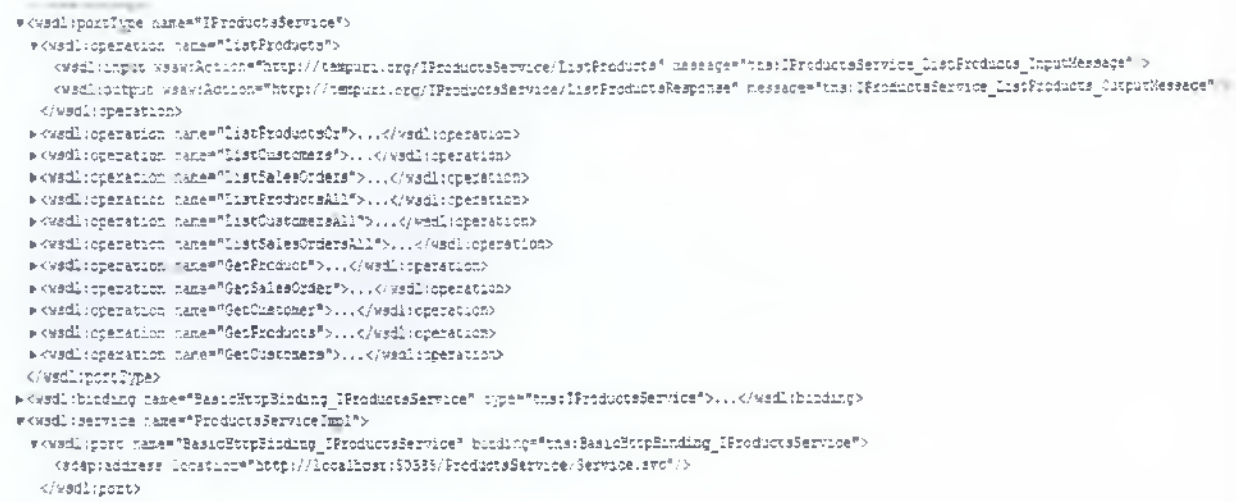
Ακόμα η εφαρμογή χρησιμοποιεί WSDL το οποίο είναι αναμενόμενο μιας και ένα αρχείο WSDL είναι ένα XML έγγραφο που περιγράφει μια σειρά μηνυμάτων SOAP αλλά και πως γίνεται η ανταλλαγή τους. Για να βρούμε το WSDL αρχείο της εφαρμογής μας πρέπει να ακολουθήσουμε την περαιτέρω διαδικασία:

1. Στο solution explorer στο C:\....\ProductsService\Website πατάμε δεξί κλικ στο αρχείο service.svc και μετά επιλέγουμε το view in browser. Ο ASP.NET Development Server ξεκινάει και ο internet explorer εμφανίζει μια σελίδα όπως η ακόλουθη (Σχήμα 4.3.1).

Σχήμα 4.3. 1 Βοηθητική σελίδα

Αυτή είναι μια βοηθητική σελίδα για την WCF υπηρεσία. Πιστοποιεί ότι η υπηρεσία έχει ρυθμιστεί σωστά. Σε περίπτωση προβλημάτων εμφανίζονται μηνύματα λάθους. Στη συγκεκριμένη σελίδα παρέχονται και πληροφορίες για τη δημιουργία client που να μπορεί να συνδέεται με την υπηρεσία.

2. Πατάμε κλικ στο URLπου εμφανίζεται στον internet explorer. Μια ακόμα σελίδα εμφανίζεται (Σχήμα 4.3.2):



Σχήμα 4.3. 2 Κώδικας WSDL που ορίζει τις λειτουργίες της υπηρεσίας.

Αυτή η σελίδα εμφανίζει τα μεταδεδομένα που περιγράφουν την υπηρεσία. Ουσιαστικά είναι ένα XML document που χρησιμοποιεί WSDL schema. Τα στοιχεία σε αυτό το document περιγράφουν τις λειτουργίες που η WCF υπηρεσία παρέχει , και τον τύπο των μηνυμάτων που μπορεί να πάρει η να δώσει. Οι λειτουργίες παράχθηκαν από τις μεθόδους που υλοποιήσαμε και τα μηνύματα βασίζονται στις παραμέτρους που κάθε μέθοδος παίρνει και στις τιμές που επιστρέφει μια μέθοδος.

Ας γίνουμε όμως πιο συγκεκριμένοι. Στο συγκεκριμένο σημείο του κώδικα ορίζονται οι λειτουργίες της υπηρεσίας. Η λίστα με τις λειτουργίες είναι η ακόλουθη:

- ListProducts-Εμφανίζει μία λίστα προϊόντων χρησιμοποιώντας τον κωδικό ενός πελάτη.
- ListCustomers- Εμφανίζει μία λίστα πελατών χρησιμοποιώντας τον κωδικό ενός προϊόντος.
- ListSalesOrders- Εμφανίζει μία λίστα παραγγελιών χρησιμοποιώντας τον κωδικό του πελάτη.
- ListProductsAll- Εμφανίζει μία λίστα με όλα τα προϊόντα.
- ListCustomersAll- Εμφανίζει μία λίστα με όλους τους πελάτες.
- ListSalesOrdersAll- Εμφανίζει μία λίστα με όλες τις παραγγελίες.
- GetProduct- Βρίσκουμε τα στοιχεία ενός προϊόντος χρησιμοποιώντας τον κωδικό του.
- GetSalesOrder- Βρίσκουμε τα στοιχεία μίας παραγγελίας με βάση τον κωδικό της.
- GetCustomer-Βρίσκουμε τα στοιχεία ενός πελάτη με βάση τον κωδικό του.
- GetProducts- Βρίσκουμε τα στοιχεία ενός προϊόντος χρησιμοποιώντας το όνομά του.
- GetCustomers- Βρίσκουμε τα στοιχεία ενός πελάτη με βάση το επώνυμό του.

Χρησιμοποιώντας σαν παράδειγμα τη λειτουργία ListProducts θα εξηγήσουμε πως λειτουργεί το WSDL σχήμα. Βλέπουμε τη ListProducts να ορίζεται μέσα στο <wsdl:operation name="ListProducts">. Η επιγραφή wsaw είναι soap εμφανίζεται σε φάκελο που καταγράφηκε από το WCF. Το ουσιαστικό όμως σημείο του WSDL είναι η δήλωση των input και output μηνυμάτων. Τα στοιχεία αυτά διευκρινίζουν τη μορφή των abstract για το αίτημα και την απάντηση αντίστοιχα.

Υπάρχει μια ετικέτα message="tns:IProductsService_ListProducts_InputMessage" η οποία χρησιμοποιεί ένα namespace που έχει γίνει import σε προηγούμενο σημείο του κώδικα (Σχήμα 4.3.3).

```
▼<wsdl:types>
  ▼<xsd:schema targetNamespace="http://tempuri.org/Imports">
     <xsd:import schemaLocation="http://localhost:50385/ProductsService/Service.svc?xsd=xsd0" namespace="http://tempuri.org/"/>
     <xsd:import schemaLocation="http://localhost:50385/ProductsService/Service.svc?xsd=xsd1" namespace="http://schemas.microsoft.com/2003/10/Serialization.
     <xsd:import schemaLocation="http://localhost:50385/ProductsService/Service.svc?xsd=xsd2" namespace="http://schemas.datacontract.org/2004/07/Products"/
   </xsd:schema>
 </wsdl:types>
```

Σχήμα 4.3. 3 Κώδικας WSDL που κάνει import το αρχείο με τους τύπους των λειτουργιών.

