Α.Τ.Ε.Ι ΠΕΛΟΠΟΝΝΗΣΟΥ
ΤΜΗΜΑ: ΛΟΓΙΣΤΙΚΗΣ ΚΑΙ ΧΡΗΜ/ΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΚΑΛΑΜΑΤΑ 2013-2014

ΘΕΜΑ:
Οργάνωση του συστήματος ραντεβού της επιχείρησης «Figura Helena» με την μέθοδο B.P.M

Φοιτητής : Μαυροβουνιώτης Φώτιος

Υπεύθυνος καθηγητής : Νικολαΐδης Νικόλαος

Α.Τ.Ε.Ι ΠΕΛΟΠΟΝΝΗΣΟΥ
ΤΜΗΜΑ: ΛΟΓΙΣΤΙΚΗΣ ΚΑΙ ΧΡΗΜ/ΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΚΑΛΑΜΑΤΑ 2013-2014

ΘΕΜΑ:
Οργάνωση του συστήματος ραντεβού της επιχείρησης «Figura Helena» με την μέθοδο B.P.M

Φοιτητής : Μαυροβουνιώτης Φώτιος

Υπεύθυνος καθηγητής : Νικολαΐδης Νικόλαος

Περιεχόμενα

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

ΕΥΡΕΤΗΡΙΟ ΣΧΕΔΙΑΓΡΑΜΑΤΩΝ

« Οι απόψει που διατυπόνονται μέσα στην εργασία είναι αποκλιστηκά δικές μου.
Δεν απηχούν τις αποψεις και την γνώμη του ιδρύματος»

1. ΠΡΟΛΟΓΟΣ

Θα ήθελα να ξεκινήσω την εργασία μου, παραθέτοντας, ένα παλαιό ανέκδοτο, πολύ γνωστό στην Ευρώπη αλλά ελάχιστα γνωστό στην χώρα μας.

Παράδεισος είναι εκεί όπου:	**Κόλαση** είναι εκεί όπου:
οι μάγειροι είναι Γάλλοι	οι μάγειροι είναι Άγγλοι
οι μηχανικοί είναι Γερμανοί	οι μηχανικοί είναι Γάλλοι
οι αστυνόμοι είναι Άγγλοι	οι αστυνόμοι είναι Γερμανοί
οι εραστές είναι Έλληνες	οι εραστές είναι Ελβετοί
την οργάνωση έχουν αναλάβει οι Ελβετοί.	την οργάνωση έχουν αναλάβει οι **Έλληνες**.

Είναι το ανέκδοτο με το οποίο οι υπόλοιποι Ευρωπαίοι σατιρίζουν την ανικανότητα των Ελλήνων να οργανωθούν.
Δυστυχώς για εμάς, η οικονομική κρίση και το μέγεθος του πλήγματος που δέχτηκε η χώρα μας τους επιβεβαιώνει.

Αλλά δεν είναι οι Ευρωπαίοι που μας διακωμωδούν για την ανικανότητα που έχουμε σαν λαός να οργανωθούμε. Το χειρότερο είναι όταν Έλληνες πολιτικοί κάνουν το ίδιο.

Το 2010 ο Γιάννος Παπαντωνίου, προσπαθώντας να εξηγήσει την κατρακύλα της χώρας μας, δήλωνε στο Al Jazeera.
"Perhaps Americans find it difficult to understand this, but Greece lacks a **build-in** culture of stability and discipline"
Ένα χρόνο αργότερα στις 19/05/2011 ο Γιάννος Παπαντωνίου συνεχίζει τον **διασυρμό** της χώρας. Αυτή την φόρα μέσα στις Βρυξέλες όπου του ζητήθηκε η γνώμη του για την κρίση και δήλωσε τα ακόλουθα
" Europe North produce huge supplies and Europe South plus islands produce huge deficit in the external accounts. The transfer of excessive saving to the periphery created conditions for extensive bowering in both private and public sectors.
Now we have two divisions here:
In physically responsible countries like Spain extensive bowering was undertaking buy private sector leading to bubbles especially in housing which internal creating insolvensing problems for banks that where eventually assume by the governments
In physically profligate countries like Greece thinks are simpler. Extensive bowering has undertaking directive by the government leading directive to explosion of budget deficit and the debt.
However in both causes we have share a very interesting characteristic. Match of the debt induce by the saving gluts in core economies and the (in) top indirectly or directly on the government books of weaker economies" [1]

1. www.youtube.com (video στο cd της πτυχιακής (D:\πηγές))

Ακούγοντας αυτές τις δηλώσεις σε κάθε νέο τις ηλικίας μου δημιουργούνται τα ακόλουθα ερωτήματα:

- Έχουμε σαν λαός, κάποια γενετική (build-in) ανωμαλία, η οποία δεν μας επιτρέπει να οργανώσουμε την πατρίδα μας;
- Εάν ναι, πως εξηγείται η εδραίωση της Ελληνικής ομογένειας σε ξένους τόπους.
- Υπάρχει κοινωνική ισότητα μεταξύ των Ευρωπαίων πολιτών;
- Είναι η Ελλάδα ακάθαρτη χώρα;

Κατά την ταπεινή μου γνώμη η Ελλάδα δεν είναι ακάθαρτη ούτε οι Έλληνες έχουν γενετικό πρόβλημα. Θελημένα ή μη η οργάνωση του Δημόσιου τομέα της χώρα μας σκοντάφτει στο κενό που υπάρχει μεταξύ των μεθόδων οργάνωσης και στην πληροφορική σαν βραχίονα επιβολής της οργάνωσης.

Έχοντας τον παραπάνω προβληματισμό, αποφάσισα να αφιερώσω την πτυχιακή μου εργασία στην οργάνωση μιας μικρής επιχείρησης. Μικρή διότι ο χρόνος και τα μέσα τα οποία είχα ήταν περιορισμένα.

Έθεσα σαν στόχο της εργασίας μου

- Τον ανασχεδιασμό (reengineering) των διαδικασιών με την μέθοδο BMP.
- Την υλοποίηση των νέων διαδικασιών με χρήση Πληροφορικής (Information Technology).

2. ΕΙΣΑΓΩΓΗ

Μια επιχείρηση αποτελεί μια μορφή οργάνωσης προσώπων και μέσων που επιδιώκει την επίτευξη συγκεκριμένων στόχων. Για την επίτευξη των στόχων στηρίζεται σε διαδικασίες (procedures) που καθοδηγούν και προσανατολίζουν τις ενέργειες της.
Ως διαδικασία ορίζουμε ένα σύνολο ενεργειών που σκοπό έχουν την επίτευξη του επιθυμητού αποτελέσματος στο βέλτιστο χρόνο με την μέγιστη ασφάλεια που μπορούμε να επιτύχουμε.

Τον συντονισμό για

- δημιουργία νέων
- και ελέγχου των υπαρχουσών

διαδικασιών έχει ο εσωτερικός ελεγκτής (internal auditor).

Όμως λόγο:

- Της εξέλιξη της τεχνολογίας,
- Της έντασης του ανταγωνισμού που προήλθε από την παγκοσμιοποίηση

οι διαδικασίες γίνονταν μεγάλες και πολύπλοκες.

Όμως το μεγαλύτερο πρόβλημα είναι η ευελιξία. Ευελιξία στις άμεσες αλλαγές τόσο στον οικονομικό όσο και στο νομικό τομέα. Ως αποτέλεσμα αυτού οι ελεγκτές, οι λογιστές και οι financial managers στράφηκαν προς τους μηχανογράφους (engineers) για την δημιουργία ενός οργανωμένου συστήματος διαδικασιών όπου θα :

- Επεξεργάζεται μεγάλο όγκο πληροφοριών
- Γίνονται καταχωρίσεις και καταγραφές στοιχείων με μηδαμινά λάθη
- **Είναι ευέλικτο στις αλλαγές**
- διακινεί πληροφορίες με ασφάλεια
- Μειώνει δυνατότητα δόλιων ενεργειών από το προσωπικό

Το σύστημα αυτό είναι γνωστό σαν Business Process Management (BMP) και περιγράφεται στο παρακάτω κεφάλαιο.

3. Business Process Management

Το Business Process Management (B.P.M) είναι μια συστηματοποιημένη προσέγγιση που με την χρήση της τεχνολογίας επιτυγχάνει τη συνεχή βελτίωση της επιχειρηματικής αποδοτικότητας, των επιχειρηματικών αποτελεσμάτων, την ανάπτυξη και την ευελιξία της επιχείρησης.
Το B.P.M σαν θεωρία Management εμφανίστηκε σε πρώιμη μορφή το 1990 και το 2010 με την χρήση της τεχνολογίας πήρε την σημερινή του μορφή.
Το B.P.M. βλέπει της διαδικασίες μιας επιχείρησης σαν περιουσιακό στοιχείο.

Οι διαδικασίες αυτές πρέπει να :

- Κατανοηθούν
- Διαχειρισθούν
- Βελτιωθούν

ώστε η επιχείρηση να παραδώσει στους πελάτες της προϊόντα και υπηρεσίες υψηλής ποιότητας(αξίας).

Για να επιτύχει το B.P.M αυτό το αποτέλεσμα χωρίζει τις διεργασίες μια επιχείρησης σε διαδικασίες που εκτελούνται από συγκεκριμένες ενέργειες.
Το B.P.M συσχέτισε τις διαδικασίες και ενέργειες της με την IT (information technology) ώστε να:

- επιταχύνει το χρόνο διεκπεραιώσεις
- μεγιστοποίηση την επίβλεψη
- μηδενίσει το κίνδυνο των ανακολουθιών και παρεκκλίσεων των ενεργειών
- συσχετίσει τις διαδικασίες μεταξύ τους

Επόμενος το B.P.M πλέον υπερτερεί από το κλασικό ιεραρχικό Management και επιπλέον επιτρέπει στους χρήστες του να :

- οραματίζονται-σχεδιάζουν λειτουργίες και διαδικασίες
- καθορίζουν το στόχο της αλλαγής
- καθορίζουν το τι είναι επιτυχία
- αναλύουν και να συγκρίνουν εκδοχές-σχεδια των διεργασιών-ενεργειών
- να βελτιώσουν και να επιλέξουν την βελτίωση που θα εφαρμόσουν
- να παρακολουθούν σε πραγματικό χρόνο μέσο των χρηστών την αποτελεσματικότητα τον αλλαγών
- να επανασχεδιάσουν τις διαδικασίες από το μηδέν

Το μεγαλύτερο πλεονέκτημα του B.P.M είναι ο επανασχεδιασμός διότι προφέρει στο υπάρχων σύστημα των διαδικασιών ευελιξία. Η ευελιξία αποτελεί το κλειδί για την δημιουργία ενός σωστού συστήματος διαδικασιών αφού μια επιχείρηση είναι ένας ζωντανός οργανισμός που πρέπει να προσαρμόζεται στις αλλαγές του περιβάλλοντος της, να συμβαδίζει με την υπάρχουσα στρατηγική της επιχείρησης και τις ισχύουσες νομοθεσίες.

Το B.P.M δεν εστιάζει μόνο στην αυτοματοποίηση των διαδικασιών άλλα και την συμμετοχή του ανθρώπου ως χρήστη. Μιλάμε για ένα ανθρωποκεντρικό σύστημα όπου ο χρήστης παίρνει μέρος στην διαδικασία παράλληλα με την τεχνολογία και είναι υπεύθυνος για ένα συγκεκριμένο αριθμό διαδικασιών. Δίνεται η δυνατότητα διαχωρισμού των διαδικασιών από την τεχνολογία ώστε σε περιπτώσεις που κατά την διάρκεια αυτών παραστεί η επέμβαση της ανθρώπινης κρίσης και εμπειρίας, η διαδικασία να ολοκληρώνονται από τον άνθρωπο.

4. ΤΡΟΠΟΣ ΛΕΙΤΟΥΡΓΙΑΣ ΤΟΥ B.P.M

Το B.P.M χωρίζει τις δραστηριότητες του σε 7 κατηγορίες:

1. Ανάληψη του προβλήματος
2. Σύλληψη της ιδέας
3. Σχεδιασμός
4. Προτυποποίηση (Modeling)
5. Εκτέλεση
6. Παρακολούθηση
7. Βελτίωση - Επανασχεδιασμός

- **Ανάληψη του προβλήματος**
  Το πρώτο βήμα είναι η ανάληψη του προβλήματος από μια οντότητα. Η ανάληψη του προβλήματος γίνεται με την ανάθεση του στον engineer

- **Σύλληψη της ιδέας**
  Πρέπει να γίνει η επεξήγηση του προβλήματος στο engineer και το αποτέλεσμα που θέλει η οντότητα επιτύχει. Ο engineer έχοντας:
  1. κατανοήσει την φύση του προβλήματος
  2. τα απαραίτητα δεδομένα του προβλήματος

  μπορεί να συμπεράνει αν το πρόβλημα είναι επιλύσιμο και να ξεκινήσει το σχεδιασμό της λύσης.

- **Σχεδιασμός**
  Γίνεται η ανάλυση της υπάρχουσας κατάστασης, εντοπίζονται τα σφάλματα των διαδικασιών και πραγματοποιείται ο επανασχεδιασμός τους.

- **Προτυποποίηση (Modeling)**
  Η διαδικασία της προτυποποίηση αναλαμβάνει να υλοποιήσει τον σχεδιασμό του συστήματος με την χρήση των κατάλληλων εργαλείων και να δημιουργήσει ένα μοντέλο

- **Εκτέλεση**
  Σε αυτή την δραστηριότητα εκτελείται το υλοποιημένο μοντέλο με την χρήση των κατάλληλων λογισμικών εργαλείων (application, executable files κ.λπ.) και μεθόδων.

- **Παρακολούθηση**
  Η εφαρμογή τίθεται σε δοκιμαστική περίοδο όπου εκεί :
  Εντοπίζονται απρόοπτα λάθη, ασυμβατότητες
  Καταγράφονται οι παρατηρήσεις και υποδείξεις τον χρηστών
  Συλλέγονται τα πρώτα στοιχεία για την αξιολόγηση της εφαρμογής

- **Βελτίωση - Επανασχεδιασμός (Re-engineering)**
  Ύστερα από τον εντοπισμό των σφαλμάτων , την καταγραφή και συλλογή των απαραίτητων δεδομένων ξεκινάει η διαδικασία (επέρχεται) η διόρθωση και η βελτίωση της εφαρμογής.

Σαν αντικείμενο της παρούσας εργασίας είναι η χρήση της BMP για την οργάνωση της παρακάτω επιχείρησης.

5. Η ΕΠΙΧΕΙΡΗΣΗ

Το Ινστιτούτο Αισθητικής Figura Helena λειτουργεί από το 1983. Η έδρα της επιχείρησης είναι στην Ηλιούπολη Αττικής στην οδό Ηρώων Κωνσταντοπούλου 17. Στόχος των ιδιοκτητών είναι η προσφορά των ποιοτικότερων υπηρεσιών στις καλύτερες τιμές ανεξαρτήτως τον οικονομικών συγκύριων που επικρατούν.
Για να επιτύχει αυτόν τον στόχο το αισθητικό κέντρο:

- έχει επενδύσει συνολικά τουλάχιστον 500.000€ σε εξοπλισμό
- προσλαμβάνει άρτιο προσωπικό
- προμηθεύεται προϊόντα περιποίησης πιστοποιημένων κατασκευαστών
- διευρύνει τον επιχειρηματικό ορίζοντα της προσθέτοντας νέες υπηρεσίες.

Ως Ελληνική επιχείρηση δεν ξεχνά το χρέος της προς την ελληνική κοινωνία.
Προσφέρει κοινωνικό έργο μέσο:

- των προγραμμάτων πρακτικής άσκησης των εκπαιδευτικών ιδρυμάτων όπου προσλαμβάνει προσωπικό για την μερική στελέχωση του κέντρου προσφέροντας στους υποψήφιους αισθητικούς άρτια εκπαίδευση
- Διοργάνωσης ομιλιών και σεμιναρίων με θέμα την υγεία και την αισθητική
- Δημιουργία ιστοσελίδας με στόχο την επιστημονική ενημέρωση του κοινού σε θέματα Αισθητικής και Διατροφολογίας (www.figura.gr)

6. ΥΠΑΡΧΟΥΣΑ ΚΑΤΑΣΤΑΣΗ

Οι *Πελάτες* της επιχείρησης υπερβαίνουν τους 1.600, εκ των οποίων οι 200-500 την επισκέπτονται σταθερά, τουλάχιστον μια φορά το εξάμηνο
Μέχρι το 2011 το Figura Helena απασχολούσε 6 άτομα.

Λόγω της οικονομικής κρίσης, το 2011 αναγκάστηκε να προβεί σε μείωση του χώρου εγκαταστάσεως και του προσωπικού.
Σήμερα στο Figura Helena , εργάζονται εκτός από την **ιδρύτρια** Λένα Βλασερού, **η αισθητικός κοσμετολόγος** Μαρία Κυριάκη και η **κλινική διαιτολόγος** Ολυμπία Βλασερού.

Η εν λόγω επιχείρηση παρέχει στο καταναλωτικό κοινό:

- Περιποιήσεις προσώπου (ενυδάτωση, βαθία peeling, καθαρισμός, σύσφιξη-αντιγήρανση, κ.α.)
- Περιποιήσεις σώματος (μασάζ, ρεφλεξολογία, ηλεκτροθεραπεία, κ.α.)
- Αποτρίχωση (κερί, ενζυμική, ηλεκτρική, IPL κ.α.)
- Διαιτολογική υποστήριξη
- Λιανική πώληση καλλυντικών προϊόντων

Από τα ανωτέρω γίνεται σαφές ότι η επιχείρηση είναι πρωτίστως εντάσεως εργασίας και δευτερευόντως εντάσεως κεφαλαίου.

Στην προσπάθεια της για μείωση του κόστους διαχείρισης, η επιχείρηση έκανε αρκετές προσπάθειες μηχανοργάνωσης. Όλες οι προσπάθειες απέτυχαν με αποτέλεσμα η επιχείρηση να εξακολουθεί να τηρεί :

- Τα στοιχεία *Πελατών (τηλέφωνο, διεύθυνση, email, κ.α.)*
- *Το ιστορικό των πελατών (κατάσταση υγείας, προτιμήσεις, ανάγκες σε θεραπεία)*
- Τα *Ραντεβού* της ημέρας
- *Το Ιστορικό των Ραντεβού* του κάθε πελάτη

σε χειρόγραφο σύστημα.

Στη συνέχεια, προκειμένου να κρατηθεί αρχείο, γινόταν καταγραφή των παραπάνω στον Η/Υ στο πρόγραμμα Microsoft Excel, το οποίο φυσικά δεν επαρκούσε διότι:

- Δεν γινόταν συσχέτιση *Πελάτη – Ραντεβού*
- Παρουσίαση *Ιστορικού τον Ραντεβού*
- Ασφαλή διατήρηση των στοιχείων του πελάτη (διπλες εγγραφές, λάθος καταχωρήσεις)
- Απουσία καρτέλα *Πελάτη*

Από τα παραπάνω φαίνεται ότι υπάρχει ανάγκη ύπαρξης γραμματειακής υποστήριξης ή περισσότερο χρόνο από το υπάρχον μειωμένο προσωπικό (εφόσον δεν υπάρχει πλέον γραμματεία).
Ο λόγος της αποτυχίας φαίνεται να είναι η χρήση έτοιμων πακέτων λογισμικού τα οποία επέβαλλαν την οργάνωση της επιχείρησης πάνω σε πρότυπα επιχειρήσεων εντάσεως κεφαλαίου (εμπορικές).
Και για να το εκφράσω πιο επιστημονικά «έβαζαν το κάρο μπροστά από το μουλάρι».

Η επιχείρηση βρέθηκε σε ένα διαχειριστικό αδιέξοδο. Η διαδικασίες που ακολουθούσε εκτός ότι ήταν από την φύση τους λανθασμένες αφού δεν είχαν στηριχτεί σε κάποιο επικυρωμένο πρότυπο οργανωμένων διαδικασιών ήταν και αδύνατο να λειτουργήσουν με το υπάρχων διαθέσιμο προσωπικό.

7. Η ΛΥΣΗ ΣΤΟ ΠΡΟΒΛΗΜΑ

Η λύση στο πρόβλημα είναι απλή. Στην εταιρεία πρέπει να δημιουργηθεί ένα σύστημα διαδικασιών όπου θα:

- Υλοποιηθεί με μικρό κεφάλαιο
- Λειτουργήσει με μικρό αριθμό προσωπικού
- Είναι εύκολη η υλοποίηση των διαδικασιών
- Αντικαταστήσει την γραμματεία
- Είναι ακέραιο, ασφαλές και χωρίς σφάλματα

Το σύστημα Figura Helena θα αναλαμβάνει:

- Το κλείσιμο ραντεβού
- Την παρουσίαση των ραντεβού
- Την επεξεργασία των ραντεβού
- Την δημιουργία βάσης πελατών
- Την δημιουργία καρτέλας πελάτη
- Την επεξεργασία των στοιχείων του πελάτη

8. ΠΡΟΔΙΑΓΡΑΦΕΣ

Το σύστημα το οποίο υλοποιήσαμε σχεδιάστηκε να έχει τις παρακάτω προδιαγραφές.

Περιβάλλον χρήστη	Internet browser συμβατό με HTML4 ή HTML5
Δυνατότητα πολυγλώσσου	Υποστήριξη μέχρι 6 γλωσσών
Δυνατότητα Multiuser	Υποστήριξη session mamagement
Ανάπτυξη εφαρμογής	J2EE (http://www.webopedia.com/TERM/J/J2EE.html)
Οργάνωση δεδομένων	Σχεσιακή βάση πληροφοριών (RDBMS)

HTML: είναι μια γλώσσα προγραμματισμού που χρησιμοποιείται ώστε οι περιηγητές (Internet Explorer, Opera, Firefox κ.τ.λ.) να πραγματοποιήσουν δόμηση σελίδων WEB.

Multiuser: ονομάζεται μία εφαρμογή, όταν έχει την δυνατότητα να χρησιμοποιείται από πολλούς χρήστες παράλληλα, χωρίς οι ενέργειες του ενός, να παρεμποδίζουν τις ενέργειες του άλλου.

J2EE: είναι βιβλιοθήκη της java η οποία υλοποιεί εφαρμογές αρχιτεκτονικής 3 επιπέδων (3-tier). Η Αρχιτεκτονική που θα χρησιμοποιήσουμε είναι οι Servlets.

Servlets: είναι java προγράμματα τα οποία υλοποιούν αμφίδρομη επικοινωνία μεταξύ

- Περιηγητή (web browser)
- και application server.

Η υλοποίηση του συστήματος θα γίνει στις παρακάτω φάσεις

Φάση	Στόχος
1η	• Σχεδιασμός • Υλοποίηση • Λειτουργία • Παρακολούθηση του συστήματος σύμφωνα με τις τεθείσες προδιαγραφές
2η	Βελτίωση της εφαρμογής Δυνατότητα του πελάτη να • κλείσει Ραντεβού • δει την καρτέλα του από • το Internet • Android app

Στα πλαίσια της παρούσης εργασίας έχει υλοποιηθεί η 1η φάση ενώ έχουν τεθεί οι βάσεις για την 2η.

9. ΤΕΧΝΙΚΟΙ ΟΡΙΣΜΟΙ

Εφαρμογή (app)	Είναι η διαδικασία ανταλλαγής πληροφορίας μεταξύ ανθρώπου και υπολογιστή.
Χρήστης (user)	Ο άνθρωπος ο οποίος χρησιμοποιεί μια Εφαρμογή.
Διαδικασία (Process)	Η διακριτή λειτουργία μιας επιχείρησης η οποία έχει • ένα στόχο • αρχή, μέση και τέλος Μια διαδικασία υλοποιείται από ένα ή περισσότερα βήματα.
Βήμα Διαδικασίας (step)	Είναι η διαδραστική λειτουργία κατά την οποία 1. η Εφαρμογή ζητά από τον Χρήστη πληροφορία 2. Ο χρήστης δίνει την πληροφορία στην Εφαρμογή
Session	Η εκτέλεση μίας **Διαδικασίας** από ένα **Χρήστη**. Η διαδικασία αρχίζει και τελειώνει μέσα σε ένα παράθυρο (Window) ενός υπολογιστή.
Βάση Πληροφοριών	Είναι η αποθήκη πληροφοριών μιας Εφαρμογής. Στόχος της βάσης είναι η διατήρηση των πληροφοριών ακόμη και όταν ο υπολογιστής είναι εκτός τάσης.