Πληκτρολογώντας τη διεύθυνση http://localhost:50385/ProductsService/Service.svc?xsd=xsd0 εμφανίζεται ένα ακόμα αρχείο που ορίζει τους τύπους. (Σχήμα 4.3.4)

```
▼<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://tempuri.org/" elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
   <xs:import schemaLocation="http://localhost:50385/ProductsService/Service.svc?xsd=xsd2" namespace="http://schemas.datacontract.org/2004/07/Products"/>
  ▼<xs:element name="ListProducts">
   ▼<xs:complexType>
    ▼<xs:sequence>
       <xs:element minOccurs="0" name="customerID" type="xs:int"/>
     </xs:sequence>
    </xs:complexType>
  </xs:element>
  ▼<xs:element name="ListProductsResponse">
   ▼<xs:complexType>
    ▼<xs:sequence>
       <xs:element xmlns:q1="http://schemas.datacontract.org/2004/07/Products" minOccurs="0" name="ListProductsResult" nillable="true" type="q1:ArrayOfProductData"/>
     </xs:sequence>
    </xs:complexType>
  </xs:element>
 ▶<xs:element name="ListProductsOr">...</xs:element>
 ▶<xs:element name="ListProductsOrResponse">...</xs:element>
 ▶<xs:element name="ListCustomers">...</xs:element>
 ▶<xs:element name="ListCustomersResponse">...</xs:element>
 ▶<xs:element name="ListSalesOrders">...</xs:element>
 ▼<xs:element name="ListSalesOrdersResponse">
  ▶<xs:complexType>...</xs:complexType>
  </xs:element>
 ▶<xs:element name="ListProductsAll">...</xs:element>
 ▶<xs:element name="ListProductsAllResponse">...</xs:element>
 ▶<xs:element name="ListCustomersAll">...</xs:element>
 ▶<xs:element name="ListCustomersAllResponse">...</xs:element>
 ▶<xs:element name="ListSalesOrdersAll">...</xs:element>
 ▶<xs:element name="ListSalesOrdersAllResponse">...</xs:element>
 ▶<xs:element name="GetProduct">...</xs:element>
 ▶<xs:element name="GetProductResponse">...</xs:element>
 ▶<xs:element name="GetSalesOrder">...</xs:element>
 ▶<xs:element name="GetSalesOrderResponse">...</xs:element>
 ▶<xs:element name="GetCustomer">...</xs:element>
 ▶<xs:element name="GetCustomerResponse">...</xs:element>
 ▶<xs:element name="GetProducts">...</xs:element>
 ▶<xs:element name="GetProductsResponse">...</xs:element>
 ▶<xs:element name="GetCustomers">...</xs:element>
 ▶<xs:element name="GetCustomersResponse">...</xs:element>
</xs:schema>
```

Σχήμα 4.3. 4 Κώδικας WSDL που ορίζει τους τύπους των μεταβλητών.

Από την δήλωση των τύπων προκύπτει ότι η ListProducts έχει ως είσοδο int(customer ID) και ως έξοδο ArrayOfProductData(ListProductResults). Συμπεραίνουμε λοιπόν ότι η λειτουργία ListProducts θα δέχεται αιτήματα με ακέραιες τιμές και θα στέλνει απαντήσεις που να περιέχουν λίστες με DataProducts.

Η ίδια εφαρμογή θα μπορούσε να έχει βασιστεί στο πρωτόκολλο REST μόνο που θα χρησιμοποιούσε WADL αντί για WSDL.

4.4 Η εφαρμογή της WCF λογικής στο πρακτικό μέρος.

Είναι γνωστό πλέον ότι μια WCF εφαρμογή απαρτίζεται από δύο μέρη:
Την WCF υπηρεσία και τον WCF client application. Στην περίπτωση της δικής μας εφαρμογής η υπηρεσία μας ονομάζεται ProductsService ενώ η client application ProductsClient. Στο σχήμα 4.3.1 που ακολουθεί φαίνεται η διάκριση μεταξύ Data contracts, Operation contracts και Service contracts που βρίσκονται στο interface της υπηρεσίας. Το interface της υπηρεσίας μας ονομάζεται IProductsService και είναι υπεύθυνο για τη δόμηση της. Από τα contracts του IProductsService παράχθηκε το WSDL που μελετήσαμε πιο πάνω.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;

namespace Products
{
    [DataContract]
    public class ProductData
    {
        [DataMember]
        public string Name;
        [DataMember]
        public string ProductNumber;
        [DataMember]
        public int ProductID;
        [DataMember]
        public string Color;
        [DataMember]
        public decimal ListPrice;
    }
    [DataContract]
    public class CustomerData
```

Σχήμα 4.4. 1 Τα Data contracts του IProductsService

```
    {
        [DataMember]
        public string FirstName;
        [DataMember]
        public string MiddleName;
        [DataMember]
        public string LastName;
        [DataMember]
        public string phone;
        [DataMember]
        public string EmailAddress;
    }
    [DataContract]
    public class SalesOrderData
    {
        [DataMember]
        public DateTime OrderDate;
        [DataMember]
        public DateTime DueDate;
        [DataMember]
        public string SalesOrderNumber;
        [DataMember]
        public int SalesOrderID;
        [DataMember]
        public int CustomerID;
    }
```

Σχήμα 4.4. 2 Τα Data contracts του IProductsService

```
[ServiceContract]
public interface IProductsService
{
    [OperationContract]
    List<ProductData> ListProducts (int customerID);
    [OperationContract]
    List<ProductData> ListProductsOr(int salesOrderID);
    [OperationContract]
    List<CustomerData> ListCustomers(int productID);
    [OperationContract]
    List<SalesOrderData> ListSalesOrders(int customerID);
    [OperationContract]
    List<ProductData> ListProductsAll();
    [OperationContract]
    List<CustomerData> ListCustomersAll();
    [OperationContract]
    List<SalesOrderData> ListSalesOrdersAll();
    [OperationContract]
    ProductData GetProduct(int productID);
    [OperationContract]
    SalesOrderData GetSalesOrder(int salesOrderID);
    [OperationContract]
    CustomerData GetCustomer(int customerID);
    [OperationContract]
    List<ProductData> GetProducts(string name);
```

Σχήμα 4.4. 3 Τα Service και Operation contracts του IProductsService

Ένα εξίσου σημαντικό αρχείο της υπηρεσίας μας είναι το ProductsService το οποίο περιέχει την υλοποίηση του interface. Για να είναι σωστή η υπηρεσία μας στο ProductsService πρεπει να έχουν υλοποιηθεί όλα τα contracts που ορίζονται μέσα στο IProductsService

Τέλος στο ProductsClient περιέχεται το Programs.cs. Εκεί βρίσκεται ο κώδικας που καλεί την proxy class η οποία είναι μια κλάση που παράγεται αυτόματα από το metadata endpoint το οποίο εμπεριέχεται στο Web.config της ProductsService και παρέχει την επικοινωνία μεταξύ WCF service και WCF client application.

4.5 Επεξήγηση πρακτικού μέρους

Η εφαρμογή μας αφορά μία εταιρία που πουλάει προϊόντα. Εξυπηρετεί τις ανάγκες αναζήτησης είτε αφορά πελάτες, είτε προϊόντα, είτε παραγγελίες. Υπάρχει η δυνατότητα εύρεσης στοιχείων μέσω πολλών και διαφορετικών κριτηρίων. Τρέχοντας την εφαρμογή στην οθόνη μας εμφανίζεται ένα μενού(βλ. Σχήμα 4.5.1) με τις επιλογές της Αναζήτησης(Search) και της Εξόδου(Exit).

Πατώντας την πρώτη επιλογή μεταφερόμαστε σ ένα υπομενού(βλ. Σχήμα 4.5.2) που μας δίνει τη δυνατότητα να επιλέξουμε αν θέλουμε να αναζητήσουμε στοιχεία προϊόντων(products), πελατών(customers) ή παραγγελιών(sales orders).

Στην επιλογή προϊόντα(products) είναι διαθέσιμες πέντε βασικές λειτουργίες(βλ. Σχήμα 4.5.3). Η επιλογή 1 δίνει στο χρήστη τη δυνατότητα να του εμφανίσει η εφαρμογή μια λίστα προϊόντων που έχει παραγγείλει ένας πελάτης κάνοντας αναζήτηση με τον κωδικό του(βλ. Σχήμα 4.5.4). Η επιλογή 2 αναζητά προϊόντα που τα έχουν παραγγείλει με την ίδια παραγγελία. Το πρόγραμμα Θα επιστρέψει μία λίστα προϊόντων όταν του δοθεί ως είσοδος ο κωδικός της παραγγελίας(βλ. Σχήμα 4.5.5). Η επιλογή 3 εμφανίζει μια λίστα με όλα τα προϊόντα(βλ. Σχήμα 4.5.6). Η επιλογή 4 μας επιτρέπει να αναζητήσουμε ένα προϊόν με βάσει τον κωδικό του(βλ. Σχήμα 4.5.7). Τέλος η 5 επιλογή μας επιστρέφει τα στοιχεία ενός προϊόντος πληκτρολογώντας το όνομα του(βλ. Σχήμα 4.5.8, Σχήμα 4.5.9).

Εάν επιλέξουμε την επιλογή αναζήτησης πελατών οι λειτουργίες που μπορούμε να εκτελέσουμε είναι οι ακόλουθες(βλ. Σχήμα 4.5.10). Η επιλογή 1 εμφανίζει μία λίστα πελατών οι

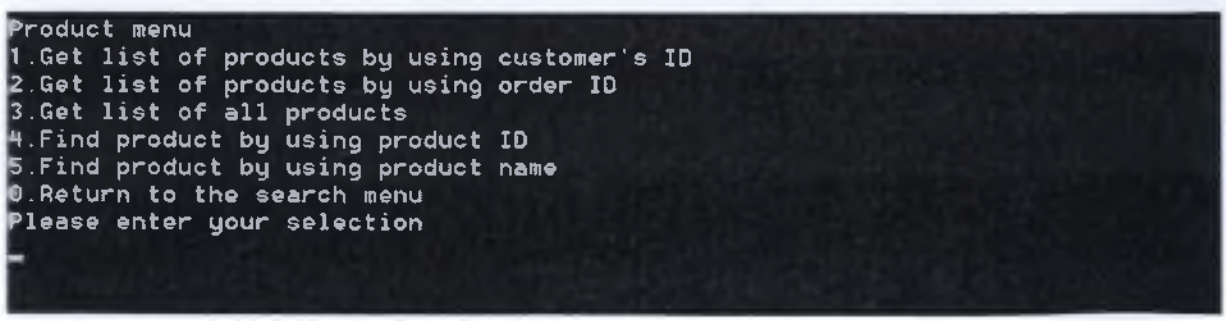
οποίοι έχουν παραγγείλει ένα προϊόν(βλ. Σχήμα 4.5.11). Η επιλογή νούμερο 2 εμφανίζει μια λίστα με όλους τους πελάτες(βλ. Σχήμα 4.5.12). Η επιλογή νούμερο 3 μας επιτρέπει να βρούμε τα στοιχεία ενός πελάτη χρησιμοποιώντας τον κωδικό του(βλ. Σχήμα 4.5.13). Η επιλογή νούμερο 4 μας δίνει τη δυνατότητα να αναζητήσουμε τα στοιχεία ενός πελάτη χρησιμοποιώντας το επίθετό του(βλ. Σχήμα 4.5.14).

Τέλος σε περίπτωση που επιλέξουμε την αναζήτηση παραγγελιών έχουμε τις επόμενες δυνατότητες(βλ. Σχήμα 4.5.15). Η επιλογή νούμερο 1 εμφανίζει μια λίστα παραγγελιών που έχει κάνει ένας πελάτης. Για να αναζητηθούν οι παραγγελίες πληκτρολογούμε τον κωδικό του πελάτη(βλ. Σχήμα 4.5.16). Η δεύτερη επιλογή μας εμφανίζει τη λίστα με όλες τις παραγγελίες(βλ. Σχήμα 4.5.17). Η τρίτη επιλογή εμφανίζει πληροφορίες για μία παραγγελία χρησιμοποιώντας τον κωδικό της(βλ. Σχήμα 4.5.18).

Επιπρόσθετα είναι επιτακτικό να αναφερθεί ότι έχουμε προβλέψει τυχών σφάλματα του χρήστη είτε πληκτρολογήσει τιμή που δεν ανήκει στο μενού είτε δεν υπάρχει κάποιο προϊόν πελάτης ή παραγγελία με τα αντίστοιχα χαρακτηριστικά που δίνει(βλ. Σχήμα 4.5.19, Σχήμα 4.5.20) .Τέλος ο χρήστης μπορεί να βγει από το αρχικό μενού πατώντας το 0 (μηδέν) για Exit και μπορεί πάλι πατώντας το 0(μηδέν) εάν βρίσκεται σε κάποιο από τα υπομενού αναζήτησης- προϊόντων, πελατών η παραγγελιών -να επιστρέψει στο γενικό μενού αναζήτησης .

Σχήμα 4.5. 1 Το αρχικό μενού της εφαρμογής

Σχήμα 4.5. 2 Το μενού αναζήτησης δίνει το δικαίωμα στο χρήστη να αναζητήσει προϊόντα, πελάτες και παραγγελίες.



Σχήμα 4.5. 3 Το μενού αναζήτησης προϊόντων και οι δυνατότητες του.

```
Please give customer's ID
29847
Number: FR-R72Y-38, Name: ML Road Frame-W - Yellow, 38
Number: FR-R72Y-48, Name: ML Road Frame-W - Yellow, 48
```

Σχήμα 4.5. 4 Επιλογή 1: Εμφάνιση λίστας προϊόντων του πελάτη με κωδικό 29847.

```
Please give order ID
71774
Number: FR-R72Y-48, Name: ML Road Frame-W - Yellow, 48
Number: FR-R72Y-38, Name: ML Road Frame-W - Yellow, 38
```

Σχήμα 4.5. 5 Επιλογή 2: Εμφάνισης λίστας προϊόντων με κωδικό παραγγελίας 71774.

```
List of all products
Number: FR-R92B-58, Name: HL Road Frame - Black, 58
Number: FR-R92R-58, Name: HL Road Frame - Red, 58
Number: HL-U509-R, Name: Sport-100 Helmet, Red
Number: HL-U509, Name: Sport-100 Helmet, Black
Number: SO-B909-M, Name: Mountain Bike Socks, M
Number: SO-B909-L, Name: Mountain Bike Socks, L
Number: HL-U509-B, Name: Sport-100 Helmet, Blue
Number: CA-1098, Name: AWC Logo Cap
Number: LJ-0192-S, Name: Long-Sleeve Logo Jersey, S
Number: LJ-0192-M, Name: Long-Sleeve Logo Jersey, M
Number: LJ-0192-L, Name: Long-Sleeve Logo Jersey, L
Number: LJ-0192-X, Name: Long-Sleeve Logo Jersey, XL
Number: FR-R92R-62, Name: HL Road Frame - Red, 62
Number: FR-R92R-44, Name: HL Road Frame - Red, 44
Number: FR-R92R-48, Name: HL Road Frame - Red, 48
Number: FR-R92R-52, Name: HL Road Frame - Red, 52
Number: FR-R92R-56, Name: HL Road Frame - Red, 56
Number: FR-R38B-58, Name: LL Road Frame - Black, 58
Number: FR-R38B-60, Name: LL Road Frame - Black, 60
Number: FR-R38B-62, Name: LL Road Frame - Black, 62
Number: FR-R38R-44, Name: LL Road Frame - Red, 44
Number: FR-R38R-48, Name: LL Road Frame - Red, 48
Number: FR-R38R-52, Name: LL Road Frame - Red, 52
Number: FR-R38R-58, Name: LL Road Frame - Red, 58
Number: FR-R38R-60, Name: LL Road Frame - Red, 60
Number: FR-R38R-62, Name: LL Road Frame - Red, 62
Number: FR-R72R-44, Name: ML Road Frame - Red, 44
Number: FR-R72R-48, Name: ML Road Frame - Red, 48
Number: FR-R72R-52, Name: ML Road Frame - Red, 52
Number: FR-R72R-58, Name: ML Road Frame - Red, 58
Number: FR-R72R-60, Name: ML Road Frame - Red, 60
Number: FR-R38B-44, Name: LL Road Frame - Black, 44
Number: FR-R38B-48, Name: LL Road Frame - Black, 48
Number: FR-R38B-52, Name: LL Road Frame - Black, 52
Number: FR-M94S-42, Name: HL Mountain Frame - Silver, 42
Number: FR-M94S-44, Name: HL Mountain Frame - Silver, 44
Number: FR-M94S-52, Name: HL Mountain Frame - Silver, 48
Number: FR-M94S-46, Name: HL Mountain Frame - Silver, 46
```

Σχήμα 4.5. 6 Επιλογή 3 εμφανίζει λίστα με όλα τα προϊόντα

```
Please give product's ID
680
Product ID:680
Product name:HL Road Frame - Black, 58
Product Number:FR-R92B-58
Color:Black
ListPrice:1431.5000
Press any key to continue . . .
```

Σχήμα 4.5. 7 Επιλογή 4: Εμφανίζει τα στοιχεία του προϊόντος με κωδικό 680

```
Please give product's name
long-sleeve logo jersey
Number: LJ-0192-L, Name: Long-Sleeve Logo Jersey, L
Number: LJ-0192-M, Name: Long-Sleeve Logo Jersey, M
Number: LJ-0192-S, Name: Long-Sleeve Logo Jersey, S
Number: LJ-0192-X, Name: Long-Sleeve Logo Jersey, XL
Press any key to continue . . .
```

Σχήμα 4.5. 8 Επιλογή 5: δίνουμε ακριβώς το όνομα του προϊόντος - Long-Sleeve Logo Jersey.