10. ΔΙΑΔΙΚΑΣΙΕΣ

Ο χρήστης με την χρήση της εφαρμογής θα ακολουθεί της παρακάτω διαδικασίες όπου θα παρουσιάζονται μέσο ενός interface βηματιστή.
Οι διαδικασίες παρουσιάζονται αναλυτικά στον παρακάτω πίνακα

	Όνομα	Λειτουργία	Στόχος
1	Νέο άτομο	Μεταβολή	Η καταχώρηση των προσωπικών στοιχείων ενός νέου Ατόμου στην *Βάση* πληροφοριών
2	Αλλαγή στοιχείων ατόμου	Μεταβολή	Η ενημέρωση της *Βάσης* με τα νέα στοιχεία ενός Ατόμου
3	Διαγραφή ατόμου	Μεταβολή	Η διαγραφή όλων των στοιχείων τα οποία αφορούν συγκεκριμένο άτομο από την *Βάση*
4	Σχόλιο για άτομο	Μεταβολή	Η καταχώρηση ενός σχολίου το οποίο αφορά συγκεκριμένο άτομο
5	Διαγραφή σχολίου	Μεταβολή	Η διαγραφή συγκεκριμένου σχολίου από συγκεκριμένο άτομο
6	Νέο Ραντεβού	Μεταβολή	Η δημιουργία ενός νέου ραντεβού για συγκεκριμένο άτομο
7	Διαγραφή Ραντεβού	Μεταβολή	Η Διαγραφή συγκεκριμένου ραντεβού από συγκεκριμένο άτομο
8	Απολογισμός Ραντεβού	Μεταβολή	Η καταχώρηση οικονομικών στοιχείων τα οποία αφορούν συγκεκριμένο ραντεβού
9	Παρουσίαση ατόμου	Ενημέρωση	Η Παρουσίαση των 1. Προσωπικών στοιχείων 2. Ραντεβού 3. Σχολίων τα οποία αφορούν συγκεκριμένο άτομο
10	Παρουσίαση Ραντεβού	Ενημέρωση	Η παρουσίαση όλων των ραντεβού
11	Ραντεβού ανά ημέρα	Ενημέρωση	Η παρουσίαση όλων των ραντεβού τα οποία αφορούν συγκεκριμένη ημέρα
12	Ραντεβού ανά υπάλληλο	Ενημέρωση	Η παρουσίαση όλων των ραντεβού τα οποία αφορούν συγκεκριμένο υπάλληλο
13	Κλείσιμο ημέρας	Ενημέρωση	Η παρουσίαση των οικονομικών στοιχείων για όλα τα ραντεβού συγκεκριμένης ημέρας
14	Πολλαπλά κινητά τηλέφωνα	Ενημέρωση	Η παρουσίαση όλων των ατόμων τα οποία έχουν το ίδιο κινητό με άλλα άτομα (πιθανή διπλή καταχώρηση)
15	Λάθος κινητά τηλέφωνα	Ενημέρωση	Η παρουσίαση όλων των ατόμων των οποίων τα κινητά έχουν καταχωρηθεί λάθος.

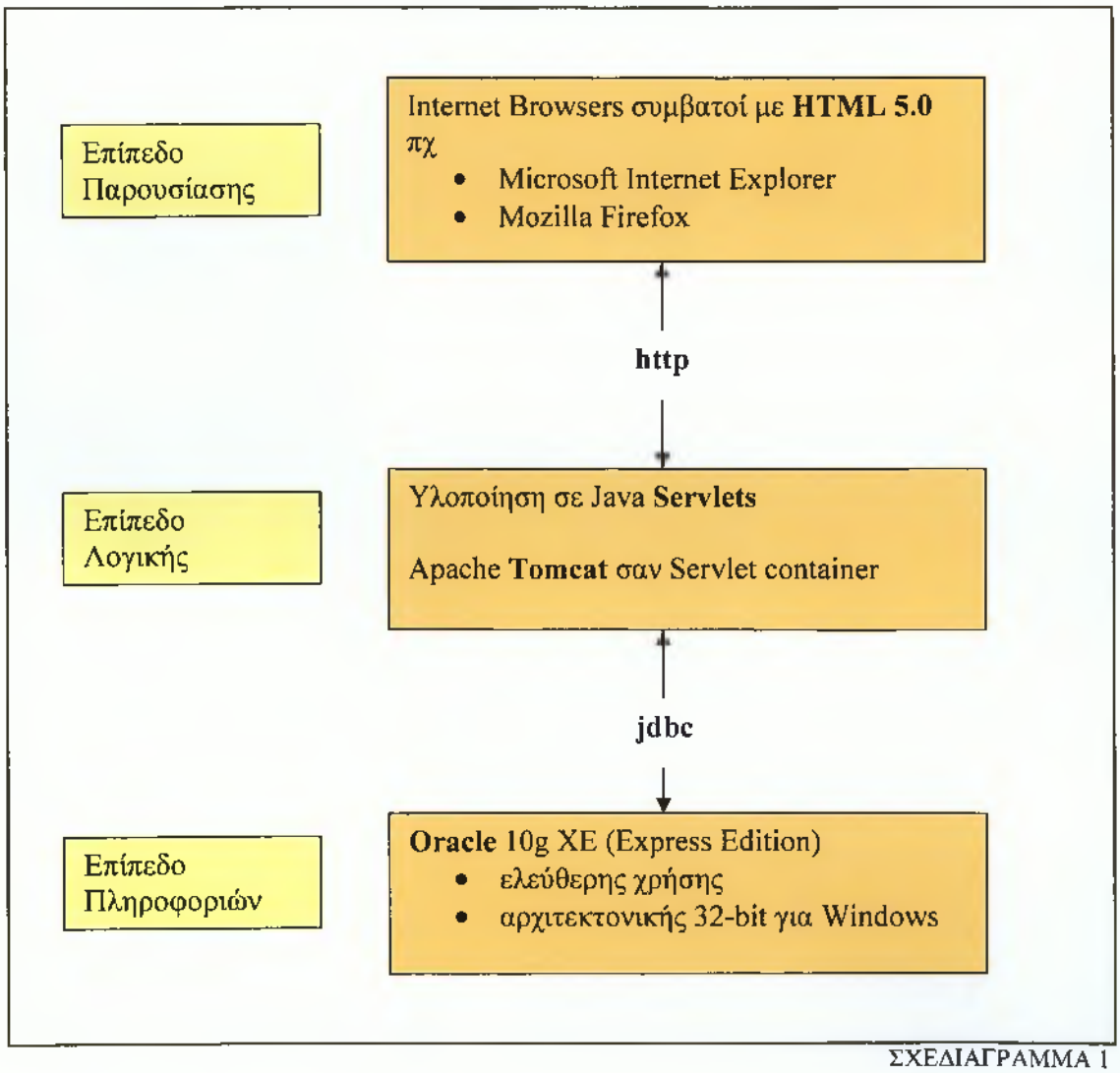
ΠΙΝΑΚΑΣ 1

11. ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ Figura Helena

Το Σύστημα Figura ακολουθεί την αρχιτεκτονική των **3 επιπέδων** (3 tier Architecture) όπως αυτή περιγράφεται στο http://en.wikipedia.org/wiki/Multitier_architecture

Στο σχήμα που ακολουθεί φαίνεται η υλοποίηση της εν λόγω αρχιτεκτονικής όσον αφορά

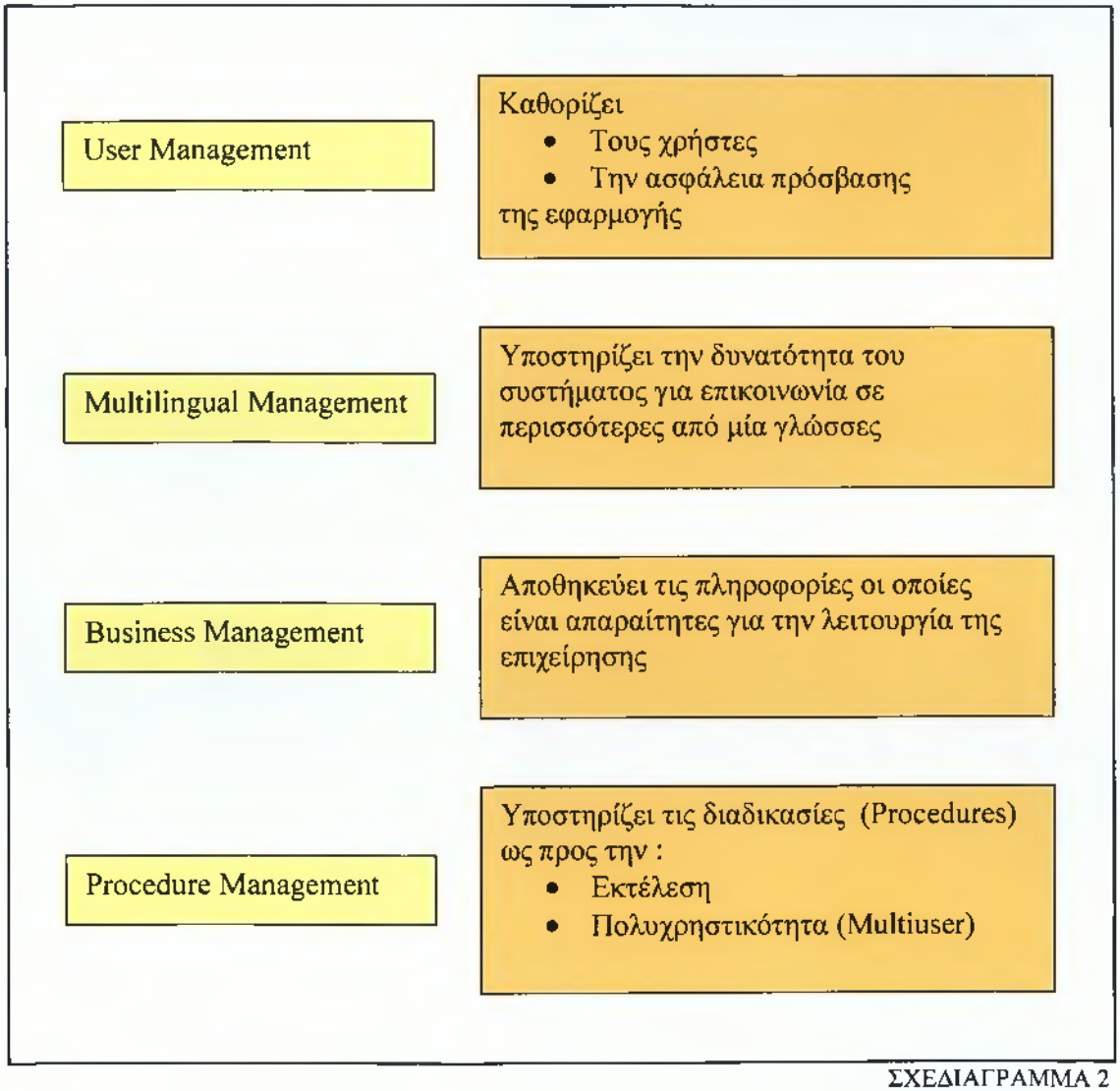
- Το κάθε επίπεδο
- Τα πρωτόκολλα επικοινωνίας μεταξύ των επιπέδων



ΣΧΕΔΙΑΓΡΑΜΜΑ 1

12. ΣΧΕΔΙΑΣΜΟΣ ΒΑΣΗ ΠΛΗΡΟΦΟΡΙΩΝ

Στο σχήμα που ακολουθεί φαίνονται τα υποσυστήματα της βάσης πληροφοριών καθώς και η λειτουργία του κάθε υποσυστήματος.



ΣΧΕΔΙΑΓΡΑΜΜΑ 2

Τα δεδομένα της εφαρμογής ανήκουν στις παρακάτω κατηγορίες

Σταθερά δεδομένα	Σαν σταθερά ορίζονται τα δεδομένα τα οποία απαιτούνται για την χρήση της εφαρμογής, αλλά δεν μπορούν να μεταβληθούν από τους χρήστες της.
Μεταβλητά δεδομένα	Σαν μεταβλητά δεδομένα ορίζονται τα δεδομένα τα οποία μεταβάλουν οι χρήστες μέσω της εφαρμογής.

Εν συνεχεία περιγράφονται τα υποσυστήματα της εφαρμογής

Υποσύστημα User Management (Διαχείρηση χρηστών)

Το προσωπικό της επιχείρησης μπορεί να είναι:

- Χρήστες της εφαρμογής
- Πάροχοι των υπηρεσιών της επιχείρησης προς τους πελάτες.
- Και τα δύο

Το υποσύστημα αυτό αποτελείται από ένα αρχείο το οποίο καθορίζει

- Τις παραμέτρους πρόσβασης (login και password) κάθε χρήστη
- Τα στοιχεία του χρήστη (Όνομα, Επώνυμο)
- Την ιδιότητα του χρήστη

	Όνομα Αρχείου	Εγγραφές	Περιγραφή
1	CFG_USR	4	Χρήστες εφαρμογής

ΠΙΝΑΚΑΣ 1.1

Υποσύστημα Multilingual Management

Το υποσύστημα αυτό αποτελείται από ένα αρχείο το οποίο καθορίζει την περιγραφή κάθε ετικέτας την οποία χρησιμοποιεί η εφαρμογή.
Το αρχείο έχει δυνατότητα υποστήριξης μέχρι 6 γλωσσών.

	Όνομα Αρχείου	Εγγραφές	Περιγραφή
1	CFG_LBL	180	Ετικέτες εφαρμογής

ΠΙΝΑΚΑΣ 1.2

Υποσύστημα Business Management

Το εν λόγω υποσύστημα αποθηκεύει και διαχειρίζεται τα δεδομένα το όποια αφορούν αμιγώς την λειτουργία της επιχείρησης.

Σταθερά δεδομένα

Στον πίνακα που ακολουθεί φαίνονται τα αρχεία των σταθερών δεδομένων.

	Όνομα Αρχείου	Εγγραφές	Περιγραφή
1	OBJ_FKEYS_L01	2	Φύλο πελατών (Θηλυκό, Αρσενικό)
2	OBJ_FKEYS_L15	617	Μικρά ονόματα Πελατών
3	OBJ_FKEYS_L04	1213	Ταχυδρομικοί Κωδικοί
4	OBJ_FKEYS_L02	11	Ηλικιακή ομάδα πελατών
5	TRT_0	7	Ομάδες Υπηρεσιών
6	TRT_1	21	Υπηρεσίες (Προϊόντα)

ΠΙΝΑΚΑΣ 1.3

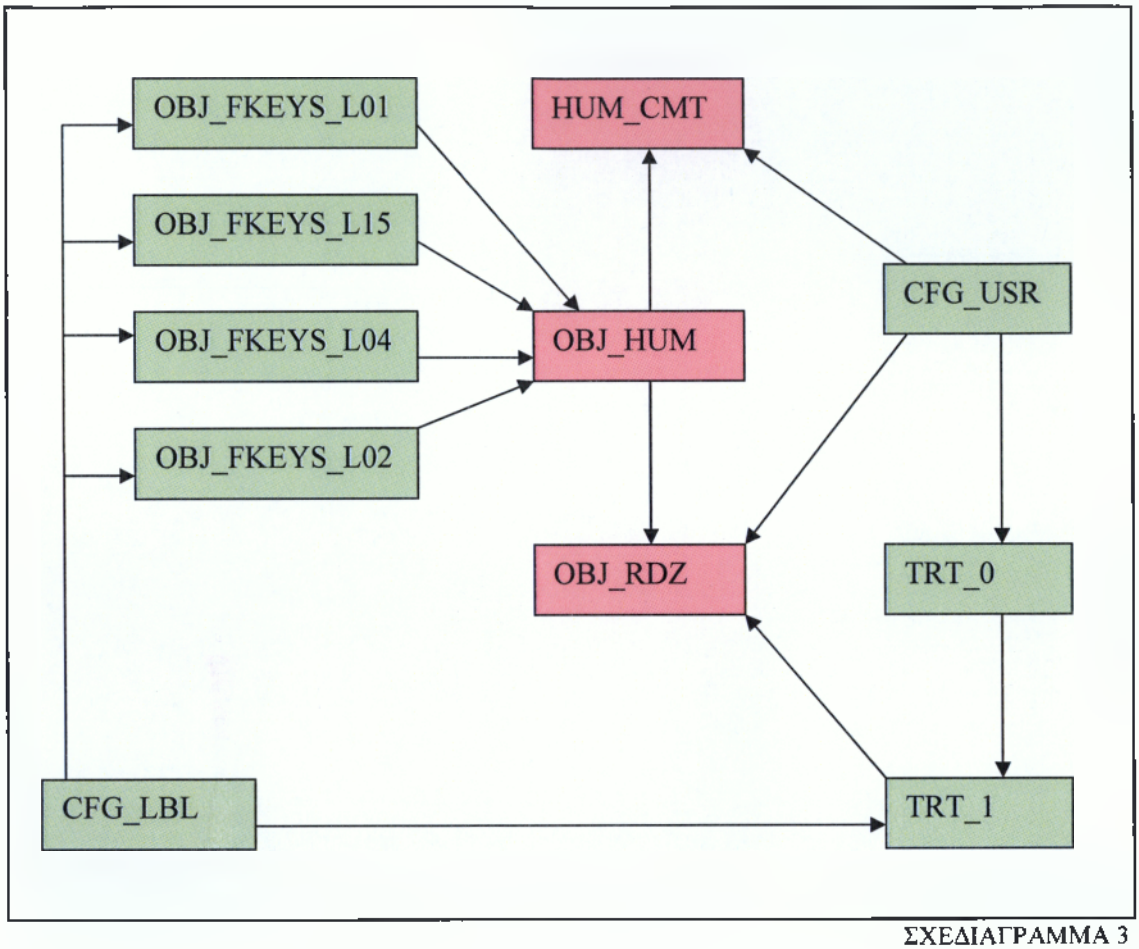
Μεταβλητά δεδομένα

Στον πίνακα που ακολουθεί φαίνονται αρχεία μεταβλητών δεδομένων του υποσυστήματος

	Όνομα Αρχείου	Περιγραφή
1	OBJ_HUM	Φυσικά πρόσωπα (πελάτες)
2	OBJ_RDZ	Ραντεβού (Rendezvous) πελατών για θεραπεία
3	HUM_CMT	Σχόλια που αφορούν τα Φυσικά πρόσωπα

ΠΙΝΑΚΑΣ 1.4

Σχήμα Υποσυστήματος (Database schema)



ΣΧΕΔΙΑΓΡΑΜΜΑ 3

Υποσύστημα Διαδικασιών (Procedure Management)

Το εν λόγω υποσύστημα αποθηκεύει τα στοιχεία που αφορούν την

- Τήρηση των διαδικασιών της επιχείρησης.
- Διαχείριση πολλαπλών χρηστών (Multiuser)

Το υποσύστημα αποτελείται από τα ακόλουθα αρχεία

Σταθερά δεδομένα
Στον πίνακα που ακολουθεί φαίνονται τα αρχεία των σταθερών δεδομένων.

	Όνομα Αρχείου	Εγγραφές	Περιγραφή
1	PRC_0	4	Ομαδοποίηση διαδικασιών (Menu)
2	PRC_1	15	Διαδικασίες
3	PRC_2	46	Βήματα Διαδικασίας
4	PRC_USERS	60	Σχέση Χρηστών με Διαδικασίες Καθορίζει τις Διαδικασίες τις οποίες μπορεί να χρησιμοποιήσει ο κάθε Χρήστης

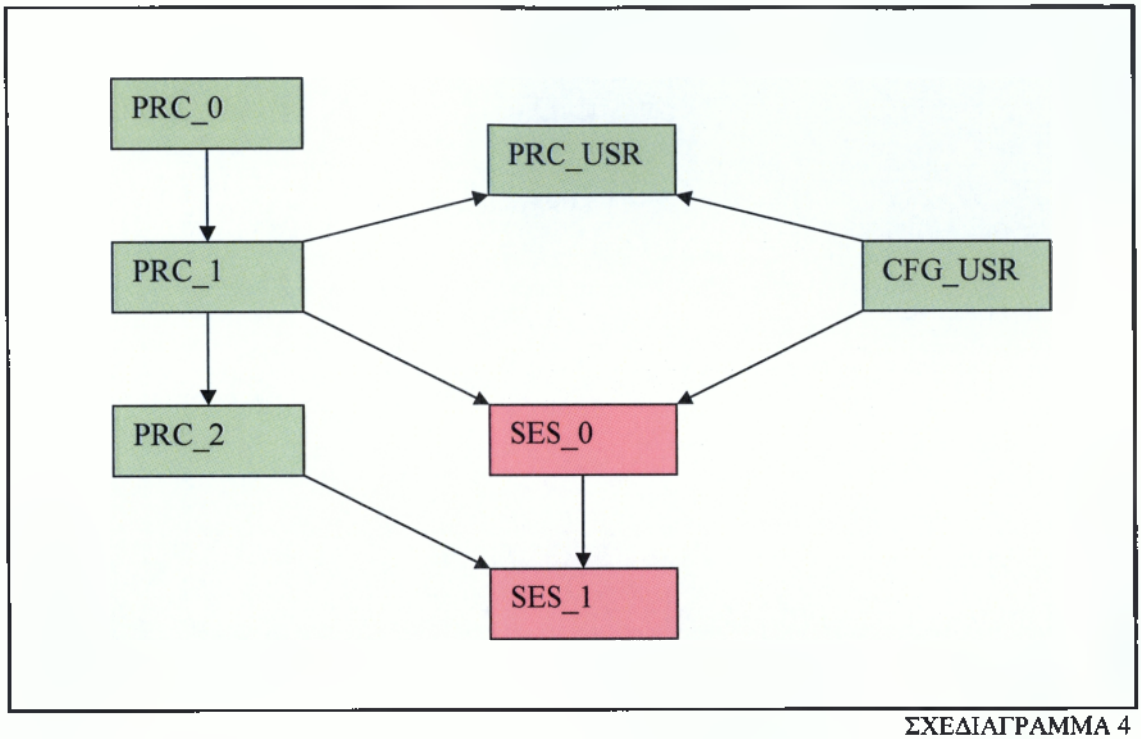
ΠΙΝΑΚΑΣ 1.5

Μεταβλητά δεδομένα
Στον πίνακα που ακολουθεί φαίνονται αρχεία μεταβλητών δεδομένων του υποσυστήματος και αφορούν το Session Management.