```
Please give product's name
jersey
Number: LJ-0192-L, Name: Long-Sleeve Logo Jersey, L
Number: LJ-0192-M, Name: Long-Sleeve Logo Jersey, M
Number: LJ-0192-S, Name: Long-Sleeve Logo Jersey, S
Number: LJ-0192-X, Name: Long-Sleeve Logo Jersey, XL
Number: SJ-0194-L, Name: Short-Sleeve Classic Jersey, L
Number: SJ-0194-M, Name: Short-Sleeve Classic Jersey, M
Number: SJ-0194-S, Name: Short-Sleeve Classic Jersey, S
Number: SJ-0194-X, Name: Short-Sleeve Classic Jersey, XL
Press any key to continue . . .
```

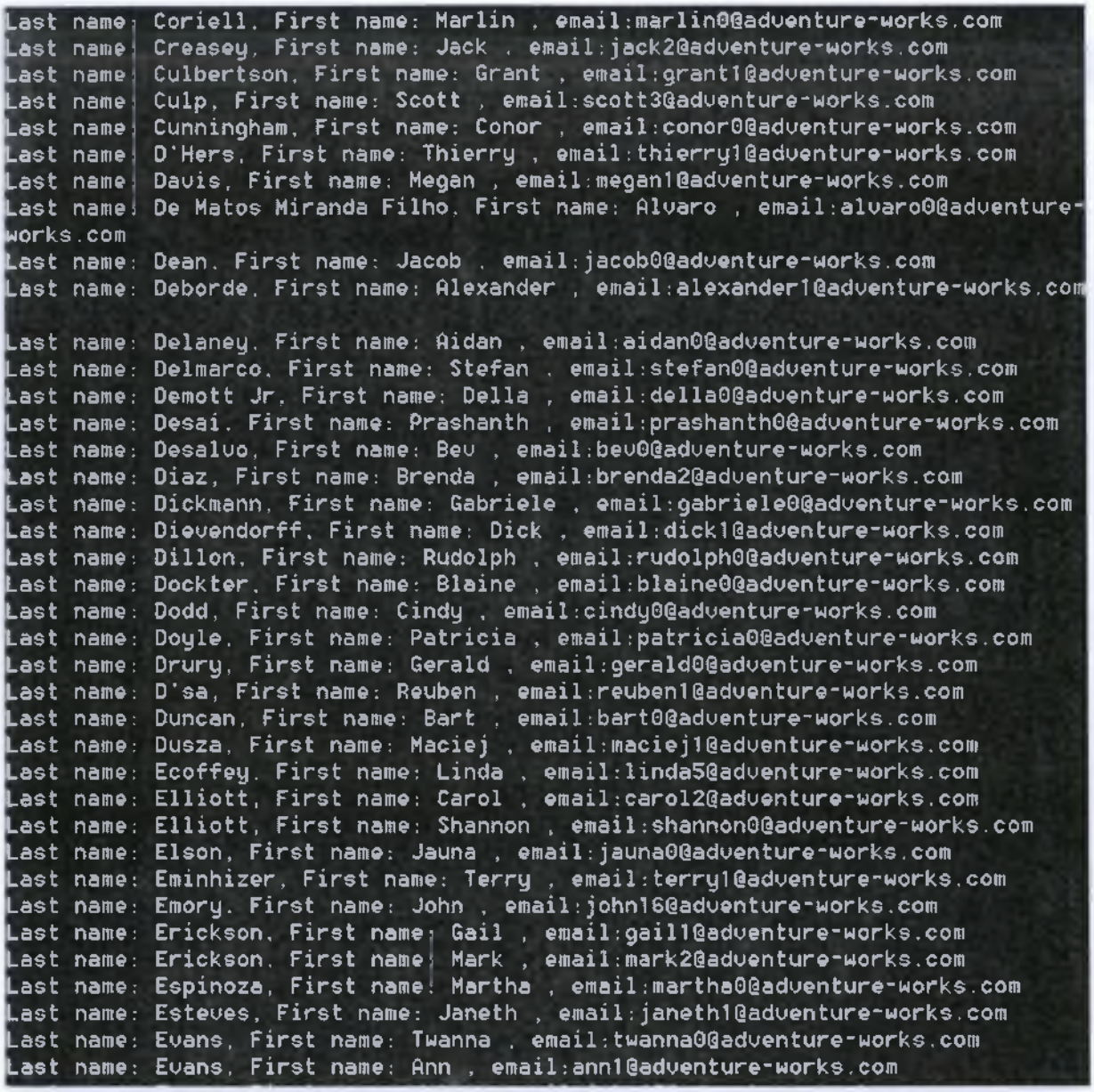
Σχήμα 4.5. 9 Επιλογή 5: Δεν θυμόμαστε ακριβώς το ονομα και πληκτρολογούμε jersey και εμφανίζει όλα τα προϊόντα που περιέχουν τη λέξη αυτή.

```
Customer menu
1.Get list of customers by using product ID
2.Get list of all customers
3.Find customer by using customer's ID
4.Find customer by using customer's last name
0.Return to the search menu
Please enter your selection
```

Σχήμα 4.5. 10 Το μενού αναζήτησης πελατών και οι δυνατότητές του.

```
Please give product ID
836
Last Name: Carroll, First Name: Rosmarie, E-mail: rosmarie0@adventure-works.com
Last Name: Grande, First Name: Jon, E-mail: jon1@adventure-works.com
Last Name: Hodgson, First Name: David, E-mail: david16@adventure-works.com
Last Name: Liu, First Name: Kevin, E-mail: kevin5@adventure-works.com
```

Σχήμα 4.5. 11 Επιλογή 1: Εμφανίζεται λίστα πελατών με κωδικό προϊόντος 836



Σχήμα 4.5. 12 Επιλογή 2: Εμφανίζει λίστα με όλους τους πελάτες.

Σχήμα 4.5. 13 Επιλογή 3: Εμφανίζει τα στοιχεία του πελάτη με κωδικό 1.

```
Please give customer's last name
blackwell
First name:Jackie
Middle name:E.
Last name:Blackwell
Phone:972-555-0163
E-mail Address:jackie0@adventure-works.com
Press any key to continue . . .
```

Σχήμα 4.5. 14 Επιλογή 4: Εμφανίζει τα στοιχεία του πελάτη με το επίθετο Blackwell

```
Sales Orders menu
1.Get list of sales orders by using customer's ID
2.Get list of all sales orders
3.Find sales order details by using sales order ID
0.Return to the search menu
Please enter your selection
```

Σχήμα 4.5. 15 Το μενού αναζήτησης παραγγελιών και οι δυνατότητές του.

```
Please give customer's ID
30113
Number: SO71780 , Order Date: 01/06/2004 00:00:00
```

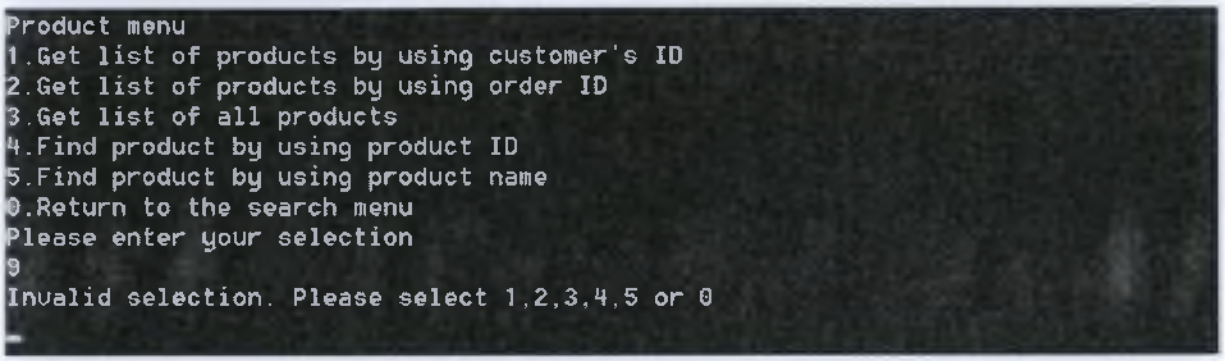
Σχήμα 4.5. 16 Επιλογή 1: Η λίστα παραγγελιών του πελάτη με κωδικό 30113.

```
List of all Sales Orders
Number  SO71774   Order Date  01/06/2004 00:00:00
Number  SO71776   Order Date  01/06/2004 00:00:00
Number  SO71780   Order Date  01/06/2004 00:00:00
Number  SO71782   Order Date  01/06/2004 00:00:00
Number  SO71783   Order Date  01/06/2004 00:00:00
Number  SO71784   Order Date  01/06/2004 00:00:00
Number  SO71796   Order Date  01/06/2004 00:00:00
Number  SO71797   Order Date  01/06/2004 00:00:00
Number  SO71815   Order Date  01/06/2004 00:00:00
Number  SO71816   Order Date  01/06/2004 00:00:00
Number  SO71831   Order Date  01/06/2004 00:00:00
Number  SO71832   Order Date  01/06/2004 00:00:00
Number  SO71845   Order Date  01/06/2004 00:00:00
Number  SO71846   Order Date  01/06/2004 00:00:00
Number  SO71856   Order Date  01/06/2004 00:00:00
Number  SO71858   Order Date  01/06/2004 00:00:00
Number  SO71863   Order Date  01/06/2004 00:00:00
Number  SO71867   Order Date  01/06/2004 00:00:00
Number  SO71885   Order Date  01/06/2004 00:00:00
Number  SO71895   Order Date  01/06/2004 00:00:00
Number  SO71897   Order Date  01/06/2004 00:00:00
Number  SO71898   Order Date  01/06/2004 00:00:00
Number  SO71899   Order Date  01/06/2004 00:00:00
Number  SO71902   Order Date  01/06/2004 00:00:00
Number  SO71915   Order Date  01/06/2004 00:00:00
Number  SO71917   Order Date  01/06/2004 00:00:00
Number  SO71920   Order Date  01/06/2004 00:00:00
Number  SO71923   Order Date  01/06/2004 00:00:00
Number  SO71935   Order Date  01/06/2004 00:00:00
Number  SO71936   Order Date  01/06/2004 00:00:00
Number  SO71938   Order Date  01/06/2004 00:00:00
Number  SO71946   Order Date  01/06/2004 00:00:00