	Όνομα Αρχείου	Περιγραφή
1	SES_0	Session
2	SES_1	Βήματα του Session

ΠΙΝΑΚΑΣ 1.6

Σχήμα υποσυστήματος



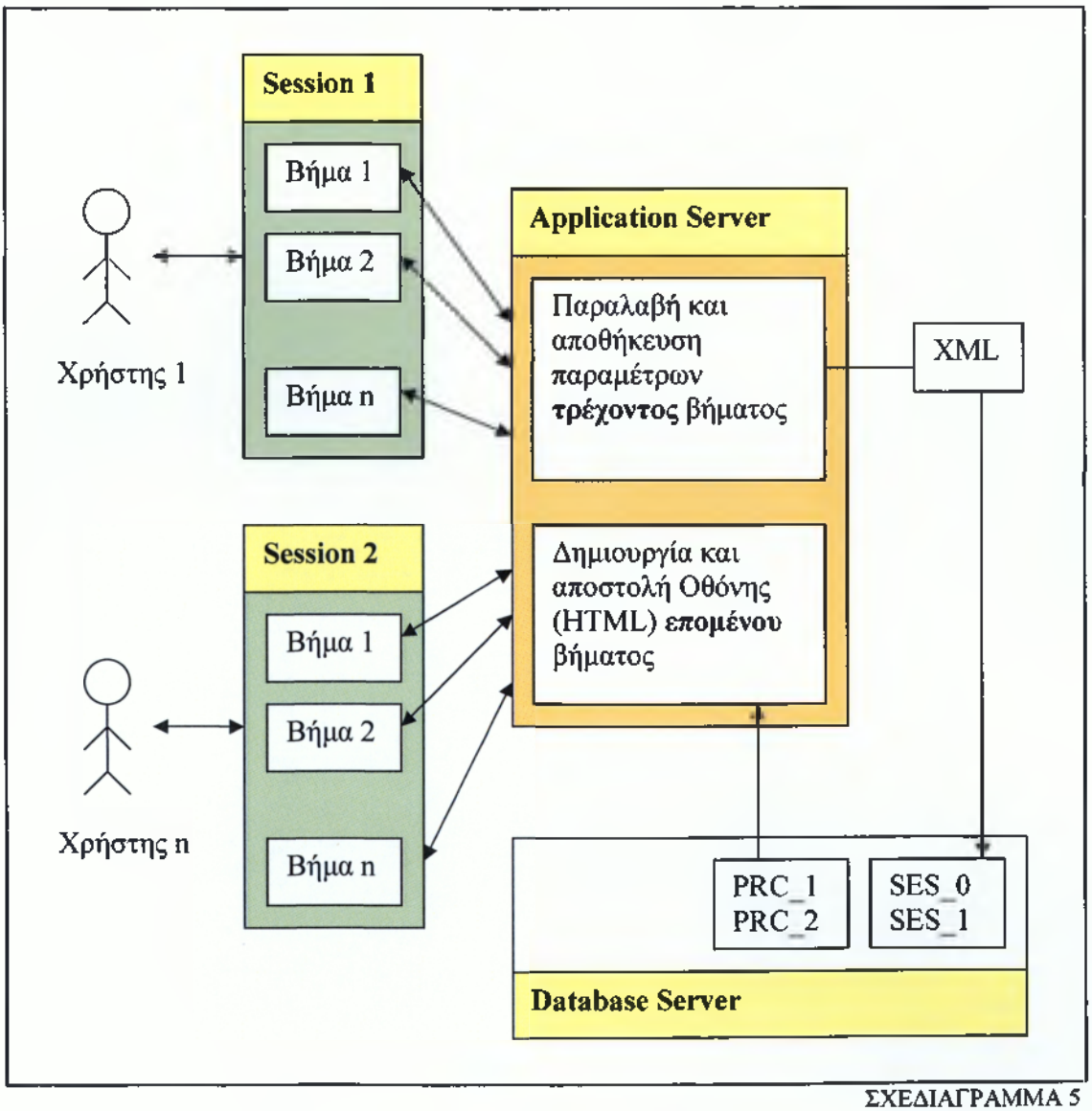
ΣΧΕΔΙΑΓΡΑΜΜΑ 4

Για λόγους παρακολούθησης της ροής το Session XML κάθε βήματος γράφεται σε αρχείο στο C:\fotis\TEMP

13. Session Management

Στο http://en.wikipedia.org/wiki/Session_(computer_science) δίνεται αναλυτική περιγραφή του Session.
Το Internet (πρωτόκολλο http) δεν τηρεί αφ' εαυτού session (stateless) και συνεπώς η τήρηση του session επαφίεται στην εφαρμογή.

Στο σχήμα που ακολουθεί φαίνεται η υλοποίηση του session όπως αυτή έγινε στην εφαρμογή μας.



ΣΧΕΔΙΑΓΡΑΜΜΑ 5

14.ΤΑ ΠΡΟΒΛΗΜΑΤΑ

Κατά την υλοποίηση της εφαρμογής έπρεπε να μοναδικοποιήσω τους πελάτες στην βάση δεδομένων. Η επίτευξη αυτού γίνεται με το εργαλείο της *oracle* το *primary key* (πρωτεύον κλειδί). Παράδειγμα πρωτεύοντος κλειδιού είναι το Α.Φ.Μ το Α.Μ.Κ.Α. Το Α.Φ.Μ και το Α.Μ.Κ.Α είναι μοναδικά για κάθε έλληνα πολίτη.
Ποιο θα ήταν όμως το πρωτεύον κλειδί για τους πελάτες του Figura Helena;

Η πρώτες μου σκέψης ήταν να οριστεί ένα από τα παρακάτω:

- Ηλικία
- Τηλέφωνο

Ύστερα από συζήτηση με την Ολυμπία Βλασερού μου ανάφερε τις παρακάτω ιδιαιτερότητες των πελατών του κέντρου:

- Ορισμένη πελάτες θέλουν να μένουν ανώνυμοι
- Δεν είναι εφικτό να ερωτάτε η ηλικία
- Δεν είναι πάντα εφικτή η καταχώρηση τηλεφώνου

Η λύση που δόθηκε είναι ότι με την καταγραφή κάθε πελάτη θα δίνετε σε αυτόν ένας **σειριακός αριθμός** από την εφαρμογή.

Κατά την παρακολούθηση

Με την λύση του παραπάνω προβλήματος και μετά την υλοποίηση της εφαρμογής έπρεπε να επέλθει το στάδιο της παρακολούθησής της.

Κατά την διάρκεια της λειτουργίας της εμφανίστηκαν :

- Προβλήματα συμβατότητας με διαφόρους περιηγητές
- Συμβατότητα ελληνικών χαρακτήρων
- Εμφάνιση Exceptions “Sorry we have a technical problem”

Όπου τελικός και αυτά λύθηκαν.

15.ΤΟ ΑΠΟΤΕΛΕΣΜΑ

Στο αισθητικό κέντρο λειτουργεί επιτυχώς η εφαρμογή με όλες τις προδιαγραφές και διαδικασίες που τέθηκαν στην αρχή της ανάληψης του έργου.
Πλέον οι εργαζόμενοι χρησιμοποιούν την εφαρμογή για το κλείσιμο των ραντεβού.

Κατά την διάρκεια του κλεισίματος ραντεβού δια του τηλεφώνου η εφαρμογή καθοδήγει τον χρήστη ώστε να:

- Καταγράφει σωστά τα στοιχεία του πελάτη
- Εντάξει όλα τα απαραίτητα στοιχεία στην καρτέλα του πελάτη

Το κλείσιμο των ραντεβού δεν απαιτεί την φυσική υπάρξει του εργαζόμενου στο χώρο της γραμματείας.
Με την χρήση ασύρματου τηλεφώνου και συσκευής με πρόσβαση στο internet μέσο ενός περιηγητή είναι δυνατόν να γίνει το κλείσιμο του ραντεβού σε οποιοδήποτε χώρο του αισθητικού κέντρου(αίθουσα μασάζ, αισθητικής περιποίησης κτλ)

Το μεγαλύτερο επίτευγμά της εφαρμογής είναι ότι είναι συμβατή με όλους τους εμπορικούς περιηγητές

Έχοντας δημιουργήσει μια σωστή βάση δεδομένων και ευέλικτες διαδικασίες οι εφαρμογή θα μπορέσει να εξελίχθη και να παρέχει περισσότερες υπηρεσίες στους χρήστες

16. The next step

Μετά από την λειτουργία ενός χρόνου και την συλλογή των κατάλληλων στοιχείων και τεχνογνωσίας η εφαρμογή θα:

- **Αναβαθμιστεί**
- **Επανασχεδιαστεί (reengineer)**

Οι νέοι στόχοι είναι:

- Μηχανογραφική τήρηση του ταμείου
- Μηχανογραφική τήρηση της αποθήκης
- Ενσωμάτωση των τηλεφωνικών κλήσεων μέσα στην εφαρμογή
- Ενίσχυση της διαφήμισης μέσο χρήσης sms
- Δημιουργία application για κλείσιμο του ραντεβού από τον πελάτη
- Βελτίωση του γραφικού περιβάλλοντος τα εφαρμογής

- **Μηχανογραφημένη τήρηση του ταμείου.**
  Η διαδικασία ολοκλήρωσης του ραντεβού συνεπάγεται την είσπραξη αμοιβής. Το άθροισμα των αμοιβών αποτελεί το ταμείο. Στο τέλος της ημέρας έχουμε το κλείσιμο του ταμείου.

- **Μηχανογραφημένη τήρηση της αποθήκης.**
  Η επιχείρηση χρησιμοποιεί υλικά (καλλυντικά κλπ). Τα υλικά πωλούνται ως έχουν ή καταναλώνονται κατά το ραντεβού. Η τήρηση αποθήκης θα βοηθήσει στον έλεγχο του κόστους.

- **Κοστολόγηση**
  Ο υπολογισμός του κόστους ανά ραντεβού είναι σημαντικός στην αποτελεσματική διοίκηση της επιχείρησης.

- **Ενσωμάτωση των τηλεφωνικών κλήσεων μέσα στην εφαρμογή**
  Στο καταχωρημένο τηλέφωνο του πελάτη θα υπάρχει η δυνατότητα κλήσης του αριθμού μέσα από την εφαρμογή.

- **Ενίσχυση της διαφήμισης μέσο χρήσης sms**
  Θα υπάρχει η δυνατότητα μαζικής αποστολής sms στους πελάτες του αισθητικού κέντρου για:
  1. Προσφορές
  2. Ημερίδες
  3. Ομιλίες

  με στόχο την διαφήμιση της επιχείρησης.

- **Δημιουργία application για κλείσιμο του ραντεβού από τον πελάτη**

  Ο πελάτης μέσο μιας android εφαρμογής θα μπορεί να κλείνει μόνος του ραντεβού και να ενημερώνεται για τις διαθέσιμες ώρες που υπάρχουν. Έτσι αποδεσμεύεται χρόνος από το προσωπικό για γραμματειακή υποστήριξη.

- **Βελτίωση του γραφικού περιβάλλοντος τα εφαρμογής**

  Η εφαρμογή θα διαμορφωθεί γραφιστικά σύμφωνα με τις απαιτήσεις των χρηστών.

17.ΕΠΙΛΟΓΟΣ

Θα ήθελα να ευχαριστήσω **όλους** τους ανθρώπους που με βοήθησαν ώστε να δημιουργηθεί αυτή η εργασία. Την οικογένεια μου που κατάφερε να με βοηθήσει να ολοκληρώσω τον κύκλο των σπουδών μου.
Στην μνήμη του φίλου μου Χρίστου Βογιατζή που από μικρό με ένταξε στο χώρο της πληροφορικής και να την εναποθέσω στις επόμενες γενιές των φοιτητών του ΑΤΕΙ ΠΕΛΟΠΟΝΗΣΣΟΥ.

Με παρούσα εργασία διέψευσα τους Έλληνες οικονομολόγους και πολιτικούς που ισχυρίζονται πως ο λαός μας έχει γενετικές ανωμαλίες.

Ο Ελληνικός λαός δεν έχει καμία γενετική ανωμαλία. Μπορεί να οργάνωση την χώρα του και τις επιχειρήσεις του.
Αφού επετεύχθη από ένα τελειόφοιτο σπουδαστή του ΑΤΕΙ ΠΕΛΟΠΟΝΗΣΣΟΥ η οργάνωση μιας επιχείρησης οι υπόλοιποι γιατί να μην μπορούμε; Όχι μόνο μπορούμε και έχουμε την ικανότατα να το πράξουμε αλλά μπορούμε να γίνουμε ανταγωνιστική και να εξυγιάνουμε τον Δημόσιο τομέα.

Η Ελλάδα **δεν είναι μια ακάθαρτη χώρα** είναι ισότιμο μέλος της Ευρωπαϊκής Ένωσης.

ΚΩΔΙΚΑΣ

```
/*-- Authentication Servlet ----------*/
/*-- Menu step 1. Receives Login and Password --*/
/*-------------------------------------------*/
import libs.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import org.w3c.dom.*;

public class M1Servlet extends HttpServlet
{
 public void service(HttpServletRequest req,HttpServletResponse res)
 {
  String    rn  = "M1Servlet";
  Connection conn = null;
  try
  {
   String eip = "";  //-- External Host IP
   String cip = "";  //-- Client IP
   String nam = "";
   String pwd = "";
   int    lng = Gen.atoi((String)req.getParameter("LNG")); if(Gen.isEmpty(lng)) lng=1;

   try
   {
    nam = (String)req.getParameter("NAM");
    pwd = (String)req.getParameter("PAS");
    eip = (String)req.getHeader("host");   //-- The host IP  (from which we got the
request)
    cip = (String)req.getRemoteAddr();     //-- Client IP
   }
   catch(Exception n) {}

      if(Gen.isEmpty(nam)) getout(res,lng,140,141);
   else if(!Gen.isWORD(nam)) getout(res,lng,140,142);
   else if(Gen.isEmpty(pwd)) getout(res,lng,140,143);
   else if(!Gen.isWORD(pwd)) getout(res,lng,140,144);
   else
   {
    conn = Gen.doConnect();

    int     uid = -1;           //- user id
    String  unm = "";           //- user name
    String[] rr  = getUID(conn,nam);
```

```
    if(Gen.isEmpty(rr))
     getout(res,lng,320,-1);
    else
    {
        if(Gen.atoi(rr[2])==1)                    getout(res,lng,322,-1);
     else if(Gen.isEmpty(rr[1]))                   getout(res,lng,322,-1);
     else if(!rr[1].toUpperCase().equals(pwd.toUpperCase())) getout(res,lng,321,-1);
     else
     {
      uid = Gen.atoi(rr[0]);
      unm = rr[3];
     }
    }

    if(!Gen.isEmpty(uid))
    {
     HtmlForm frm = new HtmlForm(1);
     frm.LNG = lng;
     frm.EIP = eip;

     frm.FRV[0]   = "";
     frm.FRV[1]   = "M2";
     frm.FRV[4]   = unm;
     frm.FRV[15]  = APL.get_LOGO();
     frm.PRM[0][0] = "MNU";
     frm.PRM[0][1] = getPRS(conn,uid,lng);
     frm.PRM[0][4] = Gen.fix_HIDDEN("UID",uid)
                + Gen.fix_HIDDEN("LNG",lng)
                + Gen.fix_HIDDEN("EIP",eip)
                + Gen.fix_HIDDEN("CIP",cip)
                ;
     frm.PRM[0][8] = fixJS();
     frm.PRM[0][11]= "Main_Menu";

     String prs = new Presenter(frm).getPRS();

     Gen.Transmit(res,prs,rn);
    }
   }
  }
  catch(Exception e) { Gen.err(res,null,rn,e.toString()); }
  finally { Gen.doDisconnect(conn); }
}

/*-- getPRS --------------------*/
/*-- Returns Presentable part --*/
```

```
/*----------------------------*/
private static String getPRS( Connection CONN
                    , int      UID_IN
                    , int      LNG_IN)
throws Exception
{
 String rn   = "getPRS";
 String outp = "";
 try
 {
  String    prs = "";
  String    qry = "select MNU_ID,LABEL_ID from PRC_0"
              + " where MNU_ID in "
              + "("
              + "select PRC_0.MNU_ID a"
              + " from PRC_0"
              +    ",PRC_1"
              +    ",PRC_USERS"
              + " where PRC_0.MNU_ID=PRC_1.MNU_ID"
              +  " and PRC_1.PRC_ID=PRC_USERS.PRC_ID"
              +  " and PRC_1.IS_BLOCKED=0"
              +  " and PRC_USERS.USR_ID="+UID_IN
              + " group by PRC_0.MNU_ID"
              + ")"
              + " order by ORD_ID"
              ;
  String[][] rr  = Gen.Qry2Array(CONN,qry);
  int       len = Gen.getLEN(rr);
  for(int j=0;j<len;j++)
  {
   int    mnu = Gen.atoi(rr[j][0]);
   String nam = Gen.get_LABEL(CONN,Gen.atoi(rr[j][1]),LNG_IN);
   prs += getLIN(mnu,nam,(j==0));
  }

  outp += "<TABLE>";
  outp += prs;
  outp += "</TABLE>";
 }
 catch(Exception e) { throw new DBException(rn,e.toString()); }
 return(outp);
}

/*-- get_LIN ---------*/
/*-- Returns a Line --*/
/*--------------------*/
```

```
private static String getLIN( int    PRC_IN
                    , String  NAM_IN
                    , boolean FTM_IN) //-- is First Time
throws Exception
{
 String rn  = "getLIN";
 String outp = "";
 try
 {
  outp += "<TR>";
  outp += "<TD><INPUT name='MNU' type=RADIO value='"+PRC_IN+"'";
  if(FTM_IN) outp += " CHECKED";
  outp += ">";
  outp += "<TD style='cursor:pointer;' onClick=z(this);>"+NAM_IN;
  outp += "</TR>";
 }
 catch(Exception e) { throw new DBException(rn,e.toString()); }
 return(outp);
}

private void getout( HttpServletResponse res
                , int LNG
                , int M1
                , int M2)
throws Exception
{
 String tmp = Gen.doProblem("M1",LNG,M1,M2);
 Gen.Transmit(res,tmp,"getout");
}

/*-- getUID ----------------------------*/
/*-- Returns the UID for a given LOGIN --*/
/*-- May be Empty                      --*/
/*----------------------------------------*/
private static String[] getUID( Connection CONN
                      , String    LOG_IN)
throws Exception
{
 String  rn  = "M1Servlet.getUID["+LOG_IN+"]";
 String[] outp = null;
 try
 {
  if(Gen.isEmpty(LOG_IN)) throw new Exception("empty_UID");

  String    qry = "select USR_ID"
                +     ",USR_DESCR"
```

```
                +      ",IS_BLOCKED"
                +      ",U_LOGIN"
                +      ",U_PASWD"
                + " from CFG_USR"
                + " where U_LOGIN is not null"
                +   " and U_LOGIN='"+LOG_IN+"'"
                ;
    String[][] rr  = Gen.Qry2Array(CONN,qry);
    int        len = Gen.getLEN(rr);
    if(len==0)
      outp = null;
    else if(len==1)
    {
      outp = new String[4];
      outp[0] = rr[0][0];  //-- uid
      outp[1] = rr[0][4];  //-- pwd
      outp[2] = rr[0][2];  //-- Blk
      outp[3] = rr[0][1];  //-- Name
    }
    else
      throw new Exception("Wrong_len ["+qry+"]");
  }
  catch(Exception e) { throw new DBException(rn,e.toString()); }
  return(outp);
 }

 private static String fixJS()
 {
  String s = "";
  s += "<SCRIPT LANGUAGE=JavaScript>";
  s += "function z(bb)";
  s += "{";
  s += "var tr=bb.parentNode;";
  s += "var r0=tr.children;";
  s += "var t0=r0.item(0);";
  s += "var r1=t0.children;";
  s += "var b1=r1.item(0);";
  s += "b1.checked=true;";
  s += "document.getElementsByName('B')[0].click();";
  s += "}";
  s += "</SCRIPT>";
  return(s);
 }
}
```

```
/*-- Menu step 2. Receives MNU,UID and LNG --*/
/*---------------------------------------*/
import libs.*;
import java.util.*;
import java.sql.Connection;
import javax.servlet.*;
import javax.servlet.http.*;
import org.w3c.dom.*;

public class M2Servlet extends HttpServlet
{
 public void service(HttpServletRequest req,HttpServletResponse res)
 {
  String     rn   = "M2Servlet";
  Connection conn = null;
  try
  {
   int    mnu = Gen.atoi((String)req.getParameter("MNU"));
   int    uid = Gen.atoi((String)req.getParameter("UID"));
   int    lng = Gen.atoi((String)req.getParameter("LNG"));
   String eip = (String)req.getParameter("EIP");
   String cip = (String)req.getParameter("CIP");

   String qry = "select PRC_1.PRC_ID,PRC_1.LABEL_ID"
          + " from PRC_1"
          +     ",PRC_USERS"
          + " where PRC_1.PRC_ID=PRC_USERS.PRC_ID"
          +   " and PRC_1.IS_BLOCKED=0"
          +   " and PRC_1.MNU_ID="+mnu
          +   " and PRC_USERS.USR_ID="+uid
          + " order by PRC_1.ORD_ID"
          ;
   conn = Gen.doConnect();

   String[][] rr  = Gen.Qry2Array(conn,qry);
   int        len = Gen.getLEN(rr);
   if(len==1)
   {
    int    prc = Gen.atoi(rr[0][0]);
    String tmp =
Gen.fixServlet("X0")+"?UID="+uid+"&LNG="+lng+"&PRC="+prc+"&EIP="+eip+"&
CIP="+cip;

    ServletContext ctx = getServletContext();
    ctx.getRequestDispatcher(tmp).forward(req,res);
   }
```

```
    else
    {
     String   des = Gen.getMenuDesc(conn,mnu,lng);
     HtmlForm frm = new HtmlForm(1);
     frm.LNG = lng;
     frm.EIP = eip;

     frm.FRV[0]   = "";
     frm.FRV[1]   = "M3";
     frm.FRV[4]   = Gen.get_UsrName(conn,uid)+" ["+des+"]";
     frm.FRV[15]  = APL.get_LOGO();
     frm.PRM[0][0] = "PRC";
     frm.PRM[0][1] = getPRS(conn,uid,lng,rr);
     frm.PRM[0][4] = Gen.fix_HIDDEN("UID",uid)
                   + Gen.fix_HIDDEN("LNG",lng)
                   + Gen.fix_HIDDEN("EIP",eip)
                   + Gen.fix_HIDDEN("CIP",cip)
                   ;
     frm.PRM[0][8] = fixJS();
     frm.PRM[0][11]= des;
     String prs = new Presenter(frm).getPRS();
     Gen.Transmit(res,prs,rn);
    }
   }
   catch(Exception e) { Gen.err(res,null,rn,e.toString()); }
   finally { Gen.doDisconnect(conn); }
  }

  /*-- getPRS ---------------------*/
  /*-- Returns Presentable part --*/
  /*-----------------------------*/
  private static String getPRS( Connection CONN
                        , int      UID_IN
                        , int      LNG_IN
                        , String[][] ARR_IN)
  throws Exception
  {
   String rn  = "M2.getPRS";
   String outp = "";
   try
   {
    String prs = "";

    int h_len = Gen.atoi(Gen.getOne(CONN,"select count(*) from OBJ_HUM"));
    int r_len = Gen.atoi(Gen.getOne(CONN,"select count(*) from OBJ_RDZ where
RDZ_DAT>trunc(sysdate)"));
```

```
    int t_len = Gen.atoi(Gen.getOne(CONN,"select count(*) from OBJ_RDZ"));
    int c_len = Gen.atoi(Gen.getOne(CONN,"select count(*) from HUM_CMT"));
    int a_len = Gen.atoi(Gen.getOne(CONN,"select count(*) from OBJ_RDZ where
PAY_DAT is null"));
    int b_len = Gen.atoi(Gen.getOne(CONN,"select count(*) from OBJ_RDZ where
PAY_DAT is not null"));

    int len = Gen.getLEN(ARR_IN);
    for(int j=0;j<len;j++)
    {
     int    prc = Gen.atoi(ARR_IN[j][0]);
     String nam = Gen.get_LABEL(CONN,Gen.atoi(ARR_IN[j][1]),LNG_IN);
     int    rws = -1;

         if(prc==205) rws = h_len;
      else if(prc==209) rws = h_len;
      else if(prc==215) rws = h_len;
      else if(prc==219) rws = c_len;
      else if(prc==221) rws = h_len;
      else if(prc==309) rws = a_len;
      else if(prc==310) rws = t_len;
      else if(prc==402) rws = r_len;
      else if(prc==406) rws = r_len;
      else if(prc==505) rws = a_len;
      else if(prc==507) rws = b_len;

     prs += getLIN(prc,rws,LNG_IN,nam,(j==0));
    }
    outp += "<TABLE>";
    outp += prs;
    outp += "</TABLE>";
   }
   catch(Exception e) { throw new DBException(rn,e.toString()); }
   return(outp);
 }

 /*-- get_LIN ---------*/
 /*-- Returns a Line --*/
 /*--------------------*/
 private static String getLIN( int    PRC_IN
                     , int    RWS_IN  //-- Number of Rows
                     , int    LNG_IN
                     , String  NAM_IN
                     , boolean FTM_IN) //-- is First Time
 throws Exception
 {
```

```
  String rn   = "getLIN";
  String outp = "";
  try
  {
   outp += "<TR title='"+PRC_IN+"'>";

   if(RWS_IN==0)
    outp += "<TD> ";
   else
   {
    outp += "<TD><INPUT name='PRC' type=RADIO value='"+PRC_IN+"'";
    if(FTM_IN) outp += " CHECKED";
    outp += ">";
   }

       if(RWS_IN<0)  outp += "<TD style='cursor:pointer;'
onClick=z(this);>"+NAM_IN;
    else if(RWS_IN==0) outp += "<TD style='cursor:not-allowed;'>"+NAM_IN;
    else            outp += "<TD style='cursor:pointer;' onClick=z(this);>"+NAM_IN+"
("+RWS_IN+")";

    outp += "</TR>";
  }
  catch(Exception e) { throw new DBException(rn,e.toString()); }
  return(outp);
 }

 private static String fixJS()
 {
  String s = "";
  s += "<SCRIPT LANGUAGE=JavaScript>";
  s += "function z(bb)";
  s += "{";
  s += "var tr=bb.parentNode;";
  s += "var r0=tr.children;";
  s += "var t0=r0.item(0);";
  s += "var r1=t0.children;";
  s += "var b1=r1.item(0);";
  s += "b1.checked=true;";
  s += "document.getElementsByName('B')[0].click();";
  s += "}";
  s += "</SCRIPT>";
  return(s);
 }
}
```

```
/*-- Menu step 3. Receives UID,PRC to follow --*/

import libs.*;
import java.sql.Connection;
import javax.servlet.*;
import javax.servlet.http.*;
import org.w3c.dom.*;

public class M3Servlet extends HttpServlet
{
 public void service( HttpServletRequest  req
                , HttpServletResponse res)
 {
  String rn = "M3Servlet";
  try
  {
   int    prc = Gen.atoi((String)req.getParameter("PRC"));
   int    uid = Gen.atoi((String)req.getParameter("UID"));
   int    lng = Gen.atoi((String)req.getParameter("LNG"));
   String eip = (String)req.getParameter("EIP");
   String cip = (String)req.getParameter("CIP");

   String tmp =
Gen.fixServlet("X0")+"?UID="+uid+"&LNG="+lng+"&PRC="+prc+"&EIP="+eip+"&
CIP="+cip;

   ServletContext ctx = getServletContext();

   ctx.getRequestDispatcher(tmp).forward(req,res);
  }
  catch(Exception e) { Gen.err(res,null,rn,e.toString()); }
 }
}
```

```
/*-- StartUpServlet ----------------------*/
/*-- Loads application System properties --*/
/*----------------------------------------*/
import libs.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class StartUpServlet extends HttpServlet
{
 public void init(ServletConfig config)
 throws ServletException
 {
  super.init(config);

  APL    pl  = new APL("StartUpServlet");
  String  msg = pl.getMSG();
  boolean ok  = pl.getOK();
  if(ok)
    Gen.Report(msg);
  else
  { try{ Gen.writeFILE("C:/StartUpServlet.err",msg); } catch(Exception e) {} }
 }
}
```

```
/*- X0Servlet ----------*/
/*- Procedure Starter for Users  -*/
/*- 1. accepts new Tree         -*/
/*- 2. discovers bean to call   -*/
/*- 3. Calls the bean           -*/
/*-----------------------------*/
import libs.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import org.w3c.dom.*;
import java.sql.Connection;
import FrmBeans.*;

public class X0Servlet extends HttpServlet
{
 public void service( HttpServletRequest  REQ_IN
                , HttpServletResponse RES_IN)
 {
  String    rn  = "X0Servlet";
  Connection conn = null;
  try
  {
   int   prc = Gen.atoi((String)REQ_IN.getParameter("PRC"));
   int   uid = Gen.atoi((String)REQ_IN.getParameter("UID"));
   int   lng = Gen.atoi((String)REQ_IN.getParameter("LNG")); if(Gen.isEmpty(lng))
lng = 1;
   String eip = (String)REQ_IN.getParameter("EIP");
   String cip = (String)REQ_IN.getParameter("CIP");

   String qry = "select BEAN_ID,STP_NO from PRC_2 where PRC_ID="+prc+" order
by STP_NO";

   conn = Gen.doConnect();                          //-* Connect to DB
   String[][] rr  = Gen.Qry2Array(conn,qry);
   int       len = Gen.getLEN(rr);
   if(len>0)
   {
    String bn  = rr[0][0];  bn ="FrmBeans."+bn+"bean";
    int   stp = Gen.atoi(rr[0][1]);

    FormBean frm = (FormBean) Class.forName(bn).newInstance();  //- Instantiate the
FormBean
    if(frm!=null)
    {
     iSession ses  = new iSession(uid,lng,prc,eip,cip);
```

```
        Document tree = ses.getTREE();

        ses.getTAG();

        Presenter pre = frm.work(conn,tree);              //- execute the FormBean
        String    prs = pre.getPRS();                   //- we get the output...
        Gen.doDisconnect(conn);                          //-* disConnect from DB

        if(!Gen.isEmpty(prs))
          Gen.Transmit(RES_IN,prs,rn);                   //- ...and send it over the Net
        else
          throw new Exception("Nothing_to_present ["+bn+"]");
       }
       else
        throw new Exception("Bean_not_found ["+bn+"]");
     }
     else
      throw new Exception("Wrong_qry ["+qry+"] ["+len+"]");
    }
    catch(Exception e)
    {
     Gen.doDisconnect(conn);
     Gen.err(RES_IN,null,rn,e.toString());
    }
   }
}
```

```
/*- X1Servlet --------------------*/
/*- Procedure stepper          -*/
/*- 1. accepts input data + tree -*/
/*- 2. steps by 1              -*/
/*- 3. discovers bean to call   -*/
/*- 4. Calls the bean          -*/
/*-------------------------------*/
import libs.*;
import FrmBeans.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import org.w3c.dom.*;
import java.sql.Connection;

public class X1Servlet extends HttpServlet
{
 public void service( HttpServletRequest  REQ_IN
                    , HttpServletResponse RES_IN)
 {
  String     rn   = "X1Servlet";
  Document   tree = null;
  Connection conn = null;
  try
  {
   String[][] vars = getWebVars(REQ_IN);
   String     fil  = Gen.getParamValue(vars,"FIL");

   if(Gen.isEmpty(fil))
    throw new DBException(rn,":No FIL");
   else
   {
    String prs = "";
    conn = Gen.doConnect();                  //-* Connect to DB
    iSession ses = new iSession(conn,fil);       //-- We have a correct session
    tree = ses.getTREE();

    if(Gen.isEmpty(tree))
     throw new Exception(rn+":No tree ["+fil+"]");
    else
     prs = getPRS(vars,conn,tree);

    Gen.doDisconnect(conn);                  //-* Disconnect from DB
    Gen.Transmit(RES_IN,prs,rn);               //-- send it over the Net
   }
  }
```

```
  catch(Exception e)
  {
   Gen.doDisconnect(conn);
   Gen.err(RES_IN,tree,rn,e.toString());
  }
 }

 private static String[][] getWebVars(HttpServletRequest REQ)
 throws Exception
 {
  String rn    = "getWebVars";
  String name  = "";
  String value = "";
  String[][] outp = null;
  try
  {
   REQ.setCharacterEncoding("UTF-8");
   String contentType = REQ.getContentType();
   if(Gen.isEmpty(contentType)) contentType="text/html";

   Enumeration e = REQ.getParameterNames();
   while(e.hasMoreElements())
   {
    name  = (String) e.nextElement();
    value = (String) REQ.getParameter(name);
    outp = Gen.addEntry(outp,name,value);
   }

   if(Gen.isEmpty(outp)) throw new Exception("No_webVars_found");
  }
  catch(Exception e) { throw new DBException(rn,e.toString()+" name="+name+"
value="+value); }
  return(outp);
 }

 private static String getPRS( String[][] WebVars
                     , Connection CONN
                     , Document   TREE)
 throws Exception
 {
  boolean TRACE = false;

  if(TRACE) Gen.Report("$$$$$$$$$$$$$$$$$$$$$$$$$$$>Starting Navigator...");

  String rn = "Navigator";
  String outp = "";
```

```
try
{
 if(Gen.isEmpty(TREE)) throw new Exception("No_TREE");

 int npr = Gen.atoi(Gen.getParamValue(WebVars,"NO_PRM"));
 if(TRACE) Gen.Report("$$$$$$$$$$$$$$$$$$$$$$$$$$>npr="+npr);

 if(npr!=1)
 {
  String prevBean = getBean(CONN,TREE);
  if(TRACE) Gen.Report("$$$$$$$$$$$$$$$$$$$$$$$$$$>PrevBean="+prevBean);

  FormBean frm1 = runBean(prevBean,TRACE);
  frm1.getVars(WebVars,CONN,TREE);
  if(TRACE) Gen.Report("$$$$$$$$$$$$$$$$$$$$$$$$$$>After PrevBean");
  Gen.Dom2File(TREE,APL.getTMPPATH()+prevBean+"_get.xml");
 }

 nextSTP(TREE,1);       //- One step forward

 String currBean = getBean(CONN,TREE);
 if(TRACE) Gen.Report("$$$$$$$$$$$$$$$$$$$$$$$$$$>:CurrBean="+currBean);

 FormBean frm2 = runBean(currBean,TRACE);
 if(TRACE) Gen.Report("$$$$$$$$$$$$$$$$$$$$$$$$$$>:After CurrBean");
 Presenter prs = frm2.work(CONN,TREE);

 if(prs==null)
 {
  if(TRACE) Gen.Report("??????????????????????????>:No Result");
 }
 else
 {
  outp = prs.getPRS();
  Gen.Dom2File(TREE,APL.getTMPPATH()+currBean+"_set.xml");
 }
}
catch(Exception e) { throw new DBException(CONN,rn,e.toString()); }
return(outp);
}

private static String getBean(Connection CONN,Document TREE)
throws Exception
{
 String rn = "getBean";
 if(CONN==null) throw new Exception(rn+" No_CONN");
```

```
  String qry  = "";
  String outp = "";
  try
  {
   int prc = Gen.getPRC(TREE);
   int stp = Gen.getSTEP(TREE);

   qry = "select BEAN_ID from PRC_2 where PRC_ID="+prc+" and STP_NO="+stp;

   String[][] rr  = Gen.Qry2Array(CONN,qry);
   int        len = Gen.getLEN(rr);

       if(len==0) throw new Exception("Empty qry=["+qry+"]");
   else if(len==1) outp = rr[0][0];
   else
    throw new Exception("More than one! qry=["+qry+"]");
  }
  catch(Exception e) { throw new DBException(CONN,rn,e.toString()); }
  return(outp);
 }

 private static FormBean runBean( String  BEAN
                        , boolean TRC_IN)
 throws Exception
 {
  String  rn  = "runBean";
  FormBean outp = null;
  try
  {
   if(Gen.isEmpty(BEAN)) throw new Exception(rn+" Bean not exists");
   String bn = "FrmBeans."+BEAN+"bean";                    //- compliment with
"bean"
   outp = (FormBean) Class.forName(bn).newInstance();          //- Instantiate the
FormBean
   if(outp==null) throw new Exception("Bean not found ["+BEAN+"]");
  }
  catch(Exception e) { throw new DBException(rn,e.toString()); }

  if(TRC_IN) Gen.Report(rn+" :Running_Bean ["+BEAN+"]...");

  return(outp);
 }

 /*-- nextSTP ----------*/
 /*-- Steps tree by 1 --*/
 /*--------------------*/
```

```
 private static void nextSTP(Document TREE,int NUM)
 throws Exception
 {
  String rn = "nextSTP";
  try
  {
   int curr_step = Gen.getSTEP(TREE);
   if(Gen.isEmpty(curr_step)) throw new Exception(rn+" EmptyStep" );
   int next_step = curr_step+NUM;
   Gen.upd1(TREE,"STP",next_step);
  }
  catch(Exception e) { throw new Exception(rn+e.toString()); }
 }
}
```