Press any key to continue . . .
```

Σχήμα 4.5. 17 Επιλογή 2: Η λίστα με όλες τις παραγγελίες

Σχήμα 4.5. 18 Επιλογή 3:Ο κωδικός της παραγγελίας που αντιστοιχούν τα ακόλουθα στοιχεία είναι 71774.



Σχήμα 4.5. 19 Μήνυμα λάθους σε περίπτωση σφάλματος του χρήστη.

Σχήμα 4.5. 20 Μήνυμα ειδοποίησης ότι ο πελάτης αυτός δεν βρέθηκε.

Συμπέρασμα

Παρόλο που οι υπηρεσίες ιστού θεωρούνται η τεχνολογία που επέδρασε καταλυτικά στην ανάπτυξη του ιστού δεν παύουν να παραμένουν ένα πεδίο που βρίσκεται ακόμα σε πρώιμο στάδιο. Πολλές επιπλέον τεχνολογίες πρέπει να ολοκληρώσουν τον πυρήνα αυτών των υπηρεσιών γι να είναι ικανές να χρησιμοποιηθούν σε πολύπλοκες και απαιτητικές εταιρικές εφαρμογές. Για να αποκτήσουν προσδόκιμο ζωής οι υπηρεσίες ιστού χρειάζονται ανάπτυξη σε θέματα ασφάλειας, transactions , διαδικασίας ροής , βελτίωση στα μηνύματα και σε άλλες ποιοτικές λειτουργίες. Μια καλύτερα δομημένη αρχιτεκτονική είναι απαραίτητη που να συμπεριλαμβάνει τις καινούριες βελτιώσεις.

Προσπάθειες για να επιτευχθούν οι παραπάνω στόχοι έχουν αρχίσει να γίνονται. Ορίζονται καινούρια standards για την ασφάλεια με σκοπό να παρέχουν προστασία στα έγγραφα και εμπιστευτικότητα. Ακόμα δημιουργούνται μηχανισμοί ελέγχου για το ποιος επιτρέπεται να έχει πρόσβαση στην υπηρεσία ιστού και σε ποιο interface του παρέχεται. Εάν επιτευχθούν όλα αυτά θα οδηγηθούμε στο επόμενο πεδίο λειτουργικότητας των υπηρεσιών ιστού.

Το επόμενο στάδιο των υπηρεσιών που θα έχουν εκπληρώσει τα παραπάνω κριτήρια θα μπορούσαν να είναι οι semantic υπηρεσίες ιστού οι οποίες αποτελούν την εξέλιξη των απλών υπηρεσιών ιστού. Οι απλές υπηρεσίες ιστού βασίζονται στο συντακτικό και όχι στη σημασιολογική έννοια των δεδομένων που μεταφέρουν. Εάν κατάφερναν να συσχετίσουν στοιχεία με βάση τη σημασιολογία τους ο προγραμματιστής θα μπορούσε να συνδέσει ευκολότερα διάφορους πόρους με νόημα. Οι semantic web services θα μπορούσαν να τρέχουν πίσω από μία αναζήτηση στον ιστό και να παράγουν πολύ καλύτερα αποτελέσματα αποκλείοντας συντακτικά σωστά αλλά σημασιολογικά λάθος αποτελέσματα που θα εμφανίσει ο browser.

Βιβλιογραφία

Βιβλία που χρησιμοποιήθηκαν ως πηγές:

1. ***Windows Communicaton Foundation Step By Step***
   By John Sharp

2. **LinQ for Dummies**
   by John Paul Mueller

3. **LINQ TO OBJECTS USING C# 4.0 USING AND EXTENDING LINQ TO OBJECTS AND PARALLEL LINQ (PLINQ)**
   By Troy Magennis

4. **Service Design Patterns: Fundamental Design Solutions for SOAP WSDL and RESTful Web Services**
   by: Robert Daigneau

5. **Google, Amazon, and Beyond: Creating and Consuming Web Services**
   by: Alexander Nakhimovsky

6. **Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI (2nd Edition)**
   by: Steve Graham, Doug Davis, Simeon Simeonov, Glen Daniels, Peter Brittenham, Yuichi Nakamura, Paul Fremantle, Dieter Koenig, Claudia Zentner

7. **Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More**
   by: Sanjiva Weerawarana, Francisco Curbera, Frank Leymann, Tony Storey, Donald F. Ferguson

8. **Understanding Web Services: XML, WSDL, SOAP, and UDDI**
   by: Eric Newcomer

9. **Programming Web Services with SOAP**
   by: James Snell

10. **Architecting Web Services**
    by: William L. Oellermann Jr.

Papers που χρησιμοποιήθηκαν ως πηγές:

1. **Introduction to Web Services**
   by Hartwig Gunzer, Sales Engineer, Borland

2. **Introduction to Web services architecture**
   by K. Gottschalk, S. Graham, H. Kreger, J. Snell

3. **Unraveling the Web Services Web. An Introduction to SOAP,WSDL, and UDD**
   By Francisco Curbera, Matthew Duftler, Rania Khalaf,William Nagy,
   Nirmal Mukhi, and Sanjiva Weerawarana • IBM T.J.Watson Research Center

4. **Introducing Windows Communication Foundation**
   By David Chappell, Chappell & Associates, January 2010

5. **SEMANTIC WEB SERVICES: A RESTFUL APPROACH**
   By Otávio Freitas Ferreira Filho, Maria Alice Grigas Varella Ferreira
   University of São Paulo, Polytechnic School
   São Paulo, Brazil

Ιστοσελίδες που χρησιμοποιήθηκαν ως πηγές:

Σελίδες με πληροφορίες που αφορούν τις υπηρεσίες ιστού:

1. http://en.wikipedia.org/wiki/Web_service
2. http://www.w3schools.com/webservices/ws_intro.asp
3. http://searchsoa.techtarget.com/definition/Web-services
4. http://www.codeproject.com/KB/XML/Defining_Web_Services.aspx
5. http://www.alistapart.com/articles/webservices/
6. http://www.wisegeek.com/what-are-web-services.htm
7. http://www.tutorialspoint.com/webservices/why_web_services.htm
8. http://www.vkinfotek.com/webservice/deploy-web-service.aspx
9. http://www.w3.org/DesignIssues/WebServices.html
10. http://www.w3.org/TR/ws-arch
11. http://www.ibm.com/developerworks/webservices/library/ws-featuddi/
12. http://www.ibm.com/developerworks/webservices/library/ws-wsa/#figure2
13. http://www.w3.org/TR/ws-arch
14. http://msdn.microsoft.com/en-us/library/dd936243.aspx
15. http://msdn.microsoft.com/en-us/magazine/cc163647.aspx
16. http://msdn.microsoft.com/en-us/library/bb945107.aspx
17. http://msdn.microsoft.com/en-us/library/ms731073.aspx#Y4206

Σελίδες με πληροφορίες που αφορούν την ιστορία των υπηρεσιών ιστού:

1. http://www.databaseanswers.org/web_services_history.htm
2. http://www.w3.org/2002/ws/history.html

Σελίδες με πληροφορίες που αφορούν το SOAP :

1. http://en.wikipedia.org/wiki/SOAP_(protocol)
2. http://webdesign.about.com/od/soap/a/what-is-xml-soap.htm
3. http://classes.soe.ucsc.edu/cmps183/Winter04/lectures/SOAP-pt1.pdf
4. http://www.ehow.com/about_6631405_soap-protocol_.html
5. http://searchsoa.techtarget.com/definition/SOAP
6. http://static.userland.com/xmlRpcCom/soap/SOAPv11.htm
7. http://www.internetwk.com/breakingNews/INW20021209S0010/
8. http://www.ibm.com/developerworks/library/ws-soap/?dwzone=ws

Σελίδες με πληροφορίες που αφορούν το REST:

1. http://en.wikipedia.org/wiki/Representational_State_Transfer
2. http://searchsoa.techtarget.com/definition/REST
3. http://www.xfront.com/REST-Web-Services.html
4. http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
5. http://www.stefan-marr.de/downloads/RESTful-Web-Services.slides.pdf
6. http://www.infoq.com/news/2009/05/Rest

Σελίδες με πληροφορίες που αφορούν το UDDI:

1. http://en.wikipedia.org/wiki/Universal_Description_Discovery_and_Integration

Σελίδες με πληροφορίες που αφορούν το WSDL:

1. http://en.wikipedia.org/wiki/Web_Services_Description_Language

Σελίδες με πληροφορίες που αφορούν το WADL:

1. http://wikis.sun.com/display/Jersey/WADL
2. http://www.w3.org/Submission/wadl/#x3-10001
3. http://www.ajaxonomy.com/2008/xml/web-services-part-2-wsdl-and-wadl
4. http://weblogs.java.net/blog/mhadley/archive/2005/05/introducing_wad.html
5. http://bitworking.org/news/193/Do-we-need-WADL

Σελίδες με πληροφορίες που αφορούν το WCF:

1. http://msdn.microsoft.com/en-us/library/aa738737.aspx
2. http://www.dotnetfunda.com/articles/article221.aspx#WhatarethemaincomponentsofWCF
3. http://www.dotnetfunda.com/articles/article495-wcf-faq-part-5-transactions-.aspx
4. http://www.dotnetfunda.com/interview/exam286-what-are-contracts-in-wcf-.aspx
5. http://itknowledgeexchange.techtarget.com/itanswers/difference-between-webservices-and-wcf/

6. http://r4r.co.in/WCF/01/tutorial/basic/Difference%20between%20WCF%20and%20Web%20servicshtml
7. http://www.codegain.com/tips/wcf/general/difference-between-web-service-and-wcf-service.aspx
8. http://jeffbarnes.net/blog/post/2007/05/10/WCF-Serialization-and-Generics.aspx

Σελίδες με πληροφορίες που αφορούν την ασφάλεια των υπηρεσιών ιστού:

1. http://msdn.microsoft.com/en-us/library/ms996507.aspx
2. http://www.xml.com/pub/a/ws/2003/03/04/security.html
3. http://www.ibm.com/developerworks/library/specification/ws-secure/

Παράρτημα Α

Αυτό είναι το πλήρες WSDL αρχείο