package libs;

```
/*-- APLoader ----------------------------------*/
/*-- Application Properties Loader          --*/
/*-- Loads system properties                --*/
/*-- Returns status (OK) and Message (MSG)  --*/
/*-------------------------------------------*/
import java.io.*;
import java.util.*;
import java.sql.*;

public class APL
{
 private final String  VER = "1.00";

 private final static String MY_HOME  = "MY_HOME";   //- Applic Home Path
 private final static String WEB_ABS  = "WEB_ABS";   //- Web server path
 private final static String WEB_REL  = "WEB_REL";   //- Web Relative Path
 private final static String SRV_REL  = "SRV_REL";   //- Servlet Relative Path
 private final static String DBHOST   = "DBHOST";    //- * Commit Host IP
 private final static String DBNAME   = "DBNAME";    //- * DB SID
 private final static String DBUSR0   = "DBUSR0";    //- * DB user0 name
 private final static String DBPAS0   = "DBPAS0";    //- * DB user0 pwd
 private final static String FOP_PATH = "FOP_PATH";  //- FOP path
 private final static String TMP_PATH = "TMP_PATH";  //- TMP path
 private final static String DOC_PATH = "DOC_PATH";  //- FOP Docs path
 private final static String MAILHOST = "MAILHOST";
 private final static String LOGO     = "LOGO";      //- Default LOGO

 private boolean OK  = false; public boolean getOK()  { return(this.OK); }
```

```
private String  MSG = "";   public String  getMSG() { return(this.MSG); }

/*-- APL Constructor --*/
/*---------------------*/
public APL(String RN_IN)
{
 try
 {
  String ps = System.getProperty("file.separator");   //-- Path separator
  String dt = Gen.getCurDATE()+' '+Gen.getCurTIME();  //-- Start time
  this.MSG = "Starting at "+dt+" from "+RN_IN;
  this.OK  = false;

  String home = "C:"+ps+"fotis"+ps;
  String abs  = "C:/TomCat7/webapps/ROOT/"; //-- Web CONTEXT ROOT
  String rel  = "/";                  //-- Web relative Path
  String srv  = "/";                  //-- Servlet relative Path
  String dbh  = "127.0.0.1";          //-- DBHOST
  String dbn  = "TAKOSDB";            //-- DBNAME
  String dbu0 = "FOTIS";              //-- DB user name
  String dbp0 = "VUNAPARTIS";         //-- DB user Password
  String fop  = home+"FOP"+ps;        //-- FOP folder
  String tmp  = home+"TEMP"+ps;       //-- Temp folder
  String doc  = home+"DOC"+ps;        //-- Doc folder
  String mlh  = "160.2.0.210";        //-- MAIL host
  String logo = "Figura";             //-- default Logo

  System.setProperty(MY_HOME,home);
  System.setProperty(WEB_ABS,abs);
  System.setProperty(WEB_REL,rel);
  System.setProperty(SRV_REL,srv);
  System.setProperty(DBHOST,dbh);
  System.setProperty(DBNAME,dbn);
  System.setProperty(DBUSR0,dbu0);
  System.setProperty(DBPAS0,dbp0);
  System.setProperty(FOP_PATH,fop);
  System.setProperty(TMP_PATH,tmp);
  System.setProperty(DOC_PATH,doc);
  System.setProperty(MAILHOST,mlh);
  System.setProperty(LOGO,logo);

  try
  {
   Connection conn = Gen.doConnect();
   this.OK = true;
   Gen.doDisconnect(conn);
```

```
    if(this.OK)
    {
     this.MSG += ", MY_HOME="+getHOME();
     this.MSG += ", WEB_ABS="+abs;
     this.MSG += ", WEB_REL="+rel;
     this.MSG += ", DBHOST="+System.getProperty(DBHOST);
     this.MSG += ", DBNAME="+System.getProperty(DBNAME);
     this.MSG += ", DBUSR0="+System.getProperty(DBUSR0);
     this.MSG += ", TMP_PATH="+getTMPPATH();
     this.MSG += ", FOP_PATH="+getFOPPATH();
     this.MSG += ", IP="+Gen.get_IP();
    }
   }
   catch(Exception e)
   {
    this.MSG += ", Can NOT find ORACLE";
    this.MSG += ", DBHOST="+System.getProperty(DBHOST);
    this.MSG += ", DBNAME="+System.getProperty(DBNAME);
    this.MSG += ", DBUSR0="+System.getProperty(DBUSR0);
    this.MSG += ", Error=["+e.toString()+"]";
   }
  }
  catch(Exception e)
  {
   this.OK  = false;
   this.MSG += "!!!Problem in APL from "+RN_IN+"["+e.toString()+"]";
  }
}

/*-- getHOME ---------------------------------*/
/*-- Returns the Application home Directory --*/
/*-----------------------------------------*/
public static String getHOME()
{
 String outp = System.getProperty(MY_HOME);
 return(outp);
}

/*-- getDBHOST -----------------------*/
/*-- Returns the IP of the DB Server --*/
/*-------------------------------------*/
public static String getDBHOST()
throws Exception
{
 String outp = System.getProperty(DBHOST);
```

```
 if(Gen.isEmpty(outp)) throw new Exception(" DBHOST is empty");
 return(outp);
}

public static String getDBNAME()
throws Exception
{
 String outp = System.getProperty(DBNAME);
 if(Gen.isEmpty(outp)) throw new Exception(" DBNAME is empty");
 return(outp);
}

public static String getDBPAS0()
throws Exception
{
 String outp = System.getProperty(DBPAS0);
 if(Gen.isEmpty(outp)) throw new Exception(" DBPASS is empty");
 return(outp);
}

public static String getDBUSR0()
throws Exception
{
 String outp = System.getProperty(DBUSR0);
 if(Gen.isEmpty(outp)) throw new Exception(" DBUSR1 is empty");
 return(outp);
}

public static String getTMPPATH()
throws Exception
{
 String outp = System.getProperty(TMP_PATH);
 if(Gen.isEmpty(outp)) throw new Exception(" TMP_PATH is empty");
 if(!Gen.folderExists(outp)) Gen.createFolder(outp);
 return(outp);
}

public static String getSRVPATH()
throws Exception
{
 String outp = System.getProperty(SRV_REL);
 if(Gen.isEmpty(outp)) throw new Exception(" SRV_PATH is empty");
 return(outp);
}

public static String getWEBPATH()
```

```
throws Exception
{
 String outp = System.getProperty(WEB_REL);
 if(Gen.isEmpty(outp)) throw new Exception(" WEB_PATH is empty");
 return(outp);
}

public static String getWEBABS()
throws Exception
{
 String outp = System.getProperty(WEB_ABS);
 if(!Gen.folderExists(outp)) outp = "";
 if(Gen.isEmpty(outp)) throw new Exception(" WEB_ABS is empty");
 return(outp);
}

public static String getDOCPATH()
throws Exception
{
 String outp = System.getProperty(DOC_PATH);
 if(Gen.isEmpty(outp)) throw new Exception(" DOC_PATH is empty");
 if(!Gen.folderExists(outp)) Gen.createFolder(outp);
 return(outp);
}

/*-- getFOPPATH ------------------------*/
/*-- Returns the FOP libraries folder --*/
/*-- If not found raise exception     --*/
/*-------------------------------------*/
public static String getFOPPATH()
throws Exception
{
 String outp = System.getProperty(FOP_PATH);
 if(Gen.isEmpty(outp)) throw new Exception(" FOP_PATH is empty");
 if(!Gen.folderExists(outp)) throw new Exception("No FOP path found");
 return(outp);
}

public static String getMAILHOST()
throws Exception
{
 String outp = System.getProperty(MAILHOST);
 if(Gen.isEmpty(outp)) throw new Exception(" MAILHOST is empty");
 return(outp);
}
```

```
 public static String get_LOGO()
 { return(System.getProperty(LOGO)); }
}
```