```
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:soap
enc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-
1.0.xsd" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:tns="http:
//tempuri.org/" xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/
policy" xmlns:wsap="http://schemas.xmlsoap.org/ws/2004/08/addressing/policy" xmlns:wsaw="http://www.w3.org/2006/05/addressing
/wsdl"xmlns:msc="http://schemas.microsoft.com/ws/2005/12/wsdl/contract" xmlns:wsa10="http://www.w3.org/2005/08/addressing" xm
lns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"name="ProductsS
erviceImpl" targetNamespace="http://tempuri.org/">
<wsdl:types>
<xsd:schema targetNamespace="http://tempuri.org/Imports">
<xsd:import schemaLocation="http://localhost:50385/ProductsService/Service.svc?xsd=xsd0" namespace="http://tempuri.org/"/>
<xsd:import schemaLocation="http://localhost:50385/ProductsService/Service.svc?xsd=xsd1" namespace="http://schemas.microsoft.
com/2003/10/Serialization/"/>
<xsd:import schemaLocation="http://localhost:50385/ProductsService/Service.svc?xsd=xsd2" namespace="http://schemas.datacontra
ct.org/2004/07/Products"/>
</xsd:schema>
</wsdl:types>
<wsdl:message name="IProductsService_ListProducts_InputMessage">
<wsdl:part name="parameters" element="tns:ListProducts"/>
</wsdl:message>
<wsdl:message name="IProductsService_ListProducts_OutputMessage">
<wsdl:part name="parameters" element="tns:ListProductsResponse"/>
</wsdl:message>
<wsdl:message name="IProductsService_ListProductsOr_InputMessage">
<wsdl:part name="parameters" element="tns:ListProductsOr"/>
</wsdl:message>
<wsdl:message name="IProductsService_ListProductsOr_OutputMessage">
<wsdl:part name="parameters" element="tns:ListProductsOrResponse"/>
</wsdl:message>
<wsdl:message name="IProductsService_ListCustomers_InputMessage">
<wsdl:part name="parameters" element="tns:ListCustomers"/>
</wsdl:message>
<wsdl:message name="IProductsService_ListCustomers_OutputMessage">
<wsdl:part name="parameters" element="tns:ListCustomersResponse"/>
</wsdl:message>
<wsdl:message name="IProductsService_ListSalesOrders_InputMessage">
<wsdl:part name="parameters" element="tns:ListSalesOrders"/>
</wsdl:message>
<wsdl:message name="IProductsService_ListSalesOrders_OutputMessage">
<wsdl:part name="parameters" element="tns:ListSalesOrdersResponse"/>
</wsdl:message>
<wsdl:message name="IProductsService_ListProductsAll_InputMessage">
<wsdl:part name="parameters" element="tns:ListProductsAll"/>
</wsdl:message>
<wsdl:message name="IProductsService_ListProductsAll_OutputMessage">
<wsdl:part name="parameters" element="tns:ListProductsAllResponse"/>
</wsdl:message>
<wsdl:message name="IProductsService_ListCustomersAll_InputMessage">
<wsdl:part name="parameters" element="tns:ListCustomersAll"/>
</wsdl:message>
<wsdl:message name="IProductsService_ListCustomersAll_OutputMessage">
<wsdl:part name="parameters" element="tns:ListCustomersAllResponse"/>
</wsdl:message>
<wsdl:message name="IProductsService_ListSalesOrdersAll_InputMessage">
<wsdl:part name="parameters" element="tns:ListSalesOrdersAll"/>
</wsdl:message>
<wsdl:message name="IProductsService_ListSalesOrdersAll_OutputMessage">
<wsdl:part name="parameters" element="tns:ListSalesOrdersAllResponse"/>
</wsdl:message>
<wsdl:message name="IProductsService_GetProduct_InputMessage">
<wsdl:part name="parameters" element="tns:GetProduct"/>
</wsdl:message>
<wsdl:message name="IProductsService_GetProduct_OutputMessage">
<wsdl:part name="parameters" element="tns:GetProductResponse"/>
</wsdl:message>
<wsdl:message name="IProductsService_GetSalesOrder_InputMessage">
<wsdl:part name="parameters" element="tns:GetSalesOrder"/>
</wsdl:message>
<wsdl:message name="IProductsService_GetSalesOrder_OutputMessage">
<wsdl:part name="parameters" element="tns:GetSalesOrderResponse"/>
</wsdl:message>
<wsdl:message name="IProductsService_GetCustomer_InputMessage">
<wsdl:part name="parameters" element="tns:GetCustomer"/>
</wsdl:message>
<wsdl:message name="IProductsService_GetCustomer_OutputMessage">
<wsdl:part name="parameters" element="tns:GetCustomerResponse"/>
</wsdl:message>
<wsdl:message name="IProductsService_GetProducts_InputMessage">
<wsdl:part name="parameters" element="tns:GetProducts"/>
</wsdl:message>
<wsdl:message name="IProductsService_GetProducts_OutputMessage">
<wsdl:part name="parameters" element="tns:GetProductsResponse"/>
</wsdl:message>
<wsdl:message name="IProductsService_GetCustomers_InputMessage">
<wsdl:part name="parameters" element="tns:GetCustomers"/>
</wsdl:message>
<wsdl:message name="IProductsService_GetCustomers_OutputMessage">
```