package libs;

```
/*-- DBException ---------------------------------*/
/*-- I do not trust the Autocommit so          --*/
/*-- We must rollback before closing Connection --*/
/*-----------------------------------------------*/
import java.sql.*;

public class DBException extends Exception
{
 public DBException(Connection CONN,String RN,String MSG)
 {
  super("["+RN+"<FOTIS>"+Gen.String2ANSI(MSG)+"]");
  if(CONN!=null)
  {
   Gen.debug(" conn closed from "+RN);
   try { CONN.close(); } catch(Exception e) {}
  }
 }

 public DBException(String RN,String MSG)
 { super(RN+":"+MSG); }
}
```

```
package libs;

/*-- Fotis --*/
/*-----------*/
import org.w3c.dom.*;
import org.xml.sax.InputSource;
import org.apache.xerces.parsers.*;
import java.sql.*;
import java.io.*;
import java.net.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.xml.transform.*;
import javax.xml.transform.stream.*;
import javax.xml.transform.dom.*;
import oracle.sql.OPAQUE;
import oracle.xdb.XMLType;

public class Gen
{
 final static private  String CODE_PAGE = "UTF-8";
 final static public   String BColor = "#E0FFFF";
 final static public   String SColor = "#6495ED"; //-- Submited Form BGround color

 final static private  int   ND[] = new int[] {31,28,31,30,31,30,31,31,30,31,30,31};
 final static private  int   BASE = 1900;

 /*-- doConnect -------*/
 /*-- Connects to DB --*/
 /*--------------------*/
 public static Connection doConnect()
 throws Exception
 {
  Connection outp = null;

  String ORA_IP  = APL.getDBHOST(); //-- "127.0.0.1"
  String ORA_SID = APL.getDBNAME(); //-- "TAKOSDB"
  String ORA_USR = APL.getDBUSR0(); //-- "FOTIS"
  String ORA_PWD = APL.getDBPAS0(); //-- "VUNAPARTIS"
  try
  {
   getJDBC();
   String    cstr  = "jdbc:oracle:thin:@"+ORA_IP+":1521:"+ORA_SID;
   Properties props = new Properties();
```

```
    props.put("user",ORA_USR);
    props.put("password",ORA_PWD);
    outp = DriverManager.getConnection(cstr,props);
    outp.setAutoCommit(false);
   }
   catch (Exception e)
   { throw new Exception("Can_not_CONNECT ip=["+ORA_IP+"] db=["+ORA_SID+"]
usr=["+ORA_USR+"] "+e.toString()); }
   return(outp);
  }

  /*-- dbSQL --------------------*/
  /*-- Executes an SQL command --*/
  /*----------------------------*/
  public static void dbSQL( Connection CONN
                    , String    SQL_IN
                    , String    RN_IN)
  throws Exception
  {
   String rn="Gen.dbSQL";
   try
   {
    String sql = String2ANSI(SQL_IN);
    execSQL(CONN,sql,RN_IN);
   }
   catch(Exception e)
   { throw new DBException(CONN,rn,e.toString()); }
  }

  /*-- fixTel -------------------*/
  /*-- Executes an SQL command --*/
  /*----------------------------*/
  public static String fixTel( String TEL_IN
                     , String CIP_IN)
  throws Exception
  {
   String rn   = "Gen.fixTel";
   String outp = "";

   if(Gen.isEmpty(TEL_IN))
    outp = "";
   else
   {
    if(isTEL(TEL_IN))
    {
     String nm = TEL_IN;
```

```
      String t1 = get3CX(TEL_IN,CIP_IN);

      if(Gen.isEmpty(t1))
        outp = nm;
      else
       outp= "<INPUT TYPE='button' VALUE='"+nm+"'
onClick={f72('"+t1+"',10,10);}>";
     }
     else
      outp = "{"+TEL_IN+"}";
   }
   return(outp);
  }

  /*-- get3CX ------------------------------------------------------------------------------*/
  /*-- Returns call to 3CX makeCall system                                          --*/
  /*--
http://10.0.0.98:5481/ivr/PbxAPI.aspx?func=make_call&from=113&to=2109914115&pi
n=58903765 --*/
  /*-------------------------------------------------------------------------------------*/
  private static String get3CX( String TEL_IN
                       , String CIP_IN)
  throws Exception
  {
   String rn   = "Gen.get3CX";
   String outp = "";
   try
   {
    String t1 = "";
    String t2 = "";

    if(CIP_IN.equals("10.0.0.101")) { t1="111"; t2="82097841"; }

    if(isEmpty(t1))
      outp = "";
    else
      outp =
"http://10.0.0.98:5481/ivr/PbxAPI.aspx?func=make_call&from="+t1+"&to="+TEL_IN+
"&pin="+t2;
   }
   catch(Exception e) { throw new DBException(rn,e.toString()); }
   return(outp);
  }

  public static String get_IP()
  throws Exception
```

```
{
 String hn = InetAddress.getLocalHost().getHostName();
 InetAddress[] ad = InetAddress.getAllByName(hn);
 if(ad==null) throw new Exception("Gen.getIPs:No IP ports");
 return(ad[0].getHostAddress());
}

public static void doDisconnect(Connection conn)
{ if(conn!=null) { try{conn.close();} catch(Exception e){} } }

/*-- getLABEL ----------------------------------------*/
/*-- Returns descr from CFG_LABELS identified by ID --*/
/*----------------------------------------------------*/
public static String get_LABEL( Connection CONN
                              , int      LBL_IN
                              , int      LNG_IN)
throws Exception
{
 String   rn  = "Gen.get_LABEL";
 String   outp = "";
 Statement stm  = null;
 ResultSet rsl  = null;
 String   qry  = "";
 try
 {
  if(LBL_IN<0)
   outp = "";
  else
  {
   qry = "select getl(LABEL_ID,"+LNG_IN+") from CFG_LBL where
LABEL_ID="+LBL_IN;
   stm = CONN.createStatement();
   stm.execute(qry);
   rsl = stm.getResultSet();

   if(rsl.next())
    outp = ANSI2String(rsl.getString(1));
   else
    outp = "{"+LBL_IN+"}";
  }
 }
 catch(Exception e)
 { throw new DBException(rn,e.toString()+" ["+qry+"]"); }
 finally
 {
  try { rsl.close(); } catch(Exception x) {}
```

```
  try { stm.close(); } catch(Exception x) {}
 }
 return(outp);
}

/*-- getLABELS ------------------------------------------------------------*/
/*-- Gets the labels (array B) from table CFG_LABELS identified by STR --*/
/*-------------------------------------------------------------------------*/
public static String[] getLABELS( Connection CONN
                        , int       LNG_IN
                        , String    STR)
throws Exception
{
 String rn="Gen.getLABELS";
 if(isEmpty(STR)) throw new DBException(rn,"NoString");
 String[] outp = {};
 String[] ff0  = String2Array(STR,',');
 int      len0 = getLEN(ff0);
 if(len0>0)
 {
  outp = new String[len0]; MakeEmpty(outp);
  int    lng = (isEmpty(LNG_IN))?1:LNG_IN;
  String tmp = "";
  for(int j0=0;j0<len0;j0++)
  {
   String tt = ff0[j0];
   if(isEmpty(tt)) tt="null";
   tmp += tt+',';
  }
  tmp = removeLAST(tmp,1);

  String    qry  = "select LABEL_ID,getl(LABEL_ID,"+lng+")"
               + " from CFG_LBL"
               + " where LABEL_ID in("+tmp+")"
               ;
  String[][] ff1  = Qry2Array(CONN,qry);
  int        len1 = getLEN(ff1);
  for(int j1=0;j1<len1;j1++)
  {
   String id = ff1[j1][0];
   String lb = ff1[j1][1];
   for(int j0=0;j0<len0;j0++) if(ff0[j0].equals(id)) outp[len0-j0-1]=lb;
  }
  for(int j0=0;j0<len0;j0++)
  {
   String a = ff0[len0-j0-1];
```

```
      String b = outp[j0];
      if(!isEmpty(a) && isEmpty(b)) outp[j0]="{"+a+"}";
     }
    }
    return(outp);
  }

  public static String[] getLABELS(int LNG,String STR)
  throws Exception
  {
    Connection conn = Gen.doConnect();
    String[] outp = getLABELS(conn,LNG,STR);
    Gen.doDisconnect(conn);
    return(outp);
  }

  /*-- Transmit ----------------------*/
  /*-- Transmits a page over Servlet --*/
  /*----------------------------------*/
  public static void Transmit( HttpServletResponse RES
                      , String          FRM_IN
                      , String          RN)
  {
    try
    {
      RES.setDateHeader("Expires",System.currentTimeMillis()+720000);   //- expire in 2
hours
      RES.setContentType("text/html;charset="+CODE_PAGE);
/*
      RES.setHeader("Cache-Control","no-store");
*/
      PrintWriter pw = RES.getWriter();
      pw.print(FRM_IN);
      pw.flush();
      pw.close();
    }
    catch(Exception e)
    {
      String tmp = "Gen.Transmit";
      if(!isEmpty(RN)) tmp += " [rn="+RN+"]";
      tmp += " ["+FRM_IN+"]";
      tmp += " ["+e.toString()+"]";
    }
  }

  /*-- atoi --------------------------------------*/
```

```
/*-- converts an ASCII string (S) into integer --*/
/*-----------------------------------------------*/
public static int atoi(String S)
{
 if(isEmpty(S)) return(-1);
 int    outp = -1;
 String tmp  = S.trim();
 tmp = removeDOTS(tmp);
 try { outp=Integer.parseInt(tmp); }
 catch(NumberFormatException e) { }
 return(outp);
}

/*-- atol ----------------------------------*/
/*-- converts an ASCII string (S) into long --*/
/*----------------------------------------------*/
public static long atol(String S)
{
 long outp = -1;
 if(!isEmpty(S))
 {
  String tmp = S.trim();
  if(!isEmpty(tmp))
  {
   tmp = removeDOTS(tmp);
   if(!isEmpty(tmp))
   {
    if(tmp.equals("0"))
      outp = 0;
    else if(tmp.equals("00"))
      outp = 0;
    else if(tmp.equals("000"))
      outp = 0;
    else if(tmp.equals("0000"))
      outp = 0;
    else
    {
      tmp = ltrim(tmp,'0');
      try { outp=Long.parseLong(tmp); } catch(NumberFormatException e) { }
    }
   }
  }
 }
 return(outp);
}
```

```
/*-- sleep ----------------*/
/*-- Sleeps SECS seconds --*/
/*------------------------*/
public static boolean sleep(int SECS)
{
 boolean outp = true;
 if(SECS>0)
 { try { Thread.sleep(SECS*1000); } catch(Exception e) { outp=false; } }
 return(outp);
}

/*-- ltrim ----------------------------------*/
/*-- Removes Char C from the begining of S --*/
/*-----------------------------------------*/
public static String ltrim(String S,char C)
{
 String outp = S;
 if(!isEmpty(S))
 {
  char c = C;
  int  p = 0;
  for(int j=0;(j<S.length() && c==C);j++)
  {
   c = S.charAt(j);
   if(c==C) p++;
  }
  if(p>0) outp = getLast(S,p);
 }
 return(outp);
}

/*-- lpad ---------------------------------*/
/*-- Returns <L> string left padded with --*/
/*-- C up to <L> chars length            --*/
/*----------------------------------------*/
public static String lpad( String S
                         , int    L
                         , String C)
{
 String outp = "";
 if(!isEmpty(S) && L>0)
 {
  int dif = L-S.length();
  for(int j=0;j<dif;j++) outp+=C;
 }
 return(outp+S);
```

```
}

public static String lpad( int    S
                         , int    L
                         , String C)
{ return(lpad(Integer.toString(S),L,C)); }

/*-- lpad ---------------------------------*/
/*-- Returns <L> string left padded with --*/
/*-- spaces up to <L> chars length       --*/
/*---------------------------------------*/
public static String lpad(String S,int L)
{ return(lpad(S,L," ")); }

/*-- rpad ---------------------------------*/
/*-- Returns <L> string right padded with --*/
/*-- spaces up to <L> chars length        --*/
/*---------------------------------------*/
public static String rpad(String S,int L,String C)
{
 String outp = "";
 if(!isEmpty(S) && L>0)
 {
  int dif = L-S.length();
  for(int j=0;j<dif;j++) outp+=C;
 }
 return(S+outp);
}

public static int getLEN(String STR)
{
 int outp = 0;
 try{ outp=STR.length(); } catch(Exception e) { }
 return(outp);
}

public static int getLEN(int[][] ARR)
{
 int outp = 0;
 try{ outp=ARR.length; } catch(Exception e) {}
 return(outp);
}

public static int getLEN(int[] ARR)
{
 int outp = 0;
```

```
  try{ outp=ARR.length; } catch(Exception e) {}
  return(outp);
 }

 public static int getLEN(long[] ARR)
 {
  int outp = 0;
  try{ outp=ARR.length; } catch(Exception e) {}
  return(outp);
 }

 public static int getLEN(String[] ARR)
 {
  int outp = 0;
  try{ outp=ARR.length; } catch(Exception e) {}
  return(outp);
 }

 public static int getLEN(String[][] ARR)
 {
  int outp = 0;
  try{ outp=ARR.length; } catch(Exception e) {}
  return(outp);
 }

 public static String rpad(String S,int L)
 { return(rpad(S,L," ")); }

 public static boolean isEmpty(String a)
 { return((a==null||a.equals("")||a.equals("null")||a.equals(" ")||a.equals("-
1")||a.equals("nfd"))); }

 public static boolean isEmpty(int a)
 { return((a==-1)?true:false); }

 public static boolean isEmpty(long a)
 { return((a==-1)?true:false); }

 public static boolean isEmpty(double a)
 { return((a==-1)?true:false); }

 public static boolean isEmpty(Node a)
 { return((a==null)?true:false); }

 public static boolean isEmpty(String[] a)
 { return(getLEN(a)==0); }
```

```
public static boolean isEmpty(String[][] a)
{ return(getLEN(a)==0); }

public static boolean isEmpty(Document a)
{ return(a==null); }

public static boolean isEmpty(Document[] a)
{
 boolean outp = true;
 int     len  = 0;
 try { len=a.length; } catch(Exception e) { }
 for(int j=0;j<len;j++) if(!isEmpty(a[j])) outp=false;
 return(outp);
}

/*-- getJDBC ---------------------*/
/*-- Registers the JDBC drivers --*/
/*--------------------------------*/
private static void getJDBC()
throws Exception
{
 try { DriverManager.registerDriver(new oracle.jdbc.OracleDriver()); }
 catch (Exception e) { throw new Exception("JDBC Problem:"+e.toString()); }
}

/*-- Debug ----------------------------*/
/*-- logs a debug message (TYPE=1) --*/
/*---------------------------------*/
public static void Debug(String MSG)
{ insLOG(1,"Debug:"+MSG); }

/*-- debug ----------------------------*/
/*-- logs a debug message (TYPE=1) --*/
/*---------------------------------*/
public static void debug(String MSG)
{ insLOG(1,"Debug:"+MSG); }

/*-- Report ---------------------------*/
/*-- logs a report message (TYPE=2) --*/
/*---------------------------------*/
public static void Report(String MSG)
{ insLOG(2,"Report:"+MSG); }

/*-- Warning --------------------------*/
/*-- logs a warning message (TYPE=3) --*/
```

```
/*-------------------------------------*/
public static void Warning(String MSG)
{ insLOG(3,"Warning:"+MSG); }

/*-- Error ------------------------------*/
/*-- Handles an Error message (TYPE=4) --*/
/*-----------------------------------------*/
public static void Error(String MSG)
{ insLOG(4,"Error:"+MSG); }

/*-- insLOG ------------------------------------------------*/
/*-- logs a message into Oracle log table                --*/
/*-- As an error function it must handle its own exception  --*/
/*-- Warning!!! The INS_LOG procedure must be present in DB --*/
/*-- The Commit is done within the INS_LOG               --*/
/*-----------------------------------------------------------*/
private static void insLOG(int TYP,String MSG)
{
 Connection conn = null;
 String     tmp  = "";
 try
 {
  if(isEmpty(MSG))
    tmp="insLOG:no MSG found";
  else
  {
    tmp = MSG.replace('\'',"");
    tmp = getFirst(tmp,3600);
  }
  conn=doConnect();
  execSQL(conn,"begin INS_LOG("+TYP+",'"+tmp+"');end;","insLOG");
  doDisconnect(conn);
 }
 catch(Exception e)
 { doDisconnect(conn); }
}

/*-- execSQL ------------------------------------*/
/*-- executes an SQL statement over a given CONN --*/
/*-- on exception rollbacks and disconnects      --*/
/*-------------------------------------------------*/
private static void execSQL( Connection CONN
                           , String     SQL_IN
                           , String     RN_IN)
throws Exception
{
```

```
  String rn="Gen.execSQL";
  if(CONN==null)       throw new Exception(rn+" No CONN");
  if(isEmpty(SQL_IN))  throw new Exception(rn+" [SQL is empty]");
  if(SQL_IN.length()>4000) throw new Exception(rn+" [SQL>4000 bytes]");

  Statement stm1 = null;
  try
  {
   stm1=CONN.createStatement();
   stm1.execute(SQL_IN);
   stm1.close();
  }
  catch(Exception e)
  { throw new Exception(rn+" ["+e.toString()+"][sql="+SQL_IN+"]"); }
 }

 /*-- getOne ----------------------------------------------*/
 /*-- Applied to Queries that respond with only one result --*/
 /*-- returns only one result from a query <QRY>          --*/
 /*--------------------------------------------------------*/
 public static String getOne( Connection CONN
                     , String    QRY)
 throws Exception
 {
  String    rn  = "Gen.getOne";
  String    outp = "";
  String[][] rr  = Qry2Array(CONN,QRY);
  int       len  = getLEN(rr);
  if(len!=1) throw new DBException(CONN,rn," wrong
result="+len+"[qry="+QRY+"]");
  outp = rr[0][0];
  return(outp);
 }

 /*-- Qry2Array -------------------------------------------*/
 /*-- on QRY problem issues a normal Exception (not DB) --*/
 /*-- !! Do not use debug inside this function         --*/
 /*--------------------------------------------------------*/
 public static String[][] Qry2Array( Connection CONN
                          , String    QRY_IN)
 throws Exception
 {
  String rn="Gen.Qry2Array";
  if(CONN==null)     throw new DBException(rn,":No CONN");
  if(isEmpty(QRY_IN)) throw new DBException(CONN,rn,":Empty QRY");
```

```
 Statement  stm  = null;
 ResultSet  rsl  = null;
 String[][] outp = {};                    //- qry holder
 try
 {
  stm = CONN.createStatement();
  rsl = stm.executeQuery(QRY_IN);
  outp = Rset2Array(rsl);
 }
 catch(Exception e)
 { throw new Exception(rn+"[Wrong qry]["+e.toString()+"][qry="+QRY_IN+"]"); }
 finally
 {
  try{ rsl.close(); } catch(Exception x) {}
  try{ stm.close(); } catch(Exception x) {}
 }

 return(outp);
}

/*-- Rset2Array ----------------------------------------*/
/*-- on QRY problem issues a normal Exception (not DB) --*/
/*-- !! Do not use debug inside this function         --*/
/*------------------------------------------------------*/
private static String[][] Rset2Array(ResultSet RSL_IN)
throws Exception
{
 String[][] outp = {};                              //- qry holder
 try
 {
  if(RSL_IN.next())                                 //- qry of at least one row
  {
   int            ROWS = 0;                         //- rows of qry
   Document       tree = String2Dom("<A></A>");
   Node           rt   = getROOT(tree);
   ResultSetMetaData rsmd = RSL_IN.getMetaData();
   int            COLS = rsmd.getColumnCount();     //- columns of qry
   Node           nd   = insREC(tree,rt,RSL_IN,COLS);
   for(ROWS=1;RSL_IN.next();ROWS++) nd = insREC(tree,nd,RSL_IN,COLS);

   outp = new String[ROWS][COLS];

   int  y = 0;
   Node n = rt;
   while(!isLEAF(n))
   {
```

```
        n = n.getChildNodes().item(0);
        for(int x=0;x<COLS;x++) outp[y][x] = getATTR(n,"X"+x);
        y++;
       }
       tree=null;
       if(y!=ROWS) throw new Exception("Gen.Rset2Array
y!=ROWS["+y+","+ROWS+"]");
     }
    }
    catch(Exception e)
    { throw new Exception("Gen.Rset2Array ["+e.toString()+"]"); }
    return(outp);
  }

  private static Node insREC( Document  TREE
                    , Node     ND_IN
                    , ResultSet RSL_IN
                    , int      COLS)
  throws Exception
  {
   Node outp = null;
   try
   {
    outp = insNODE(TREE,ND_IN,"Y",null);
    for(int j=0;j<COLS;j++)
    {
     String aa = RSL_IN.getString(j+1);
     aa = ANSI2String(aa);
     insATTR(TREE,outp,"X"+j,aa);
    }
   }
   catch(Exception e)
   { throw new Exception("Gen.insREC["+e.toString()+"]"); }
   return(outp);
  }

  /*-- String2Dom ----------------------*/
  /*-- Converts a XML string into DOM  --*/
  /*-------------------------------------*/
  public static Document String2Dom(String STR)
  throws Exception
  {
   if(isEmpty(STR)) throw new Exception("String2Dom:STR is empty");
   Document outp = Stream2DOM(new
ByteArrayInputStream(STR.getBytes("UTF8")));
   return(outp);
```

```
}

/*-- Stream2DOM ---------------------------*/
/*-- Returns the DOM coming from a stream --*/
/*-----------------------------------------*/
private static Document Stream2DOM(InputStream STREAM)
throws Exception
{
 String  rn  = "Stream2DOM";
 Document outp = null;
 try
 {
  DOMParser prs = new DOMParser();
  prs.parse(new InputSource(STREAM));
  outp = prs.getDocument();
 }
 catch(Exception e) { throw new Exception(rn+e.toString()); }
 return(outp);
}

/*-- Dom2String ---------------------------------------*/
/*-- Converts a DOM into String using UTF8 encoding  --*/
/*-- using the Xerces serializer                     --*/
/*-- Replaces Single quote to space (Oracle reasons) --*/
/*-----------------------------------------------------*/
public static String Dom2String(Document TREE)
throws Exception
{
 String rn  = "Dom2String";
 String outp = "";
 try
 {
  Node rt = getROOT(TREE);
  ByteArrayOutputStream os = new ByteArrayOutputStream();
  printNode(rt,os);
  outp = os.toString();
  outp = outp.replace('\'',' ');
 }
 catch(Exception e) { throw new Exception(rn+":"+e.toString()); }
 return(outp);
}

private static void printNode(Node ND,ByteArrayOutputStream OS)
{
 switch(ND.getNodeType())
 {
```

```
   case Node.DOCUMENT_NODE:
     NodeList nodes = ND.getChildNodes();
     if(nodes != null)
       for(int i=0;i<nodes.getLength();i++)
         printNode(nodes.item(i),OS);
     break;
   case Node.ELEMENT_NODE:
     String name = ND.getNodeName();
     wr("<"+name,OS);
     NamedNodeMap attributes = ND.getAttributes();
     for(int i=0;i<attributes.getLength();i++)
     {
      Node current = attributes.item(i);
      wr(" "+current.getNodeName()+"=\""+current.getNodeValue()+"\"",OS);
     }
     wr(">",OS);
     NodeList children = ND.getChildNodes();
     if(children != null)
       for(int i=0;i<children.getLength();i++)
         printNode(children.item(i),OS);
     wr("</"+name+">",OS);
     break;
   case Node.TEXT_NODE:
     wr(ND.getNodeValue(),OS);
     break;

  }
}

private static void wr(String a,ByteArrayOutputStream os)
{ for(int i=0;i<a.length();i++) os.write(a.charAt(i)); }

/*-- Dom2File ---------------------------------*/
/*-- Converts a DOM into a UTF8 encoded file --*/
/*-----------------------------------------------*/
public static void Dom2File( Document TREE
                           , String   FIL_IN)
throws Exception
{
 String rn = "Dom2File";
 try
 {
  if(isEmpty(FIL_IN)) throw new Exception("No_file");

  String tr  = Dom2String(TREE);
```

```
   FileOutputStream fd = new FileOutputStream(FIL_IN);
   fd.write(tr.getBytes());
   fd.close();
  }
  catch(Exception e) { throw new Exception(rn+":"+e.toString()); }
 }

 /*-- getXML ------------------------*/
 /*-- Returns an XML stored into DB --*/
 /*-- and retrieved by QRY          --*/
 /*-----------------------------------*/
 public static Document[] getXML( Connection CONN
                        , String    QRY)
 throws Exception
 {
  String rn = "lib_ORA.getXML";
  if(CONN==null) throw new Exception(rn+":No CONN");
  if(isEmpty(QRY)) throw new DBException(CONN,rn,":Empty QRY");
  Document[] outp = {};
  String    t_in = "";
  int       ROWS = 0;                //- fetched rows
  int       BSZE = 99;               //- buffer size

  try
  {
   Document[]       buf  = new Document[99];
   PreparedStatement stmt = CONN.prepareStatement(QRY);
   ResultSet        rset = stmt.executeQuery();
   while(rset.next() && ROWS<99)
   {
    OPAQUE  op   = (OPAQUE)rset.getObject(1);
    XMLType xt   = XMLType.createXML(op);
    String  tree = xt.getStringVal();
    tree = tree.trim();
    try { buf[ROWS++] = String2Dom(tree); }
    catch (Exception e) { throw new DBException(CONN,rn,"Wrong tree "+tree+"
"+e.toString()); }
   }
   rset.close();
   stmt.close();

   if(ROWS>0)
   {
    outp = new Document[ROWS];
    for(int j=0;j<ROWS;j++) outp[j]=buf[j];
   }
```

```
    buf=null;
   }
   catch (Exception e) { throw new
DBException(CONN,rn,"[qry="+QRY+"]"+e.toString()); }

   return(outp);
  }

  /*-- getROOT ---------------------------------------*/
  /*-- Returns the root node of the specified TREE --*/
  /*-------------------------------------------------*/
  public static Node getROOT(Document TREE)
  { return(TREE.getDocumentElement()); }

  /*-- isLEAF ----------------------*/
  /*-- Checks if nd_in is a LEAF node --*/
  /*--------------------------------*/
  private static boolean isLEAF(Node nd_in)
  {
   int len=nd_in.getChildNodes().getLength();
   return((len==0)||(len==1 && nd_in.getFirstChild().getNodeType()==3))?true:false;
  }

  /*-- getATTR ---------------------------------------------*/
  /*-- Returns the value of attribute a_in in node node_in --*/
  /*--------------------------------------------------------*/
  private static String getATTR(Node node_in,String a_in)
  {
   if(node_in==null) return(null);
   NamedNodeMap nnm = node_in.getAttributes();
   if(nnm==null) return(null);
   for (int i=0;i<nnm.getLength();i++)
   {
    Node n=nnm.item(i);
    if(n.getNodeName().equals(a_in)) return(n.getNodeValue());
   }
   return(null);
  }

  /*-- insNODE ---------------------------------------------------------*/
  /*-- Inserts a new node of TREE under ND_IN with name=a and value=t_in --*/
  /*--------------------------------------------------------------------*/
  public static Node insNODE(Document TREE,String a,String t_in)
  throws Exception
  { return( insNODE(TREE,getROOT(TREE),a,t_in)); }
```

```
public static Node insNODE(Document TREE,String a,int t_in)
throws Exception
{ return( insNODE(TREE,getROOT(TREE),a,t_in)); }

public static Node insNODE(Document TREE,String a,long t_in)
throws Exception
{ return( insNODE(TREE,getROOT(TREE),a,t_in)); }

public static Node insNODE(Document TREE,Node ND_IN,String a,String t_in)
throws Exception
{
 if(TREE==null)  throw new DBException("Gen.insNODE","NoTree");
 if(ND_IN==null) throw new DBException("Gen.insNODE","NoNode");
 Node outp=null;
 try
 {
  outp=ND_IN.appendChild(TREE.createElement(a));
  if(!isEmpty(t_in)) outp.appendChild(TREE.createTextNode(t_in));
 }
 catch(Exception e)
 { throw new DBException("Gen.insNODE","Cant insert["+a+"] "+e.toString()); }
 if(isEmpty(outp)) throw new DBException("Gen.insNODE","unknown Problem");
 return(outp);
}

public static Node insNODE(Document TREE,Node ND_IN,String a,int t_in)
throws Exception
{ return(insNODE(TREE,ND_IN,a,Integer.toString(t_in))); }

public static Node insNODE(Document TREE,Node ND_IN,String a,long t_in)
throws Exception
{ return(insNODE(TREE,ND_IN,a,Long.toString(t_in))); }

/*-- insATTR ------------------------------------------------*/
/*-- Inserts string v_in for attribute a_in of node node_in --*/
/*-----------------------------------------------------------*/
public static void insATTR(Document doc_in,Node node_in,String a_in,String v_in)
{
 Attr tmp = doc_in.createAttribute(a_in);
 tmp.setNodeValue (v_in);
 node_in.getAttributes().setNamedItem(tmp);
}

public static void insATTR(Node ND_IN,String a_in,String v_in)
{
 Document tree = ND_IN.getOwnerDocument();
```

```
  insATTR(tree,ND_IN,a_in,v_in);
 }

 public static void insATTR(Document doc_in,Node node_in,String a_in,int v_in)
 { insATTR(doc_in,node_in,a_in,Integer.toString(v_in)); }

 public static void insATTR(Node ND_IN,String a_in,int v_in)
 {
  Document tree = ND_IN.getOwnerDocument();
  insATTR(tree,ND_IN,a_in,v_in);
 }

 /*-- updNODE ----------------------------------------------*/
 /*-- Update a Node given its ParentNode and its NodeName --*/
 /*-- Using the ParentNode finds the Node with NodeName   --*/
 /*-- If Node exists updates it else creates it           --*/
 /*-------------------------------------------------------*/
 public static Node updNODE(Document TREE,Node ParentNode,String
NodeName,String VAL)
 throws Exception
 {
  String rn   = "updNODE";
  Node   outp = null;
  if(TREE==null)        throw new DBException(rn,"NoTree");
  if(isEmpty(ParentNode)) throw new DBException(rn,"No ParentNode for
["+VAL+"]");
  if(isEmpty(NodeName))   throw new DBException(rn,"No NodeNode for
["+VAL+"]");
  try
  {
   delNODE(ParentNode,NodeName);
   outp = insNODE(TREE,ParentNode,NodeName,VAL);
  }
  catch(Exception e) { throw new Exception(rn+e.toString()); }
  return(outp);
 }

 public static Node updNODE(Document TREE,Node ParentNode,String NodeName,int
VAL)
 throws Exception
 { return(updNODE(TREE,ParentNode,NodeName,Integer.toString(VAL))); }

 public static Node updNODE(Document TREE,Node ParentNode,String
NodeName,long VAL)
 throws Exception
 { return(updNODE(TREE,ParentNode,NodeName,Long.toString(VAL))); }
```

```
public static Node updNODE(Document TREE,String NodeName,String VAL)
throws Exception
{ return(updNODE(TREE,getROOT(TREE),NodeName,VAL)); }

public static Node updNODE(Document TREE,String NodeName,int VAL)
throws Exception
{ return(updNODE(TREE,getROOT(TREE),NodeName,Integer.toString(VAL))); }

public static Node updNODE(Document TREE,String NodeName,long VAL)
throws Exception
{ return(updNODE(TREE,getROOT(TREE),NodeName,Long.toString(VAL))); }

/*-- upd1 ------------------------------------------------------------*/
/*-- Updates the value of the first line node named <p_in> to <n_in> --*/
/*------------------------------------------------------------------*/
public static void upd1(Document TREE,String p_in,String n_in)
throws Exception
{
 String rn = "upd1";
 if(TREE==null) throw new DBException(rn,"NoTree");
 if(p_in==null) throw new DBException(rn,"Null s_in");
 try
 {
  Node rt = getROOT(TREE);
  delNODE(rt,p_in);
  insNODE(TREE,rt,p_in,n_in);
 }
 catch(Exception e) { throw new Exception(rn+e.toString()); }
}

public static void upd1(Document TREE,String p_in,int n_in)
throws Exception
{ upd1(TREE,p_in,Integer.toString(n_in)); }

public static void upd1(Document TREE,String p_in,long n_in)
throws Exception
{ upd1(TREE,p_in,Long.toString(n_in)); }

/*-- delNODE ---------------*/
/*-- Deletes node <nd_in> --*/
/*-------------------------*/
public static void delNODE(Node nd_in)
{
 if(!isEmpty(nd_in))
 {
```

```
   Node ParentNode = nd_in.getParentNode();
   if(!isEmpty(ParentNode))
   {
    ParentNode.removeChild(nd_in);
    nd_in = null;
   }
  }
}

/*-- delNODE --------------------------------*/
/*-- Deletes node NodeName under ParentNode --*/
/*-----------------------------------------*/
public static void delNODE(Node ParentNode,String NodeName)
{
 if(!isEmpty(ParentNode))
 {
  Node tmp = findNODE(ParentNode,NodeName);
  if(!isEmpty(tmp)) ParentNode.removeChild(tmp);
 }
}

public static void delNODE(Document TREE,String NodeName)
{ delNODE(getROOT(TREE),NodeName); }

/*-- findNODE ------------------------------------------------------*/
/*-- Points to the node with tagname VAR_IN one level under ND_IN --*/
/*------------------------------------------------------------------*/
public static Node findNODE(Node ND_IN,String VAR_IN)
{
 Node outp = null;
 if(!isEmpty(ND_IN) && !isEmpty(VAR_IN))
 {
  int     len = 0;
  NodeList nl  = ND_IN.getChildNodes();
  try{ len=nl.getLength(); } catch(Exception r) {}
  for(int j=0;j<len;j++)
  {
   Node nd = nl.item(j);
   String a=nd.getNodeName();
   if(a.equals(VAR_IN)) outp = nd;
  }
 }
 return(outp);
}

public static Node findNODE(Document TREE_IN,String VAR_IN)
```

```
{ return(findNODE(getROOT(TREE_IN),VAR_IN)); }

/*-- findNODE ------------------------------------*/
/*-- Points to the node one level under node_in --*/
/*-- with tagname var_in and value val_in       --*/
/*-----------------------------------------------*/
public static Node findNODE(Node ND_IN,String VAR_IN,String VAL_IN)
{
 Node outp = null;
 if(!isEmpty(ND_IN))
 {
  if(isEmpty(VAL_IN))
  {
   outp = findNODE(ND_IN,VAR_IN);
  }
  else
  {
   NodeList nl=ND_IN.getChildNodes();
   for(int i=0;i<nl.getLength();i++)
   {
    Node   nd = nl.item(i);
    String a  = nd.getNodeName();
    String b  = readNODE(nd);
    if(a.equals(VAR_IN) && b.equals(VAL_IN)) outp = nd;
   }
  }
 }
 return(outp);
}

public static Node findNODE(Node ND_IN,String NAM_IN,int VAL_IN)
{ return(findNODE(ND_IN,NAM_IN,Integer.toString(VAL_IN))); }

public static Node findNODE(Document TREE,String NAM_IN,String VAL_IN)
{ return(findNODE(getROOT(TREE),NAM_IN,VAL_IN)); }

public static Node findNODE(Document TREE,String NAM_IN,int VAL_IN)
{ return(findNODE(getROOT(TREE),NAM_IN,Integer.toString(VAL_IN))); }

/*-- readNODE ----------------------*/
/*-- Returns the value of node_in --*/
/*-----------------------------------*/
public static String readNODE(Node ND_IN)
{
 String outp = "";
 if(!isEmpty(ND_IN))
```

```
 {
  Node nd=ND_IN.getChildNodes().item(0);
  if(!isEmpty(nd))
  {
   outp = nd.getNodeValue();
   try { outp = URLDecoder.decode(outp,"UTF-8"); } catch(Exception r) { }
  }
 }
 return(outp);
}

/*-- getEIP(TREE) ---------------------------------------*/
/*-- returns the External IP of the Server of this tree --*/
/*-------------------------------------------------------*/
public static String getEIP(Document TREE)
throws Exception
{
 String rn= "Gen.getEIP(TREE)";
 if(isEmpty(TREE)) throw new DBException(rn,"Empty TREE");
 String outp = "";
 try { outp = getNODE(getROOT(TREE),"EIP"); }
 catch(Exception e) { throw new DBException(rn,e.toString()); }
 if(isEmpty(outp)) outp = "";
 return(outp);
}

/*-- getPRC(TREE) ---------------------------------------------------*/
/*-- returns the Session's current form Procedure number from the TREE --*/
/*-------------------------------------------------------------------*/
public static int getPRC(Document TREE)
throws Exception
{
 int    outp = -1;
 String rn   = "Gen.getPRC(TREE)";
 if(!isEmpty(TREE))
 {
  String prc = getNODE(getROOT(TREE),"PRC");
  outp = atoi(prc);
 }
 return(outp);
}

/*-- getSTEP ----------------------------------*/
/*-- returns the STEP of the current procedure --*/
/*---------------------------------------------*/
public static int getSTEP(Document TREE)
```

```
throws Exception
{
  String rn = "Gen.getSTEP";
  String rl = null;
  try { rl=getNODE(getROOT(TREE),"STP"); }
  catch(Exception e) { throw new DBException(rn,e.toString()); }
  if(isEmpty(rl)) throw new DBException(rn,"STEP is empty");
  int outp = atoi(rl);
  if(outp==-1)        throw new DBException(rn,"STEP is NaN:"+rl);
  if(outp<0 || outp>30) throw new DBException(rn,"STEP is wrong:"+outp);
  return(outp);
}

/*-- getSES(TREE) ----------------------*/
/*-- returns the Session of this tree --*/
/*---------------------------------------*/
public static int getSES(Document TREE)
throws Exception
{
  int outp = atoi(getNODE(getROOT(TREE),"SES"));
  if(isEmpty(outp)) throw new DBException("Gen.getSES(TREE)","No Sess in
TREE");
  return(outp);
}

/*-- getNODE ---------------------------------------*/
/*-- Returns the value of NAME_IN under NODE_IN --*/
/*-------------------------------------------------*/
public static String getNODE(Node NODE_IN,String NAME_IN)
throws Exception
{ return(readNODE(findNODE(NODE_IN,NAME_IN))); }

/*-- getNODE -----------------------------------------------*/
/*-- Returns the value of NAME_IN under the root of a TREE --*/
/*-----------------------------------------------------------*/
public static String getNODE(Document TREE_IN,String NAME_IN)
throws Exception
{ return(getNODE(getROOT(TREE_IN),NAME_IN)); }

private static boolean isNULL(String a)
{ return(a==null||a.equals("null")||a.equals("")); }

public static void showbytes(String STR)
throws Exception
{
  for(int j=0;j<getLEN(STR);j++)
```

```
  {
   int c = (int) STR.charAt(j);
   if(c>31)
   {
    char a = (char) c;
    Debug(j+":"+c+" "+a);
   }
   else
    Debug(j+":"+c);
  }
}

/*-- getCurTIME ------------------------*/
/*-- Returns the current time        --*/
/*-- If getSecs thue includes Seconds --*/
/*---------------------------------------*/
public static String getCurTIME(boolean getSecs)
{
 char   ts  = ':';
 Calendar c   = Calendar.getInstance();
 String  hr  = lpad(Integer.toString(c.get(Calendar.HOUR_OF_DAY)),2,"0");
 String  mn  = lpad(Integer.toString(c.get(Calendar.MINUTE)),2,"0");
 String  outp = hr+ts+mn;
 if(getSecs)
 {
  String sc = lpad(Integer.toString(c.get(Calendar.SECOND)),2,"0");
  outp += ts+sc;
 }
 return(outp);
}

/*-- getCurTIME ----------------*/
/*-- Returns the current time --*/
/*-- In HH:MIN format        --*/
/*-----------------------------*/
public static String getCurTIME()
{ return(getCurTIME(false)); }

/*-- getCurDATE ----------------*/
/*-- Returns the current date --*/
/*-----------------------------*/
public static String getCurDATE()
{
 Calendar c = Calendar.getInstance();
 String outp = makeDATE( c.get(Calendar.DAY_OF_MONTH)
                , c.get(Calendar.MONTH)+1
```

```
                    , c.get(Calendar.YEAR));
  return(outp);
}

/*-- makeDATE ------------------------------*/
/*-- Converts 3 integers into DATE format --*/
/*-----------------------------------------*/
private static String makeDATE(int DD,int MM,int YY)
{
  char   dlm  = '/';
  String dd   = lpad(Integer.toString(DD),2,"0");
  String mm   = lpad(Integer.toString(MM),2,"0");
  String yy   = Integer.toString(YY);
  String outp = dd+dlm+mm+dlm+yy;
  return(outp);
}

/*-- folderExists ----------------*/
/*-- Checks if a path is Folder --*/
/*--------------------------------*/
public static boolean folderExists(String DIR)
{
  boolean outp = false;
  try
  {
    File dir = new File(DIR);
    outp = dir.exists() && dir.isDirectory();
  }
  catch(Exception e) { }
  return(outp);
}

/*-- createFolder ----------------------*/
/*-- Create a folder for a given PATH --*/
/*-- if folder exists does nothing    --*/
/*---------------------------------------*/
public static void createFolder(String DIR)
{
  File dir = new File(DIR);
  dir.mkdirs();
}

/*-- writeFILE -------------------------------------*/
/*-- writes a String into File using UTF8 encoding --*/
/*-- !!! do not use UTF8 getBytes()               --*/
/*-------------------------------------------------*/
```

```
public static void writeFILE(String FL,String STR)
throws Exception
{
 String rn="Gen.writeFILE";
 try
 {
  FileOutputStream fd=new FileOutputStream(FL);
  fd.write(STR.getBytes());
  fd.close();
 }
 catch(Exception e) { throw new Exception(rn+":"+e.toString()); }
}

public static boolean isWORD(String S)
{
 boolean outp = true;
 int     len  = getLEN(S);
 if(len==0)
  outp = false;
 else
  for(int j=0;j<len;j++) if(!isASCII(S.charAt(j))) outp=false;
 return(outp);
}

private static boolean isASCII(char c)
{ return(((47<c && c<58)||(64<c && c<91)||(96<c && c<123))?true:false); }

/*-- MakeEmpty ----------------------*/
/*-- fills a String array with EMPTY --*/
/*-----------------------------------*/
public static void MakeEmpty(String[] AR)
{ Arrays.fill(AR,""); }

/*-- MakeEmpty ----------------------*/
/*-- fills an int array with NaN  --*/
/*-----------------------------------*/
public static void MakeEmpty(int[] AR)
{ for(int i=0;i<AR.length;i++) AR[i]=-1; }

/*-- MakeEmpty ----------------------*/
/*-- fills an int array with NaN  --*/
/*-----------------------------------*/
public static void MakeEmpty(String[][] AR)
{ for(int i=0;i<AR.length;i++) for(int j=0;j<AR[i].length;j++) AR[i][j]=""; }

/*-- MakeEmpty ----------------------*/
```

```
/*-- fills an int array with NaN  --*/
/*---------------------------------*/
public static void MakeEmpty(int[][] AR)
{ for(int i=0;i<AR.length;i++) for(int j=0;j<AR[i].length;j++) AR[i][j]=-1; }

/*-- MakeZero ---------------------*/
/*-- fills an int array with zero --*/
/*---------------------------------*/
public static void MakeZero(int[] AR)
{ for(int i=0;i<AR.length;i++) AR[i]=0; }

/*-- String2Array ------------------------------------------------*/
/*-- Returns an array out of a string (S) delimited by char (c) --*/
/*-----------------------------------------------------------------*/
public static String[] String2Array(String S,char c)
{
 String[] outp = {};
 if(!isEmpty(S)) outp = S.split("\\"+c);
 return(outp);
}

/*-- repl -------------------------------------------------*/
/*-- Gets string STR and Replaces string b with string c --*/
/*---------------------------------------------------------*/
public static String repl(String STR,String b,String c)
{
 String outp = STR;
 if(!isEmpty(STR))
 {
  int l1 = getLEN(b);
  if(l1>0)
  {
   int l2 = getLEN(c);
   if(l2==0)
     outp = STR.replace(b,"");
   else
     outp = STR.replace(b,c);
  }
 }
 return(outp);
}

public static String repl(String a,String b,int c)
{ return(repl(a,b,Integer.toString(c))); }

/*-- isDATETIME -------------------------------------------------*/
```

```
/*-- Checks the date_in against the <DD/MM/YYYY hh:mi:ss> format --*/
/*-- If IS_DATE then returns false but loads DT array          --*/
/*-----------------------------------------------------------------*/
public static boolean isDATETIME(String DATE_IN,int DT[])
{
 boolean outp = false;
 MakeEmpty(DT);
 int d1[] = new int[3];
 if(isDATE(DATE_IN,d1))
 {
  DT[0] = d1[0];
  DT[1] = d1[1];
  DT[2] = d1[2];
  DT[3] = 0;
  DT[4] = 0;
  DT[5] = 0;
 }
 else
 {
  int k=DATE_IN.indexOf(" ");
  String dp = getFirst(DATE_IN,k);
  if(isDATE(dp,d1))
  {
   String dt   = getLast(DATE_IN,k+1);
   int    t1[] = new int[3];
   if(isTIME(dt,t1))
   {
    DT[0] = d1[0];
    DT[1] = d1[1];
    DT[2] = d1[2];
    DT[3] = t1[0];
    DT[4] = t1[1];
    DT[5] = t1[2];
    outp = true;
   }
  }
 }
 return(outp);
}

/*-- isDATE ---------------------------------------*/
/*-- Checks date_in against <DD/MM/YYYY> format --*/
/*-----------------------------------------------*/
public static boolean isDATE(String STR_IN)
{
 int dd[]=new int[3];
```

```
 return(isDATE(STR_IN,dd));
}

/*-- isDATE ---------------------------------------*/
/*-- Checks date_in against <DD/MM/YYYY> format --*/
/*-- and returns the dd[] with its parts       --*/
/*-----------------------------------------------*/
public static boolean isDATE(String STR_IN,int dd[])
{
 MakeEmpty(dd);
 if(isEmpty(STR_IN)) return(false);
 String X=STR_IN;
 if(isYEAR(X)) X="02"+"/"+"01"+"/"+X;
 int len=X.length();
 if(len<8 || len>10) return(false);
 String[] tmp = String2Array(X,'/');

 if(tmp.length<3) return(false);
 dd[0] = atoi(tmp[0]); if(!isDAY(dd[0]))   return(false);
 dd[1] = atoi(tmp[1]); if(!isMONTH(dd[1])) return(false);
 dd[2] = atoi(tmp[2]); if(!isYEAR(dd[2]))  return(false);
 try
 {
  if(getDAYSOFMONTH(dd[1],dd[2])<dd[0])   return(false);
 }
 catch(Exception e) { return(false); }

 try{ dd[3]=0; } catch(Exception e) { }
 try{ dd[4]=0; } catch(Exception e) { }
 try{ dd[5]=0; } catch(Exception e) { }
 return(true);
}

/*-- isTIME ---------------------------------*/
/*-- Checks TIME_IN against             --*/
/*-- 1. <hh24:mi:ss>                   --*/
/*-- 2. <hh24:mi> format               --*/
/*-- and returns the dd[] with its parts --*/
/*---------------------------------------*/
public static boolean isTIME(String TIME_IN,int TT[])
{
 MakeEmpty(TT);
 boolean outp = false;
 if(!isEmpty(TIME_IN))
 {
  String[] tmp = String2Array(TIME_IN,':');
```

```
   if(tmp.length>0)
   {
    outp = true;
    try
    {
     TT[0] = atoi(tmp[0]);
     if(!isHOUR(TT[0])) outp = false;
    }
    catch(Exception e) { TT[0]=0; }
    try
    {
     TT[1] = atoi(tmp[1]);
     if(!isMINUTE(TT[1])) outp = false;
    }
    catch(Exception e) { TT[1]=0; }
    try
    {
     TT[2] = atoi(tmp[2]);
     if(!isSECOND(TT[2])) outp = false;
    }
    catch(Exception e) { TT[2]=0; }
   }
  }
  return(outp);
 }

 /*-- addDAYS ------------------------------------------------*/
 /*-- adds the days_in to date_in and returns the result --*/
 /*-- the M_IN (days) can be negative or positive       --*/
 /*-----------------------------------------------------------*/
 public static String addDAYS(String D_IN,int M_IN)
 throws Exception
 {
  String rn   = "Gen.addDAYS";
  if(!isDATE(D_IN)) throw new Exception(rn+"not a date:"+D_IN);
  String qry  = "";
  String outp = "";
  try
  {
   if(M_IN==0)
    outp=D_IN;
   else
   {
    qry="select to_char((to_date('"+D_IN+"','dd/mm/yyyy')+"+M_IN+"),'dd/mm/yyyy')
from dual";
    Connection conn = doConnect();
```

```
      outp = getOne(conn,qry);
      doDisconnect(conn);
     }
    }
    catch(Exception e) { throw new Exception(rn+e.toString()+" qry="+qry); }
    return(outp);
  }

  /*-- addMONTHS ------------------------------------------*/
  /*-- adds the months_in to date_in and returns the result --*/
  /*-- the M_IN (months) can be negative or positive       --*/
  /*-------------------------------------------------------*/
  public static String addMONTHS(String D_IN,int M_IN)
  throws Exception
  {
    String rn   = "Gen.addMONTHS";
    if(!isDATE(D_IN)) throw new Exception(rn+"not a date:"+D_IN);
    String qry  = "";
    String outp = "";
    try
    {
     if(M_IN==0)
       outp=D_IN;
     else
     {
      qry="select
to_char(add_months(to_date('"+D_IN+"','dd/mm/yyyy'),"+M_IN+"),'dd/mm/yyyy') from
dual";
      Connection conn = doConnect();
      outp = getOne(conn,qry);
      doDisconnect(conn);
     }
    }
    catch(Exception e) { throw new Exception(rn+e.toString()+" qry="+qry); }
    return(outp);
  }

  public static boolean isYEAR(String YY_IN)
  {
    int l = getLEN(YY_IN); if(l!=4)       return(false);
    int y = atoi(YY_IN);   if(isEmpty(y)) return(false);
    return(isYEAR(y));
  }

  private static boolean isYEAR(int YY_IN)
  { return((YY_IN>2250 || YY_IN<BASE)?false:true); }
```

```
private static boolean isDAY(int day_in)
{ return((day_in>31 || day_in<1)?false:true); }

private static boolean isMONTH(int month_in)
{ return((month_in>12 || month_in<1)?false:true); }

private static boolean isHOUR(int H_IN)
{ return((H_IN>23 || H_IN<0)?false:true); }

private static boolean isMINUTE(int M_IN)
{ return((M_IN>59 || M_IN<0)?false:true); }

private static boolean isSECOND(int S_IN)
{ return((S_IN>59 || S_IN<0)?false:true); }

/*-- getDAYSOFMONTH ------------------------------------------*/
/*-- Returns the number of days of the specified month and year --*/
/*------------------------------------------------------------*/
private static int getDAYSOFMONTH(int month_in,int year_in)
throws Exception
{
  if(!isMONTH(month_in)) throw new Exception("Gen.getDAYSOFMONTH: Not a
month:"+month_in);
  if(!isYEAR(year_in))   throw new Exception("Gen.getDAYSOFMONTH: Not a
year:"+year_in);
  int outp=ND[month_in-1];
  if(year_in%4==0 && outp==28) outp=29;
  return(outp);
}

/*-- isLaterOrEqual ----------------------------------*/
/*-- Compares d1 and d2 and returns true if dt1>=dt2 --*/
/*--------------------------------------------------*/
public static boolean isLaterOrEqual(String dt1,String dt2)
throws Exception
{ return(isLATER(dt1,dt2) || isEQUAL(dt1,dt2)); }

/*-- isLATER -----------------------------------------*/
/*-- Compares d1 and d2 and returns true if dt1>dt2 --*/
/*--------------------------------------------------*/
public static boolean isLATER(String dt1,String dt2)
throws Exception
{
 String rn="lib_CAL.isLATER";
 int d1[]=new int[6];
```

```
   int d2[]=new int[6];
   try
   {
    if(!isDATE(dt1,d1) && !isDATETIME(dt1,d1)) throw new
DBException(null,rn,"No date{"+dt1+"}");
    if(!isDATE(dt2,d2) && !isDATETIME(dt2,d2)) throw new
DBException(null,rn,"No date{"+dt2+"}");
    if(d1[2]>d2[2]) return(true);
    if(d1[2]==d2[2] && d1[1]>d2[1]) return(true);
    if(d1[2]==d2[2] && d1[1]==d2[1] && d1[0]>d2[0]) return(true);
    if(d1[2]==d2[2] && d1[1]==d2[1] && d1[0]==d2[0] && d1[3]>d2[3]) return(true);
    if(d1[2]==d2[2] && d1[1]==d2[1] && d1[0]==d2[0] && d1[3]==d2[3] &&
d1[4]>d2[4]) return(true);
    if(d1[2]==d2[2] && d1[1]==d2[1] && d1[0]==d2[0] && d1[3]==d2[3] &&
d1[4]==d2[4] && d1[5]>d2[5]) return(true);
   }
   catch(Exception e) { throw new DBException(null,rn,e.toString()); }
   return(false);
  }

  /*-- isEQUAL ---------------------------------------*/
  /*-- Compares d1 and d2 and returns true if dt1=dt2 --*/
  /*-------------------------------------------------*/
  public static boolean isEQUAL(String dt1,String dt2)
  throws Exception
  {
   String rn="lib_CAL.isEQUAL";
   int d1[]=new int[6];
   int d2[]=new int[6];
   try
   {
    if(!isDATE(dt1,d1) && !isDATETIME(dt1,d1)) throw new Exception("No
date{"+dt1+"}");
    if(!isDATE(dt2,d2) && !isDATETIME(dt2,d2)) throw new Exception("No
date{"+dt2+"}");
    if(d1[0]==d2[0] && d1[1]==d2[1] && d1[2]==d2[2] && d1[3]==d2[3] &&
d1[4]==d2[4] && d1[5]==d2[5]) return(true);
   }
   catch(Exception e) { throw new DBException(null,rn,e.toString()); }
   return(false);
  }

  /*-- isEARLIER -------------------------------------*/
  /*-- Compares d1 and d2 and returns true if dt1<dt2 --*/
  /*-------------------------------------------------*/
  public static boolean isEARLIER(String dt1,String dt2)
```

```
 throws Exception
 {
  String rn="lib_CAL.isEARLIER";
  int d1[]=new int[6];
  int d2[]=new int[6];
  try
  {
   if(!isDATE(dt1,d1) && !isDATETIME(dt1,d1)) throw new Exception("No
date{"+dt1+"}");
   if(!isDATE(dt2,d2) && !isDATETIME(dt2,d2)) throw new Exception("No
date{"+dt2+"}");
   if(d1[2]<d2[2]) return(true);
   if(d1[2]==d2[2] && d1[1]<d2[1]) return(true);
   if(d1[2]==d2[2] && d1[1]==d2[1] && d1[0]<d2[0]) return(true);
   if(d1[2]==d2[2] && d1[1]==d2[1] && d1[0]==d2[0] && d1[3]<d2[3]) return(true);
   if(d1[2]==d2[2] && d1[1]==d2[1] && d1[0]==d2[0] && d1[3]==d2[3] &&
d1[4]<d2[4]) return(true);
   if(d1[2]==d2[2] && d1[1]==d2[1] && d1[0]==d2[0] && d1[3]==d2[3] &&
d1[4]==d2[4] && d1[5]<d2[5]) return(true);
  }
  catch(Exception e) { throw new DBException(null,rn,e.toString()); }
  return(false);
 }

 /*-- setDAYS ------------------------------------------------*/
 /*-- Returns the Date that apexei DAYS_IN days from 1/1/1900 --*/
 /*-----------------------------------------------------------*/
 private static String setDAYS(int DAYS_IN)
 throws Exception
 {
  String outp="";
  if(DAYS_IN<0) throw new Exception("lib_CAL.setDAYS: Negative DAYS_IN
["+DAYS_IN+"]");
  try
  {
   int dd = 0;
   int mm = 1;
   int yy = BASE;

   for(int j=0;j<DAYS_IN;j++)
   {
    dd++;
    if(dd>27 && dd>getDAYSOFMONTH(mm,yy))
    {
     dd=1;
     mm++;
```

```
      if(mm>12)
      {
       mm=1;
       yy++;
      }
     }
    }
    outp = makeDATE(dd,mm,yy);
  }
  catch(Exception e) { throw new Exception("lib_CAL.setDAYS: "+e.toString()); }
  return(outp);
}

public static String fix_HIDDEN(String n,String v)
{
  String outp="<INPUT type='HIDDEN' id='"+n+"' name='"+n+"'";
  if(!Gen.isEmpty(v)) outp+=" value='"+v+"'";
  outp+="></INPUT>";
  return(outp);
}

public static String fix_HIDDEN(String n,int v)
{ return(fix_HIDDEN(n,Integer.toString(v))); }

public static String fix_HIDDEN(String n,long v)
{ return(fix_HIDDEN(n,Long.toString(v))); }

/*-- count ----------------------------------------*/
/*-- Counts the existence of char c into a string s --*/
/*-------------------------------------------------*/
public static int count( String s
                       , char   c)
{
  int outp = 0;
  int len  = getLEN(s);
  for(int i=0;i<len;i++) if(s.charAt(i)==c) outp++;
  return(outp);
}

/*-- getArrayTop ----------------------------------*/
/*-- Returns the first empty place for array AR[] --*/
/*-------------------------------------------------*/
public static int getArrayTop(String AR[])
{
  int outp = -1;
  int len  = getLEN(AR);
```

```
 for(int i=0;i<len;i++) if(!isEmpty(AR[i])) outp=i;
 return(outp+1);
}

/*-- getArrayTop ----------------------------------*/
/*-- Returns the first empty place for array AR[] --*/
/*-------------------------------------------------*/
public static int getArrayTop(int AR[])
{
 int outp =-1;
 int len  = getLEN(AR);
 for(int i=0;i<len;i++) if(!isEmpty(AR[i])) outp=i;
 return(outp+1);
}

/*-- getArrayTop ----------------------------------*/
/*-- Returns the first empty place for array AR[] --*/
/*-------------------------------------------------*/
public static int getArrayTop(String AR[][])
{
 int outp = -1;
 int len  = getLEN(AR);
 for(int j=0;j<len;j++) if(!isEmpty(AR[j][0])) outp=j;
 return(outp+1);
}

/*-- makeDIV --------------------------------------*/
/*-- Returns the DIV representation of a Parameter --*/
/*-------------------------------------------------*/
public static String makeDIV(String STR,String CLASS,String TITLE)
{
 String outp="<DIV VALIGN=top ALIGN=left";
 if(!isEmpty(CLASS)) outp += " class="+CLASS;
 if(!isEmpty(TITLE))
 {
  outp += " title='"+TITLE+"'";
  outp += " style=cursor:help;";
 }
 outp+=">"+STR+"</DIV>";
 return(outp);
}

/*-- makeSPAN -------------------------------------*/
/*-- Returns the SPAN representation of a Parameter --*/
/*-------------------------------------------------*/
public static String makeSPAN(String VAL,String CLS)
```

```
{ return(makeSPAN(VAL,CLS,"","",-1)); }

/*-- makeSPAN ---------------------------------------*/
/*-- Returns the SPAN representation of a Parameter --*/
/*-------------------------------------------------*/
public static String makeSPAN(String STR,String CLASS,String TITLE,String URL,int
WIDTH)
{
 String tmp = "";
 String outp="<SPAN VALIGN=top ALIGN=left";
 if(!isEmpty(CLASS)) outp+=" class="+CLASS;

 if(!isEmpty(TITLE))
 {
  outp+=" title='"+TITLE+"'";
  tmp +="cursor:help;";
 }
 else if(!isEmpty(URL))
 {
  outp+=" onclick={WN=f7('"+URL+"',40,40,600,800);}";
  tmp +="cursor:hand;";
 }

 if(WIDTH>0) tmp+="width:"+WIDTH+";";
 if(!isEmpty(tmp)) outp+=" style="+tmp;
 outp+=">"+STR+"</SPAN>";
 return(outp);
}

/*-- Frm.doProblem ----------------------*/
/*-- Reports a problem of a PRC       --*/
/*-- and presents 2 messages to user  --*/
/*-- M1  = 1st message (LABEL_ID)     --*/
/*-- M2  = 2nd message (LABEL_ID)     --*/
/*-----------------------------------*/
public static String doProblem(String RN,int LNG,int M1,int M2)
throws Exception
{
 String  tmp = "";
 int     m1  = M1;   if(isEmpty(m1)) m1=0;
 int     m2  = M2;   if(isEmpty(m2)) m2=0;
 String  mes = Integer.toString(m1)+","+Integer.toString(m2);
 String[] BR  = getLABELS(LNG,mes);
 return(doProblem(RN,LNG,BR[1],BR[0]));
}
```

```
 public static String doProblem( String RN
                       , int    LNG
                       , String M1
                       , String M2)
 throws Exception
 {
  String[] BR  = getLABELS(LNG,"89,140");
  String   ttl = BR[0]+"!!! ("+RN+")";

  String prs = "";
  if(!isEmpty(M1)) prs += "<DIV class=Y> "+M1+"</DIV>";
  if(!isEmpty(M2)) prs += "<DIV class=Y> "+M2+"</DIV>";

  prs += " <INPUT type=SUBMIT name='B' value='"+BR[1]+"'
onClick='history.go(-1)'>";

  String outp = fixHTML(null,prs,ttl,null,null,null,null,null,null);
  return(outp);
 }

 /*-- err ---------------------------------------------------*/
 /*-- Alerts the environment that the Servlet created an error --*/
 /*----------------------------------------------------------*/
 public static void err(HttpServletResponse RES,Document TREE,String RN,String
MESS)
 {
  try
  {
   String prs = err(TREE,RN,MESS,BColor,"black",APL.getHOME());
   Transmit(RES,prs,"Frm.err");
  }
  catch(Exception e) { }
 }

 /*-- err ---------------------------------------------------*/
 /*-- Alerts the environment that the Servlet created an error --*/
 /*----------------------------------------------------------*/
 public static String err( Document TREE
                   , String   RN
                   , String   MESS   //-- Message
                   , String   BCOL   //-- BackGround Color
                   , String   TCOL   //-- TextColor
                   , String   HOME)  //-- Application Home
 {
  String outp = "";
  String t1   = RN+":"+MESS;
```

```
int   uid = -1;
int   prc = -1;
long  ses = -1;
int   stp = -1;

try
{
 String flg = "*";

 if(!isEmpty(TREE))
 {
  try
  {
   uid = atoi(getNODE(getROOT(TREE),"UID"));
   prc = getPRC(TREE);
   ses = getSES(TREE);
   stp = getSTEP(TREE);
  }
  catch(Exception e1) {}

  try
  { Dom2File(TREE,HOME+"err_"+RN+".xml"); }
  catch(Exception e2) { flg = ""; }
 }

 try
 { LogServletError(uid,ses,prc,stp,t1); }
 catch(Exception e3) {}

 String msg = "Sorry! We have a technical problem. Please try later";
 String ttl = "Problem("+RN+flg+")";

 String tmp = makeSPAN(msg,"",t1,"",-1);

 outp = fixHTML( null
           , tmp
           , ttl
           , ""
           , ""
           , ""
           , ""
           , ""
           , ""
           , BCOL
           , TCOL
           , ""
           ,
```

```
                , ""
                , ""
                , ""
                , "");
  }
  catch(Exception e)
  {
   System.out.println("********:"+t1);
   LogServletError(uid,ses,prc,stp,t1);
  }

  return(outp);
}

/*-- LogServletError ----------------------*/
/*-- logs a servlet error into log table --*/
/*-----------------------------------------*/
private static void LogServletError( int  UID
                                   , long SES
                                   , int  PRC
                                   , int  STP
                                   , String MSG)
{
  String tmp = "ServletErr";
  try
  {
   tmp += "+"+
   if(!isEmpty(UID)) tmp += "uid="+UID+",";
   if(!isEmpty(SES)) tmp += "ses="+SES+",";
   if(!isEmpty(PRC)) tmp += "prc="+PRC+",";
   if(!isEmpty(STP)) tmp += "stp="+STP+",";
   tmp = removeLAST(tmp,1);
  }
  catch(Exception e) { tmp += "internal error"; }

  String msg = repl(MSG,"DBException: ","");
  Error(tmp+":"+msg);
}

/*-- Frm.fixHTML ----------------------------*/
/*-- Completes a Normal HTML form         --*/
/*-- PRS   = HTML content to present      --*/
/*-- TTL   = The title of the Form        --*/
/*-- NServ = The Servlet to be called next --*/
/*-----------------------------------------*/
public static String fixHTML( Document TREE
                            , String   PRS
```

```
                    , String  TTL
                    , String  NServ
                    , String  OnFocus
                    , String  OnLoad
                    , String  OnExit
                    , String  enctype
                    , String  EIP)
throws Exception
{
 String outp = "";
 outp = fixHTML( TREE
           , PRS
           , TTL
           , NServ
           , OnFocus
           , "f0();"+OnLoad
           , OnExit
           , enctype
           , EIP
           , BColor
           , "black"
           , SColor
           , "et.css"
           , "et.js"
           , "sorttable.js");
 return(outp);
}

/*-- fixHTML ----------------------------*/
/*-- 1. Completes an HTML form        --*/
/*-- 2. Stores TREE into DB           --*/
/*-----------------------------------*/
/*-- iso-8859-7 = Greek CodePage      --*/
/*-- UTF-8      = UTF CodePage        --*/
/*-----------------------------------*/
public static String fixHTML( Document TREE
                    , String  PRS      //-- HTML content to present
                    , String  TTL      //-- The title of the Form
                    , String  NServ    //-- The Servlet to be called next
                    , String  OnFocus
                    , String  OnLoad
                    , String  OnExit
                    , String  enctype
                    , String  EIP
                    , String  BC       //-- BackGround Color
                    , String  TC       //-- Text Color
```

```
                    , String   SC       //-- Submit BackGround Color
                    , String   CSS      //-- CSS file Name
                    , String   JS1
                    , String   JS2)
 throws Exception
 {
  String rn = "Gen.fixHTML";
  if(isEmpty(PRS)) throw new DBException(rn,":PRS is empty");
  String outp = "";
  try
  {
   String ttl = TTL;
   String tag = "";

   if(TREE!=null)
   {
    iSession ses = new iSession(TREE);
    int      prc = ses.getPRC();