```
<wsdl:part name="parameters" element="tns:GetCustomersResponse"/>
</wsdl:message>
<wsdl:portType name="IProductsService">
<wsdl:operation name="ListProducts">
<wsdl:input wsaw:Action="http://tempuri.org/IProductsService/ListProducts" message="tns:IProductsService_ListProducts_InputMe
ssage"/>
<wsdl:output wsaw:Action="http://tempuri.org/IProductsService/ListProductsResponse" message="tns:IProductsService_ListProduct
s_OutputMessage"/>
</wsdl:operation>
<wsdl:operation name="ListProductsOr">
<wsdl:input wsaw:Action="http://tempuri.org/IProductsService/ListProductsOr" message="tns:IProductsService_ListProductsOr_Inp
utMessage"/>
<wsdl:output wsaw:Action="http://tempuri.org/IProductsService/ListProductsOrResponse" message="tns:IProductsService_ListProdu
ctsOr_OutputMessage"/>
</wsdl:operation>
<wsdl:operation name="ListCustomers">
<wsdl:input wsaw:Action="http://tempuri.org/IProductsService/ListCustomers" message="tns:IProductsService_ListCustomers_Input
Message"/>
<wsdl:output wsaw:Action="http://tempuri.org/IProductsService/ListCustomersResponse" message="tns:IProductsService_ListCustom
ers_OutputMessage"/>
</wsdl:operation>
<wsdl:operation name="ListSalesOrders">
<wsdl:input wsaw:Action="http://tempuri.org/IProductsService/ListSalesOrders" message="tns:IProductsService_ListSalesOrders_I
nputMessage"/>
<wsdl:output wsaw:Action="http://tempuri.org/IProductsService/ListSalesOrdersResponse" message="tns:IProductsService_ListSale
sOrders_OutputMessage"/>
</wsdl:operation>
<wsdl:operation name="ListProductsAll">
<wsdl:input wsaw:Action="http://tempuri.org/IProductsService/ListProductsAll" message="tns:IProductsService_ListProductsAll_I
nputMessage"/>
<wsdl:output wsaw:Action="http://tempuri.org/IProductsService/ListProductsAllResponse" message="tns:IProductsService_ListProd
uctsAll_OutputMessage"/>
</wsdl:operation>
<wsdl:operation name="ListCustomersAll">
<wsdl:input wsaw:Action="http://tempuri.org/IProductsService/ListCustomersAll" message="tns:IProductsService_ListCustomersAll
_InputMessage"/>
<wsdl:output wsaw:Action="http://tempuri.org/IProductsService/ListCustomersAllResponse" message="tns:IProductsService_ListCus
tomersAll_OutputMessage"/>
</wsdl:operation>
<wsdl:operation name="ListSalesOrdersAll">
<wsdl:input wsaw:Action="http://tempuri.org/IProductsService/ListSalesOrdersAll" message="tns:IProductsService_ListSalesOrder
sAll_InputMessage"/>
<wsdl:output wsaw:Action="http://tempuri.org/IProductsService/ListSalesOrdersAllResponse" message="tns:IProductsService_ListS
alesOrdersAll_OutputMessage"/>
</wsdl:operation>
<wsdl:operation name="GetProduct">
<wsdl:input wsaw:Action="http://tempuri.org/IProductsService/GetProduct" message="tns:IProductsService_GetProduct_InputMessag
e"/>
<wsdl:output wsaw:Action="http://tempuri.org/IProductsService/GetProductResponse" message="tns:IProductsService_GetProduct_Ou
tputMessage"/>
</wsdl:operation>
<wsdl:operation name="GetSalesOrder">
<wsdl:input wsaw:Action="http://tempuri.org/IProductsService/GetSalesOrder" message="tns:IProductsService_GetSalesOrder_Input
Message"/>
<wsdl:output wsaw:Action="http://tempuri.org/IProductsService/GetSalesOrderResponse" message="tns:IProductsService_GetSalesOr
der_OutputMessage"/>
</wsdl:operation>
<wsdl:operation name="GetCustomer">
<wsdl:input wsaw:Action="http://tempuri.org/IProductsService/GetCustomer" message="tns:IProductsService_GetCustomer_InputMess
age"/>
<wsdl:output wsaw:Action="http://tempuri.org/IProductsService/GetCustomerResponse" message="tns:IProductsService_GetCustomer_
OutputMessage"/>
</wsdl:operation>
<wsdl:operation name="GetProducts">
<wsdl:input wsaw:Action="http://tempuri.org/IProductsService/GetProducts" message="tns:IProductsService_GetProducts_InputMess
age"/>
<wsdl:output wsaw:Action="http://tempuri.org/IProductsService/GetProductsResponse" message="tns:IProductsService_GetProducts_
OutputMessage"/>
</wsdl:operation>
<wsdl:operation name="GetCustomers">
<wsdl:input wsaw:Action="http://tempuri.org/IProductsService/GetCustomers" message="tns:IProductsService_GetCustomers_InputMe
ssage"/>
<wsdl:output wsaw:Action="http://tempuri.org/IProductsService/GetCustomersResponse" message="tns:IProductsService_GetCustomer
s_OutputMessage"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="BasicHttpBinding_IProductsService" type="tns:IProductsService">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="ListProducts">
<soap:operation soapAction="http://tempuri.org/IProductsService/ListProducts" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="ListProductsOr">
<soap:operation soapAction="http://tempuri.org/IProductsService/ListProductsOr" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="ListCustomers">
<soap:operation soapAction="http://tempuri.org/IProductsService/ListCustomers" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
```

```
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="ListSalesOrders">
<soap:operation soapAction="http://tempuri.org/IProductsService/ListSalesOrders" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="ListProductsAll">
<soap:operation soapAction="http://tempuri.org/IProductsService/ListProductsAll" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="ListCustomersAll">
<soap:operation soapAction="http://tempuri.org/IProductsService/ListCustomersAll" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="ListSalesOrdersAll">
<soap:operation soapAction="http://tempuri.org/IProductsService/ListSalesOrdersAll" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="GetProduct">
<soap:operation soapAction="http://tempuri.org/IProductsService/GetProduct" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="GetSalesOrder">
<soap:operation soapAction="http://tempuri.org/IProductsService/GetSalesOrder" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="GetCustomer">
<soap:operation soapAction="http://tempuri.org/IProductsService/GetCustomer" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="GetProducts">
<soap:operation soapAction="http://tempuri.org/IProductsService/GetProducts" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="GetCustomers">
<soap:operation soapAction="http://tempuri.org/IProductsService/GetCustomers" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="ProductsServiceImpl">
<wsdl:port name="BasicHttpBinding_IProductsService" binding="tns:BasicHttpBinding_IProductsService">
<soap:address location="http://localhost:50385/ProductsService/Service.svc"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Αυτό είναι το WSDL αρχείο που ορίζει τους τύπους.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://tempuri.org/" elementFormDefault="qualified" targetN
amespace="http://tempuri.org/">
<xs:import schemaLocation="http://localhost:50385/ProductsService/Service.svc?xsd=xsd2" namespace="http://schemas.datacontrac
t.org/2004/07/Products"/>
```

```
<xs:element name="ListProducts">
<xs:complexType>
<xs:sequence>
<xs:element minOccurs="0" name="customerID" type="xs:int"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="ListProductsResponse">
<xs:complexType>
<xs:sequence>
<xs:element xmlns:q1="http://schemas.datacontract.org/2004/07/Products" minOccurs="0" name="ListProductsResult" nillable="tru
e" type="q1:ArrayOfProductData"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="ListProductsOr">
<xs:complexType>
<xs:sequence>
<xs:element minOccurs="0" name="salesOrderID" type="xs:int"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="ListProductsOrResponse">
<xs:complexType>
<xs:sequence>
<xs:element xmlns:q2="http://schemas.datacontract.org/2004/07/Products" minOccurs="0" name="ListProductsOrResult" nillable="t
rue" type="q2:ArrayOfProductData"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="ListCustomers">
<xs:complexType>
<xs:sequence>
<xs:element minOccurs="0" name="productID" type="xs:int"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="ListCustomersResponse">
<xs:complexType>
<xs:sequence>
<xs:element xmlns:q3="http://schemas.datacontract.org/2004/07/Products" minOccurs="0" name="ListCustomersResult" nillable="tr
ue" type="q3:ArrayOfCustomerData"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="ListSalesOrders">
<xs:complexType>
<xs:sequence>
<xs:element minOccurs="0" name="customerID" type="xs:int"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="ListSalesOrdersResponse">
<xs:complexType>
<xs:sequence>
<xs:element xmlns:q4="http://schemas.datacontract.org/2004/07/Products" minOccurs="0" name="ListSalesOrdersResult" nillable="
true" type="q4:ArrayOfSalesOrderData"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="ListProductsAll">
<xs:complexType>
<xs:sequence/>
</xs:complexType>
</xs:element>
<xs:element name="ListProductsAllResponse">
<xs:complexType>
<xs:sequence>
<xs:element xmlns:q5="http://schemas.datacontract.org/2004/07/Products" minOccurs="0" name="ListProductsAllResult" nillable="
true" type="q5:ArrayOfProductData"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="ListCustomersAll">
<xs:complexType>
<xs:sequence/>
</xs:complexType>
</xs:element>
<xs:element name="ListCustomersAllResponse">
<xs:complexType>
<xs:sequence>
<xs:element xmlns:q6="http://schemas.datacontract.org/2004/07/Products" minOccurs="0" name="ListCustomersAllResult" nillable=
"true" type="q6:ArrayOfCustomerData"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="ListSalesOrdersAll">
<xs:complexType>
<xs:sequence/>
</xs:complexType>
</xs:element>
<xs:element name="ListSalesOrdersAllResponse">
<xs:complexType>
<xs:sequence>
<xs:element xmlns:q7="http://schemas.datacontract.org/2004/07/Products" minOccurs="0" name="ListSalesOrdersAllResult" nillabl
e="true" type="q7:ArrayOfSalesOrderData"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="GetProduct">
<xs:complexType>
<xs:sequence>
```