    int      stp = ses.getSTEP();
    ttl += " [prc="+prc+",stp="+stp+"]";
    tag = ses.getTAG();
   }

   outp +=
FormOpen(NServ,ttl,OnFocus,OnLoad,OnExit,enctype,EIP,BC,TC,SC,CSS,JS1,JS2);
   outp += PRS;
   outp += FormClose(tag,NServ);
  }
  catch(Exception e) { throw new Exception(rn+e.toString()); }
  return(outp);
 }

 /*-- FormOpen ----------------*/
 /*-- 1. Opens an HTML form  --*/
 /*---------------------------*/
 private static String FormOpen( String NXT_IN   //-- NextPage
                    , String TTL_IN   //-- Form Title
                    , String OnFocus
                    , String OnLoad
                    , String OnExit
                    , String enctype
                    , String EIP
                    , String BCOL
                    , String TCOL
                    , String SCOL
                    , String CSS
```

```
                        , String JS1
                        , String JS2)
 throws Exception
 {
  String rn   = "Gen.FormOpen";
  String outp = "";
  try
  {
   String  home = APL.getWEBPATH();
   boolean fav  = fileExists(APL.getWEBABS()+"fav.ico");

   outp += "<!DOCTYPE html>";

   outp += "<HTML>";
   outp += "<HEAD>";
   outp += "<META HTTP-EQUIV='Content-Type' CONTENT='text/html;
charset="+CODE_PAGE+"'>";

   if(!Gen.isEmpty(TTL_IN)) outp += "<TITLE>"+TTL_IN+"</TITLE>";

   if(!Gen.isEmpty(CSS))    outp += "<LINK rel=\"stylesheet\" type=\"text/css\"
href='"+home+CSS+"'>";
   if(fav)                  outp += "<LINK rel=\"shortcut icon\" href='"+home+"fav.ico'
type='image/x-icon'/>";
   if(!Gen.isEmpty(JS1))    outp += "<SCRIPT type='text/javascript'
src='"+home+JS1+"'></SCRIPT>";
   if(!Gen.isEmpty(JS2))    outp += "<SCRIPT type='text/javascript'
src='"+home+JS2+"'></SCRIPT>";
   outp += "</HEAD>";

   outp += "<BODY bgcolor='"+BCOL+"' text='"+TCOL+"' leftmargin='0'
topmargin='0'>";  //-- BODY start
   if(!Gen.isEmpty(OnLoad))  outp += " onunload=" onLoad={"+OnLoad+"}";
   if(!Gen.isEmpty(OnFocus)) outp += " onfocus={"+OnFocus+"}";
   outp += ">";                                          //-- BODY end

   if(!Gen.isEmpty(NXT_IN))
   {
    String act = "http://"+EIP+fixServlet(NXT_IN);

    outp += "<FORM name='frm' method='POST' action='"+act+"'";
    if(!Gen.isEmpty(enctype)) outp += " enctype='multipart/form-data'";
    outp += " onsubmit='{GO_ON=false;IS_SB=false;if(!IS_SB &&
DT_OK)GO_ON=true;";
    if(!Gen.isEmpty(OnExit)) outp += "if(GO_ON){"+OnExit+"}";
    outp += "if(GO_ON){IS_SB=true;}return(GO_ON);}'";
```

```
    outp += ">";
   }
  }
  catch(Exception e) { throw new Exception(rn+e.toString()); }
  return(outp);
}

/*-- FormClose -------------------------*/
/*-- 1. Puts the TAG as an HTML param --*/
/*-- 2. Closes an HTML form          --*/
/*---------------------------------------*/
private static String FormClose(String TAG,String NextPage)
throws Exception
{
  String rn   = "Gen.FormClose";
  String outp = "";
  try
  {
   if(!Gen.isEmpty(TAG)) outp += fix_HIDDEN("FIL",TAG);
   outp +="<DIV id='dv'></DIV>";
   if(!Gen.isEmpty(NextPage)) outp += "</FORM>";
   outp += "</BODY></HTML>";
  }
  catch(Exception e) { throw new DBException(rn,e.toString()); }
  return(outp);
}

/*-- fixServlet ------------------------*/
/*-- Complements a Servlet for calling --*/
/*----------------------------------------*/
public static String fixServlet(String SRV)
throws Exception
{ return(APL.getSRVPATH()+SRV+"Servlet"); }

/*-- fileExists ------------------------------------------------*/
/*-- Checks if a File of Folder exists into the Op. system --*/
/*-------------------------------------------------------------*/
public static boolean fileExists(String FL)
{
  File fd = new File(FL);
  return(fd.exists());
}

public static String get_UsrName( Connection CONN
                                , int        UID_IN)
throws Exception
```

```
{
 String rn   = "get_UsrName";
 String outp = "";
 try
 {
  String qry = "select USR_DESCR from CFG_USR where USR_ID="+UID_IN;
  outp = getOne(CONN,qry);
 }
 catch(Exception e) { throw new DBException(CONN,rn,e.toString()); }
 return(outp);
}

/*-- get_GRP ----------------------------*/
/*-- returns the Group of a Treatment --*/
/*-----------------------------------*/
public static int get_GRP( Connection CONN
                         , int       TRT_IN)
throws Exception
{
 String rn   = "get_GRP";
 int    outp = -1;
 try
 {
  String qry = "select GRP_ID from TRT_1 where ID="+TRT_IN;
  outp = atoi(getOne(CONN,qry));
 }
 catch(Exception e) { throw new DBException(CONN,rn,e.toString()); }
 return(outp);
}

/*-- getMenuDesc ----------------------*/
/*-- returns the name of a Menu Item --*/
/*-----------------------------------*/
public static String getMenuDesc( Connection CONN
                                , int       MNU_IN
                                , int       LNG_IN)
throws Exception
{
 String rn   = "getMenuDesc";
 String outp = "";
 try
 {
  String qry = "select LABEL_ID from PRC_0 where MNU_ID="+MNU_IN;
  int lbl = atoi(getOne(CONN,qry));
  outp = get_LABEL(CONN,lbl,LNG_IN);
 }
```

```
 catch(Exception e) { throw new DBException(CONN,rn,e.toString()); }
 return(outp);
}

/*-- getPrcDescr ----------------*/
/*-- returns the name of a PRC --*/
/*-------------------------------*/
public static String getPrcDescr( Connection CONN
                        , int       PRC_IN
                        , int       LNG_IN)
throws Exception
{
 String rn   = "getPrcDescr";
 String outp = "";
 try
 {
  String qry = "select LABEL_ID from PRC_1 where PRC_ID="+PRC_IN;
  int lbl = atoi(getOne(CONN,qry));
  outp = get_LABEL(CONN,lbl,LNG_IN);
 }
 catch(Exception e) { throw new DBException(CONN,rn,e.toString()); }
 return(outp);
}

/*-- addEntry ---------------------------------*/
/*-- Adds a new entry (pair) into an array  --*/
/*-- The array must be of type String[x][2] --*/
/*-------------------------------------------*/
public static String[][] addEntry(String[][] ARR,String PRM,String VAL)
{
 String[][] outp = null;

 if(isEmpty(PRM))
  outp = ARR;
 else if(isEmpty(ARR))
 {
  outp = new String[1][2];
  outp[0][0] = PRM;
  outp[0][1] = VAL;
 }
 else if(isIn(ARR,PRM))
  outp = ARR;
 else
 {
  String vv = isEmpty(VAL)?"":VAL;
  int l0 = getLEN(ARR);            //- initial len
```

```
   int l1 = l0+1;                  //- New len
   String[][] buf = new String[l1][2]; //- Temp holder
   System.arraycopy(ARR,0,buf,0,l0);
   buf[l0][0] = PRM;               //- append data to holder
   buf[l0][1] = vv;
   outp = buf;                     //- move holder to outp
   buf  = null;                    //- send holder to garbage collector
  }
  return(outp);
}

/*-- isIn ----------------------------------------*/
/*-- Finds out if a string exists in an array   --*/
/*-- Warning!! The array may have empty entries --*/
/*-----------------------------------------------*/
private static boolean isIn(String[][] ARR,String STR)
{
 int     pos  = getPOS(ARR,STR);
 boolean outp = (isEmpty(pos))?false:true;
 return(outp);
}

/*-- getPOS --------------------------------*/
/*-- Returns the first occurence of <STR> --*/
/*-- within the NON ordered array <ARR>   --*/
/*-- if not found returns NaN             --*/
/*-----------------------------------------*/
public static int getPOS(String ARR[][],String STR)
{ return(getPOS(ARR,STR,-1)); }

public static int getPOS(String ARR[][],int VAL)
{ return(getPOS(ARR,Integer.toString(VAL),-1)); }

public static int getPOS(String ARR[][],String STR,int LEN)
{
 int    len = getLEN(ARR);
 String val = (!isEmpty(STR))?STR:"";
 for(int j=0;j<len;j++)
 {
  try
  {
   String tmp = (!isEmpty(ARR[j][0]))?ARR[j][0]:"";
   if(tmp.equals(val)) return(j);
  }
  catch(Exception e) { }
 }
```

```
 return(-1);
}

public static int getPOS(String ARR[],String STR)
{
 int    len = getLEN(ARR);
 String val = (!isEmpty(STR))?STR:"";
 for(int j=0;j<len;j++)
 {
  try
  {
   String tmp = (!isEmpty(ARR[j]))?ARR[j]:"";
   if(tmp.equals(val)) return(j);
  }
  catch(Exception e) { }
 }
 return(-1);
}

/*-- getParamValue ------------------------------------*/
/*-- Returns the Value for a given WebForm parameter --*/
/*-- if the param not exists then returns NotFound   --*/
/*-- else returns its value even if null             --*/
/*----------------------------------------------------*/
public static String getParamValue(String[][] webVars,String PARAM)
{
 String outp = "nfd";
 if(webVars==null)
   Gen.Warning("getParamValue:webVars is empty");
 else if(isEmpty(PARAM))
   Gen.Warning("getParamValue:PARAM is empty");
 else
 {
  int pos = getPOS(webVars,PARAM);
  if(!Gen.isEmpty(pos)) outp=webVars[pos][1];
 }
 return(outp);
}

public static void showChars(String STR_IN)
throws Exception
{
 String aa = "";
 int    len = getLEN(STR_IN);

 for(int j=0;j<len;j++)
```

```
 {
  int c = (int) STR_IN.charAt(j);

  aa = "";
  aa += lpad(Integer.toString(j),4," ");
  aa += ": ";
  aa += lpad(Integer.toString(c),3," ");
  if(c>31)
  {
   char a = (char) c;
   aa += " "+a;
  }

  Debug(aa);
  sleep(1);
 }

 Debug("");
}

/*-- ANSI2String ----------------------------*/
/*-- Converts an ANSI string into Java UTF8 --*/
/*-----------------------------------------*/
private static String ANSI2String(String STR)
{
 if(isNULL(STR))
  return("");
 else
 {
  char[] rr = STR.toCharArray();
  for(int j=0;j<rr.length;j++)
  {
   int c = (int) rr[j];
   if(126<c && c<255)
   {
    c = c+720;
    rr[j]=(char) c;
   }
  }
  return(String.valueOf(rr));
 }
}

/*-- String2ANSI ----------------------------*/
/*-- Converts Java UTF8 into Greek ANSI string --*/
/*-----------------------------------------*/
```

```
public static String String2ANSI(String STR)
{
 if(isNULL(STR))
  return("");
 else
 {
  char[] rr = STR.toCharArray();
  for(int j=0;j<rr.length;j++)
  {
   int c = (int) rr[j];
   if(900<c && c<980)
   {
    c = c-720;
    rr[j]=(char) c;
   }
  }
  return(String.valueOf(rr));
 }
}

/*-- LoopBack ---------------------------------*/
/*-- Returns LoopBack string for a given UID --*/
/*-------------------------------------------*/
public static String LoopBack( Connection CONN
                      , int       UID_IN)
throws Exception
{
 String rn   = "Gen.LoopBack";
 String outp = "";
 try
 {
  String[] rr  = get_USR(CONN,UID_IN);
  String   nam = rr[0];
  String   pas = rr[1];
  outp += fix_HIDDEN("NAM",nam);
  outp += fix_HIDDEN("PAS",pas);
 }
 catch(Exception e) { throw new DBException(rn,e.toString()); }
 return(outp);
}

/*-- get_USR ----------------------------*/
/*-- Returns the UID for a given LOGIN --*/
/*-------------------------------------*/
private static String[] get_USR( Connection CONN
                      , int       UID_IN)
```

```
throws Exception
{
 String   rn   = "get_USR ["+UID_IN+"]";
 String[] outp = null;
 try
 {
  if(isEmpty(UID_IN)) throw new Exception("empty_UID");

  String    qry = "select U_LOGIN"
              +      ",U_PASWD"
              + " from CFG_USR"
              + " where USR_ID="+UID_IN
              ;
  String[][] rr  = Gen.Qry2Array(CONN,qry);
  int        len = Gen.getLEN(rr);
  if(len==1)
  {
   outp = new String[2];
   outp[0] = rr[0][0];  //-- login
   outp[1] = rr[0][1];  //-- paswd
  }
  else
   throw new Exception("Wrong_len ["+qry+"]");
 }
 catch(Exception e) { throw new DBException(rn,e.toString()); }
 return(outp);
}

/*-- get_NAM ------------------------*/
/*-- returns the Name of the Human --*/
/*-----------------------------------*/
public static String get_NAM( Connection CONN
                            , int       L15)
throws Exception
{
 String rn   = "Gen.get_NAM";
 String outp = "";
 if(L15>999)
 {
  String qry = "select ONM1 from OBJ_FKEYS_L15 where FKEY_ID="+L15;
  outp = getOne(CONN,qry);
 }
 return(outp);
}

public static String get_CusName( Connection CONN
```

```
                        , int       ID_IN)
throws Exception
{
 String rn   = "Gen.get_CusName";
 String outp = "";
 try
 {
  String    qry = "select V01,L15,L16,NAM from OBJ_HUM where ID="+ID_IN;
  String[][] rr  = Gen.Qry2Array(CONN,qry);
  int       len = Gen.getLEN(rr);
  if(len==1)
  {
   String v01 = rr[0][0];          //-- Surname
   int   l15 = Gen.atoi(rr[0][1]);
   int   l16 = Gen.atoi(rr[0][2]);
   String nam = rr[0][3];          //-- NickName

                    outp += v01;
       if(!isEmpty(nam)) outp += " " + nam;
   else if(l15>9)       outp += " " + get_NAM(CONN,l15);
   else if(l16>9)       outp += " " + get_NAM(CONN,l16);
  }
  else
   throw new Exception("Wrong_len ["+qry+"]");
 }
 catch(Exception e) { throw new DBException(rn,e.toString()); }
 return(outp);
}

public static String get_TRT( Connection CONN
                    , int       TRT_IN
                    , int       LNG_IN)
throws Exception
{
 String rn   = "Gen.get_TRT";
 String outp = "";
 try
 {
  if(TRT_IN<0)
   outp = "";
  else
  {
   String qry = "select LABEL_ID from TRT_1 where ID="+TRT_IN;
   int   lbl = atoi(getOne(CONN,qry));
   outp = get_LABEL(CONN,lbl,LNG_IN);
  }
```

```
  }
  catch(Exception e) { throw new DBException(rn,e.toString()); }
  return(outp);
 }

 /*-- get_AGU ---------------------*/
 /*-- returns the Age Unit descr --*/
 /*-------------------------------*/
 public static String get_AGU( Connection CONN
                      , int      L02_IN
                      , int      LNG_IN)
 throws Exception
 {
  String rn   = "Gen.get_AGU";
  String outp = "";
  try
  {
   if(L02_IN>0)
   {
    String qry = "select LABEL_ID from OBJ_FKEYS_L02 where
FKEY_ID="+L02_IN;
    int    lbl = atoi(getOne(CONN,qry));
    outp = get_LABEL(CONN,lbl,LNG_IN);
   }
  }
  catch(Exception e) { throw new DBException(rn,e.toString()); }
  return(outp);
 }

 /*-- get_ZIP ----------------*/
 /*-- returns the ZIP descr --*/
 /*---------------------------*/
 public static String get_ZIP( Connection CONN
                      , int      L04_IN)
 throws Exception
 {
  String rn   = "Gen.get_ZIP";
  String outp = "";
  try
  {
   if(L04_IN<999)
    outp = "{"+L04_IN+"}";
   else
   {
    String qry = "select DESCR from OBJ_FKEYS_L04 where FKEY_ID="+L04_IN;
    outp = getOne(CONN,qry);
```

```
   }
  }
  catch(Exception e) { throw new DBException(rn,e.toString()); }
  return(outp);
}

/*-- count ----------------------------------------------*/
/*-- Counts the existence of String c into a string s --*/
/*-------------------------------------------------------*/
public static int count( String s
                       , String c)
{
  int outp = 0;
  try
  {
    int len = c.length();
    if(len>0)
    {
      int idx = 0;
      while((idx=s.indexOf(c,idx))!=-1)
      {
        outp++;
        idx += len-1;
      }
    }
  }
  catch(Exception e) { }
  return(outp);
}

/*-- get_LNG ----------------------------*/
/*-- returns the Language of this tree --*/
/*---------------------------------------*/
public static int get_LNG(Document TREE)
throws Exception
{
  String rn   = "Gen.get_LNG";
  int    outp = 1;
  try
  {
    if(isEmpty(TREE))
      outp = 1;
    else
    {
      try { outp = atoi(getNODE(getROOT(TREE),"LNG")); }
      catch(Exception e) { throw new DBException(rn,e.toString()); }
```

```
    }
    if(outp<0) outp = 1;
    if(outp>6) throw new Exception("Wrong_LNG:"+outp);
   }
   catch(Exception e) { }
   return(outp);
  }

  /*-- lookForNode ------------------------------------*/
  /*-- Searches the whole TREE for node named NAME_IN --*/
  /*-- If found returns contents else null          --*/
  /*-------------------------------------------------*/
  public static String lookForNode( Document TREE
                          , String   NAME_IN)
  throws Exception
  {
   String outp = "";
   if(!isEmpty(TREE) && !isEmpty(NAME_IN)) outp =
lookForNode(getROOT(TREE),NAME_IN);
   return(outp);
  }

  /*-- lookForNode ----------------------------*/
  /*-- Searches the TREE under node NODE_IN --*/
  /*-- for a node named NAME_IN             --*/
  /*-- If found returns content else null   --*/
  /*-----------------------------------------*/
  public static String lookForNode( Node   NODE_IN
                          , String NAME_IN)
  throws Exception
  {
   if(isEmpty(NAME_IN)) return(null);
   NodeList nl=((Element)NODE_IN).getElementsByTagName(NAME_IN);
   for(int i=0;i<nl.getLength();i++) return(readNODE(nl.item(i)));
   return(null);
  }

  /*-- c2e ------------------------------------*/
  /*-- Cents -> Euro String                --*/
  /*-- Converts a Euro cents LONG integer  --*/
  /*-- into a EURO string with 2 decimal digits --*/
  /*-------------------------------------------*/
  public static String c2e(int S)
  {
   String outp = "";
   String s    = Integer.toString(Math.abs(S));
```

```
 String a   = "";
 String b   = "";
 int   l   = getLEN(s);

 if(s.equals("0") || s.equals("00"))
 {
  a = "0";
  b = "00";
 }
 else if(l>2)
 {
  a = getFirst(s,l-2);
  a = putDOTS(a);
  b = s.substring(l-2);
 }
 else
 {
  a = "0";
  b = s;
 }

 if(isEmpty(b)) outp=a;
 else outp= a+','+b;

 String sign=(S<0)?"-":"";

 return(sign+outp);
}

private static String putDOTS(String s)
{
 String z = "";
 String a = removeDOTS(s);
 int   l = getLEN(a);
 for(int i=0;i<l;i++)
 {
  if((l-i)%3==0 && i!=0) z+='.';
  z += a.charAt(i);
 }
 return(z);
}

/*-- removeDOTS ----------------------------------*/
/*-- Removes the dot characters (.) from a string --*/
/*------------------------------------------------*/
public static String removeDOTS(String s)
```

```
{
 if(isEmpty(s)) return(null);
 String z   = "";
 int    len = getLEN(s);
 if(len==0) return("");
 for(int i=0;i<len;i++)
 {
  char k=s.charAt(i);
  if(k != '.') z+=k;
 }
 return(z);
}

/*-- getFirst ------------------------------*/
/*-- Returns the characters of string <S> --*/
/*-- starting from point 0 until point P  --*/
/*-- P is 0 offset                        --*/
/*-----------------------------------------*/
private static String getFirst(String S,int P)
{
 String outp = "";
 if(!isEmpty(S) && P>=0)
 {
  int len = getLEN(S);
  int aa  = (len>P)?P:len;
  outp = S.substring(0,aa);
 }
 return(outp);
}

/*-- getLast -------------------------------*/
/*-- Returns the characters of string <S> --*/
/*-- starting from point P until the end  --*/
/*-- P is 0 offset                        --*/
/*-----------------------------------------*/
public static String getLast(String S,int P)
{
 String outp = "";
 if(!isEmpty(S))
 {
  if(P<1)
    outp = S;
  else
  {
   int len = getLEN(S);
   int dif = len-P;
```

```
    if(dif>0) outp = S.substring(P);
   }
  }
  return(outp);
}

/*-- removeLAST ----------------------------------*/
/*-- Removes the last <L> characters of string <S> --*/
/*-- L>0                                          --*/
/*-------------------------------------------------*/
public static String removeLAST(String S,int L)
{ return(getFirst(S,(getLEN(S)-L))); }

/*-- removeFirst ---------------------------------*/
/*-- Removes the first <L> characters of a string <S> --*/
/*-- L>0                                          --*/
/*-------------------------------------------------*/
public static String removeFirst(String S,int L)
{ return(getLast(S,L)); }

/*-- isTEL ---------------------------------------*/
/*-- Returns true id TEL_IN is Telephone Number --*/
/*-----------------------------------------------*/
public static boolean isTEL(String TEL_IN)
{
  boolean outp = false;

      if(TEL_IN.startsWith("00")) outp = true;
  else if(isTelMobile(TEL_IN))    outp = true;
  else if(isTelStatic(TEL_IN))    outp = true;
  else if(TEL_IN.equals("11655")) outp = true;
  else if(TEL_IN.equals("11818")) outp = true;
  else if(TEL_IN.equals("11880")) outp = true;
  else if(TEL_IN.equals("11888")) outp = true;
  else if(TEL_IN.equals("1158"))  outp = true;

  return(outp);
}

/*-- isTelMobile ---------------------*/
/*-- Returns true id TEL_IN is Mobile --*/
/*-------------------------------------*/
public static boolean isTelMobile(String TEL_IN)
{
  boolean outp = false;
  if(getLEN(TEL_IN)==10 && TEL_IN.startsWith("6")) outp = true;
```

```
  return(outp);
 }

 /*-- isTelStatic ----------------------*/
 /*-- Returns true id TEL_IN is Static --*/
 /*---------------------------------------*/
 public static boolean isTelStatic(String TEL_IN)
 {
  boolean outp = false;
  int     len  = getLEN(TEL_IN);
      if(len==10 && TEL_IN.startsWith("2")) outp = true;
  else if(len==11 && TEL_IN.startsWith("8")) outp = true;
  return(outp);
 }
}
```

```
package libs;

/*---- HTML form presenter ----*/
/*-----------------------------*/
import libs.*;
import org.w3c.dom.*;

public class Presenter
{
 private String PRS;    //- Outgoing Presentable content
 private String HDN;    //- Outgoing Hidden content

 public String getPRS() { return(this.PRS); }
 public String getHDN() { return(this.HDN); }

 public Presenter(String a,String b)
 {
  this.PRS = a;
  this.HDN = b;
 }

 /*-- Presenter -------------------------*/
 /*-- Presents PRM array in HTML format --*/
 /*---------------------------------------*/
 public Presenter(Document TREE,String PRM[][],String FRV[])
 throws Exception
 {
  String rn = "Presenter";
```

```
 try
 {
  this.PRS = "";
  this.HDN = "";
  if(Gen.isEmpty(FRV[1])) FRV[1] = "X1";
  work(TREE,PRM,FRV,Gen.getEIP(TREE),Gen.get_LNG(TREE));
 }
 catch(Exception e) { throw new Exception(rn+":"+e.toString()); }
}

/*-- Presenter -------------------------*/
/*-- Presents PRM array in HTML format --*/
/*-- given the HtmlForm               --*/
/*---------------------------------------*/
public Presenter(Document TREE,HtmlForm FRM)
throws Exception
{
 this.PRS = "";
 this.HDN = "";
 if(Gen.isEmpty(FRM.FRV[1])) FRM.FRV[1] = "X1";
 work(TREE,FRM.PRM,FRM.FRV,FRM.EIP,Gen.get_LNG(TREE));
}

public Presenter(HtmlForm FRM)
throws Exception
{
 this.PRS = "";
 this.HDN = "";
 if(Gen.isEmpty(FRM.FRV[1])) FRM.FRV[1] = "X1";
 work(null,FRM.PRM,FRM.FRV,FRM.EIP,FRM.LNG);
}

private void work( Document TREE
                 , String   PRM[][]
                 , String   FRV[]
                 , String   EIP
                 , int      LNG_IN)
throws Exception
{
 String  rn       = "Presenter.work";

 int     lng      = LNG_IN; if(Gen.isEmpty(lng)) lng = 1;
 boolean hasContent = false;            //- presentable content exists
 String  htm      = "";                 //- PRS holder
 String  hidden   = "";                 //- HDN holder
 int     mrg      = 8;                  //- margin of LeftPart
```

```
String[] aa        = Gen.getLABELS(lng,"64,54,65,21");
int      len       = Gen.getArrayTop(PRM);    //- First Empty Param
if(len<1) throw new Exception(rn+" No_Params [len="+len+"]");

FRV[7] = Gen.repl(FRV[7]," "," ");
if(Gen.isEmpty(FRV[3])) FRV[3] = aa[0];
FRV[10] = aa[3];
String TopForm    = FRV[0];
String ContButton = FRV[3];
String ContServ   = FRV[1];
if(!Gen.isEmpty(FRV[11])) FRV[11] += "_"+lng;  //- help form is lng dependent
try
{
 String  onLoad    = "";  //- onload trigger
 String  onex_1    = "";  //- onexit triggers
 String  onex_2    = "";
 String  nonpre    = "";  //- non presentable string
 String  js        = "";  //- javascripts
 boolean emptyCheck = false;
 boolean isFirst    = true;

 for(int j=0;j<len;j++)
 {
  if(!Gen.isEmpty(PRM[j][7])) onex_1+=PRM[j][7];

  if(!Gen.isEmpty(PRM[j][2]))
  {
   js+=fixJS1(PRM[j][2]);
   onex_2 += "chk1("+PRM[j][2]+'\"'+FRV[7].replace(' ','_')+'\"'+");";
  }

  if(!Gen.isEmpty(PRM[j][3]))
  {
   emptyCheck = true;
   onex_2 += "if(GO_ON) GO_ON=chk2("+'\"'+PRM[j][3]+'\"'+");";
  }

  if(!Gen.isEmpty(PRM[j][4])) nonpre += PRM[j][4];
  if(!Gen.isEmpty(PRM[j][8])) hidden += PRM[j][8];
  if(!Gen.isEmpty(PRM[j][6])) onLoad += PRM[j][6];
 }
 if(emptyCheck) js+=fixJS2(FRV[8]);

 String onExit  = onex_1+onex_2;

 /*------------------ Starting to compose HTML page ---------------------*/
```

```
String tmp = "";
/*---------------------------- JavaScript --*/
tmp += js;
/*---------------------------- non presentable info --*/
if(!Gen.isEmpty(nonpre)) tmp += nonpre;

/*---------------------------- Start of presentation --*/
tmp += "<TABLE height='100%' width='100%' cellSpacing='0' cellPadding='0'>";

/*---------------------------- Top Line --*/
tmp += "<TR class='Y' id='u1'><TD height='30' align='center'
colspan='2'>"+APL.get_LOGO();

tmp += "<TR VALIGN=top>";
/*---------------------------- LeftPart -*/
tmp += "<TD>";
tmp += "<TABLE width='100%' cellSpacing='0' cellPadding='0'>";
/*---------------------------- toolbar -*/
tmp += "<TR><TD class='J'>"+get_TOOLBAR(FRV);
/*---------------------------- main part -*/
tmp += "<TR VALIGN='top'><TD>";
tmp += "<TABLE><TR>";
tmp += "<TD width='"+mrg+"'>";
tmp += "<TD>";
if(!Gen.isEmpty(TopForm))
{
 tmp += "<TABLE><TR><TD>"+TopForm;
 tmp += "<TR><TD bgColor='black' height='1'>";
 tmp += "<TR height='"+mrg+"'><TD></TABLE>";
}
tmp += "<TABLE cellSpacing='0' cellPadding='0'>";
tmp += "<TR height='"+mrg+"'><TD></TR>";

for(int j=0;j<len;j++)
{
 if(!Gen.isEmpty(PRM[j][1]))
 {
  tmp += showPARAM(PRM,j);
  hasContent = true;
 }
}

tmp += "</TABLE>";
tmp += "<TD width='"+mrg+"'>";
tmp += "</TABLE>";
tmp += "</TABLE>";
```

```
    tmp += "</TABLE>";

    /*---------------------------- hidden info --*/
    if(!Gen.isEmpty(hidden)) tmp += hidden;

    htm =
Gen.fixHTML(TREE,tmp,FRV[4],ContServ,FRV[6],onLoad,onExit,FRV[13],EIP);
   }
   catch(Exception e) { throw new Exception(rn+":"+e.toString()); }

   if(hasContent)
    this.PRS = htm;
   else
    this.HDN = hidden;
 }

 /*-- get_TOOLBAR ----------------------*/
 /*-- Creates the toolbar out of FRV --*/
 /*-----------------------------------*/
 private static String get_TOOLBAR(String FRV[])
 throws Exception
 {
  String rn   = "get_TOOLBAR";
  String outp = "";
  try
  {
   String cb = FRV[3];          //- Continue button title
   if(!Gen.isEmpty(cb))
   {
    String ab = FRV[5];         //- Alternate route button title
    String af = FRV[9];         //- Alternate route button servlet
    String ft = FRV[4];         //- form title
    String hb = FRV[10];        //- Help button title
    String hf = FRV[11];        //- HTML form to be called on Help

    outp += "<div style='float:left;'>";
    outp += "<INPUT type='SUBMIT' name='B' value='"+cb+"'>";
    if(!Gen.isEmpty(af))
    {
     outp += " ";
     outp += "<INPUT type='SUBMIT' value='"+ab+"'
onclick={frm.action='/servlet/"+af+"Servlet';}>";
    }
    outp += "</div>";

    if(!Gen.isEmpty(hf))
```

```
      {
       outp += "<div style='float:right;'>";
       outp += "<INPUT type='BUTTON' value='"+hb+"'
onclick={WN=f7('/"+hf+".html',40,40,450,650);}>";
       outp += "</div>";
      }

      if(!Gen.isEmpty(ft))
      {
       int len = Gen.getLEN(ft)-Gen.getLEN(hf);
       int ofs = (int) (50+len*0.5);
       outp += "<div style='font-size:18;float:right;width:"+ofs+"%;'>";
       outp += ft;
       outp += "</div>";
      }
     }
    }
    catch(Exception e) { throw new Exception(rn+":"+e.toString()); }
    return(outp);
   }

   /*-- showPARAM ---------------------------------------------------*/
   /*-- presents a piece of input information in a particular order --*/
   /*-- ttl = title of the information                          --*/
   /*-- prm = the input information itself                      --*/
   /*-- inf = the help text related to prm                      --*/
   /*-----------------------------------------------------------------*/
   /*-- Output Layout  <TR><TD>ttl  <TD>prm</TD></TD></TR>        --*/
   /*--              <TR><TD>empty<TD>inf</TD></TD></TR>         --*/
   /*-----------------------------------------------------------------*/
   private String showPARAM(String PRM[][],int P)
   throws Exception
   {
    String rn="showPARAM";
    String prm = PRM[P][1];    //- presentable part
    if(Gen.isEmpty(prm)) throw new Exception(rn+"{PARAM is empty}");
    String outp = "";
    try
    {
     String ttl = PRM[P][11];   //- Param title
     String inf = PRM[P][12];   //- Info string
     String aft = PRM[P][5];    //- After info string

     outp += "<TR><TD>";
     if(Gen.isEmpty(ttl))
      outp += " ";
```

```
    else
      outp += Gen.makeDIV(ttl+" ","Y",null);
    outp+="<TD>"+prm;

    String tmp="";

    if(!Gen.isEmpty(inf))
    {
     String aa = (inf.endsWith("."))?inf:inf+".";
     tmp += "<TR><TD> <TD>"+Gen.makeDIV(aa,"X",null);
    }

    if(!Gen.isEmpty(aft)) tmp +=
"<TR><TD> <TD>"+Gen.makeDIV(aft,"X",null);

    tmp += "<HR>";
    outp += tmp;
   }
   catch(Exception e) { throw new Exception(rn+":"+e.toString()); }

   return(outp);
  }

  /*-- fixJS1 ---------------------------------*/
  /*-- GO_ON must not be initially set here --*/
  /*---------------------------------------*/
  private String fixJS1(String LIMS)
  {
   String els="";
   int    p = Gen.count(LIMS,',')/2;
   String s = "<SCRIPT LANGUAGE=JavaScript>function chk1(";
   for(int i=0;i<p;i++) s+="i"+i+",a"+i+",";
   s += "k1){";
   for(int i=0;i<p;i++)
   {
    s+=els+"if(!f2(i"+i+",a"+i+",k1)) GO_ON=false;";
    els="else ";
   }
   s += "}";
   s += "</SCRIPT>";
   return(s);
  }

  /*-- fixJS2 ---------------------------------*/
  /*-- Produces the JavaScript to check the --*/
  /*-- emptyness of a field                 --*/
```

```
 /*-----------------------------------------*/
 private String fixJS2(String MSG)
 {
  String s = "<SCRIPT LANGUAGE=JavaScript>";
  s += "function chk2(a)";
  s += "{return(f3(a,'"+MSG+"'));}";
  s += "</SCRIPT>";
  return(s);
 }
}
```