```
<xs:element minOccurs="0" name="productID" type="xs:int"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="GetProductResponse">
<xs:complexType>
<xs:sequence>
<xs:element xmlns:q8="http://schemas.datacontract.org/2004/07/Products" minOccurs="0" name="GetProductResult" nillable="true"
 type="q8:ProductData"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="GetSalesOrder">
<xs:complexType>
<xs:sequence>
<xs:element minOccurs="0" name="salesOrderID" type="xs:int"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="GetSalesOrderResponse">
<xs:complexType>
<xs:sequence>
<xs:element xmlns:q9="http://schemas.datacontract.org/2004/07/Products" minOccurs="0" name="GetSalesOrderResult" nillable="tr
ue" type="q9:SalesOrderData"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="GetCustomer">
<xs:complexType>
<xs:sequence>
<xs:element minOccurs="0" name="customerID" type="xs:int"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="GetCustomerResponse">
<xs:complexType>
<xs:sequence>
<xs:element xmlns:q10="http://schemas.datacontract.org/2004/07/Products" minOccurs="0" name="GetCustomerResult" nillable="tru
e" type="q10:CustomerData"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="GetProducts">
<xs:complexType>
<xs:sequence>
<xs:element minOccurs="0" name="name" nillable="true" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="GetProductsResponse">
<xs:complexType>
<xs:sequence>
<xs:element xmlns:q11="http://schemas.datacontract.org/2004/07/Products" minOccurs="0" name="GetProductsResult" nillable="tru
e" type="q11:ArrayOfProductData"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="GetCustomers">
<xs:complexType>
<xs:sequence>
<xs:element minOccurs="0" name="lastname" nillable="true" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="GetCustomersResponse">
<xs:complexType>
<xs:sequence>
<xs:element xmlns:q12="http://schemas.datacontract.org/2004/07/Products" minOccurs="0" name="GetCustomersResult" nillable="tr
ue" type="q12:ArrayOfCustomerData"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Παράρτημα Β

Άλλα αρχεία που περιέχονται σε μια WCF εφαρμογή

Σε κάθε WCF service υπάρχει το αρχείο service.svc το οποίο εμπεριέχει το όνομα και την τοποθεσία της κλάσης που υλοποιεί το WCF. Όπως φαίνεται και στο σχήμα 4.3.2 η κλάση ονομάζεται Products.ProductsServiceImpl και βρίσκεται στο "~/App_Code/ProductsService.cs".

```
<%@ ServiceHost Language="C#" Debug="true" Service="Products.ProductsServiceImpl" CodeBehind="~/App_Code/ProductsService.cs" %>
```

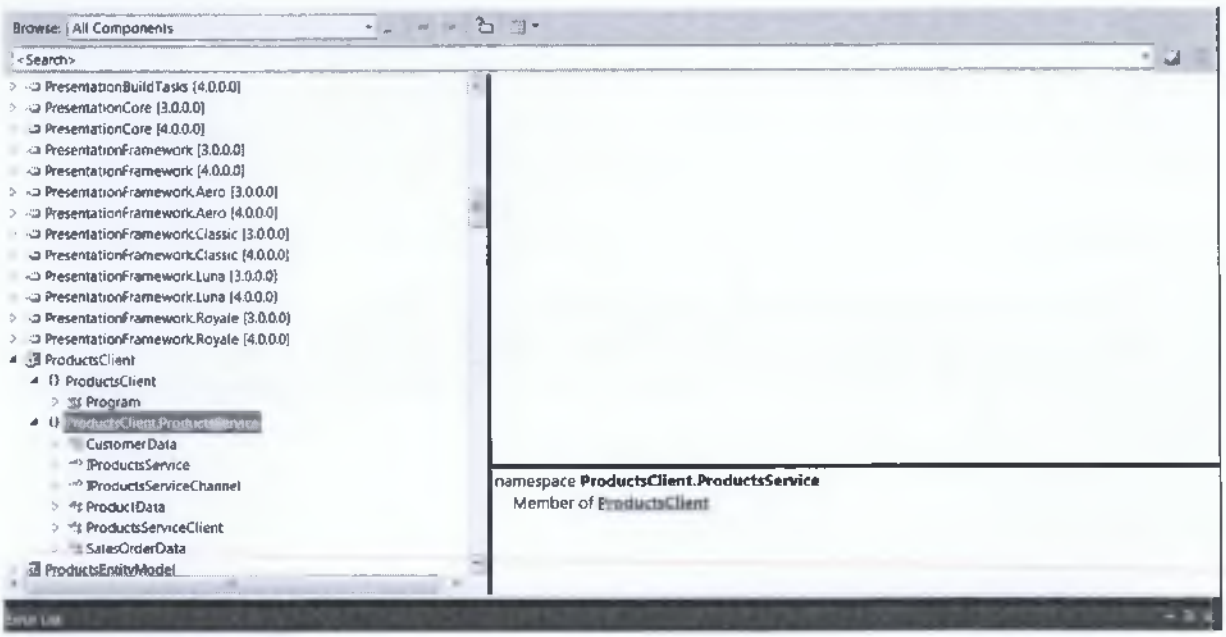
Ακόμα η ProductsService περιέχει το αρχείο Web.config . Βλέπουμε στο σχήμα που ακολουθεί ότι η ετικέτα <behavior> ορίζει το metadata endpoint και θέτοντας την τιμή του true αφήνει την υπηρεσία να εκθέτει μεταδεδομένα. Από αυτό το endpoint θα δημιουργηθεί αργότερα η proxy class.

```
<?xml version="1.0"?>
<configuration>
    <connectionStrings>
        <add name="AdventureWorksLTEntities" connectionString="metadata=res://*/ProductsModel.csdl|res://*/ProductsModel.ssdl|res://*/ProductsModel.msl;provider=Sy
    </connectionStrings>
    <system.web>
        <compilation debug="true" targetFramework="4.0">
            <assemblies>
                <add assembly="System.Data.Entity, Version=4.0.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089"/></assemblies></compilation>
    </system.web>
    <system.serviceModel>
        <behaviors>
            <serviceBehaviors>
                <behavior>
                    <!-- To avoid disclosing metadata information, set the value below to false and remove the metadata endpoint above before deployment -->
                    <serviceMetadata httpGetEnabled="true"/>
                    <!-- To receive exception details in faults for debugging purposes, set the value below to true.  Set to false before deployment to avoid disc
                    <serviceDebug includeExceptionDetailInFaults="false"/>
                </behavior>
            </serviceBehaviors>
        </behaviors>
        <serviceHostingEnvironment multipleSiteBindingsEnabled="true"/>
    </system.serviceModel>
    <system.webServer>
        <modules runAllManagedModulesForAllRequests="true"/>
    </system.webServer>
</configuration>
```

Στο Product Entity Model περιέχεται το αρχείο App.config στο οποίο περιέχεται ένα αλφαριθμητικό που δημιουργήθηκε για να συνδέει την υπηρεσία με τη βάση δεδομένων. Το όνομα της βάσης δεδομένων είναι AdventureWorksLT και το αλφαριθμητικό που κάνει τη σύνδεση με την ProductsService είναι το
"metadata=res://*/ProductsModel.csdl|res://*/ProductsModel.ssdl|res://*/ProductsModel.msl;provider=System.Data.SqlClient;provider connection string="Data Source=.\SQLExpress;Initial Catalog=AdventureWorksLT;Integrated Security=True;MultipleActiveResultSets=True"" providerName="System.Data.EntityClient".

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <connectionStrings>
    <add name="AdventureWorksLTEntities" connectionString="metadata=res://*/ProductsModel.csdl|res://*/ProductsModel.ssdl|res://*/Pro
  </connectionStrings>
</configuration>
```

Η ProductsClient που όπως προείπαμε είναι η WCF client application της εφαρμογής μας περιέχει το ProductsService το οποίο είναι το αρχείο που περιέχει την proxy class. Στην εικονα 4.3.6 φαίνεται η τοποθεσία της proxy class που βρίσκεται στο namespace ProductsClient.ProductsService



Επιπρόσθετα η ProductsClient περιέχει και το app.config στο οποίο υπάρχει κώδικας που χρησιμοποιείται για τη σύνδεση με τη WCF service. Η ετικέτα <client> ορίζει πως ο πελάτης θα συνδεθεί με το WCF service πιο συγκεκριμένα ορίζει το endpoint. Μεσα σε αυτό δίνεται :

- Η διεύθυνσή του που θα στέλνει ο client τα request: "http://localhost:49331/ProductsService/Service.svc"
- Ο τύπος του binding που χρησιμοποιείται(BasicHttpBinding).
- Το contract που υλοποιεί(ProductsService.IProductsService).

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <system.serviceModel>
        <bindings>
            <basicHttpBinding>
                <binding name="BasicHttpBinding_IProductsService" closeTimeout="00:01:00"
                    openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
                    allowCookies="false" bypassProxyOnLocal="false" hostNameComparisonMode="StrongWildcard"
                    maxBufferSize="524288" maxBufferPoolSize="524288" maxReceivedMessageSize="524288"
                    messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
                    useDefaultWebProxy="true">
                    <readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"
                        maxBytesPerRead="4096" maxNameTableCharCount="16384" />
                    <security mode="None">
                        <transport clientCredentialType="None" proxyCredentialType="None"
                            realm="" />
                        <message clientCredentialType="UserName" algorithmSuite="Default" />
                    </security>
                </binding>
            </basicHttpBinding>
        </bindings>
        <client>
            <endpoint address="http://localhost:49331/ProductsService/Service.svc"
                binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding_IProductsService"
                contract="ProductsService.IProductsService" name="BasicHttpBinding_IProductsService" />
        </client>
    </system.serviceModel>
</configuration>
```