package libs;

```
/*---- Form presentation functions ----*/
/*-----------------------------------*/
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import org.w3c.dom.*;

public class HtmlForm
{
 /*-----------------------------------------------------------------*/
 /*-- FRV array holds all the information regarding the Form    --*/
 /*-- FRV[0]  = top part of the form                            --*/
 /*-- FRV[1]  = Servlet or bean to be called                    --*/
 /*-- FRV[2]  = WorkFlow                                       --*/
 /*-- FRV[3]  = Title for normal route (continue) button       --*/
 /*-- FRV[4]  = form title                                     --*/
 /*-- FRV[5]  = Title for alternative route button             --*/
 /*-- FRV[6]  = onfocus trigger                                --*/
 /*-- FRV[7]  = lower limit message for makeNUMERIC            --*/
 /*-- FRV[8]  = is empty message for makeNUMERIC and makeALPHA  --*/
 /*-- FRV[9]  = Form to be called as the alternate route       --*/
 /*-- FRV[10] = Title for help button                          --*/
 /*-- FRV[11] = Form to be called for help button              --*/
 /*-- FRV[12] = Use SSL (0=No, 1=Yes)                          --*/
 /*-- FRV[13] = Use enctype (type=FILE)                        --*/
 /*-- FRV[14] = Current form                                   --*/
 /*-- FRV[15] = Adv on the top                                 --*/
 /*-----------------------------------------------------------------*/

 /*-----------------------------------------------------------------*/
 /*-- PRM array holds all the Parameter's information          --*/
 /*-- PRM[i][0]  = Parameter Name                              --*/
```

```
/*-- PRM[i][1] = representation of Parameter in HTML        --*/
/*-- PRM[i][2] = array of params to be checked for lower limit--*/
/*-- PRM[i][3] = array of params to be checked for emptyness  --*/
/*-- PRM[i][4] = not presentable strings                     --*/
/*-- PRM[i][5] = after Info string                          --*/
/*-- PRM[i][6] = onload trigger                             --*/
/*-- PRM[i][7] = onexit trigger                             --*/
/*-- PRM[i][8] = Hiden Params holder                        --*/
/*-- PRM[i][9] = Column number                              --*/
/*-- PRM[i][10] = Param type                                --*/
/*-- PRM[i][11] = Title (to present)                        --*/
/*-- PRM[i][12] = Info (to present)                         --*/
/*-- where i is the Parameter offset                        --*/
/*-------------------------------------------------------------*/

 private int      MLen  = 13;     /*- PRM[][MLen]       -*/
 private int      FLen  = 16;     /*- FRV[FLen]         -*/
 private int      RLen  = 100;    /*- width of Right part -*/

 private Document  TREE;
 private int      PRC;
 private int      STP;
 public  int      LNG;
 private int      UID;
 public  String[]  FRV = null;
 public  String[][] PRM = null;
 public  String    EIP;

 public HtmlForm(int N)
 {
  this.FRV = new String[this.FLen]; Gen.MakeEmpty(this.FRV);
  this.PRM = new String[N][this.MLen];
 }
}
```

package libs;

```
/*-- iSession ------------------*/
/*-----------------------------*/
import java.util.*;
```

```
import oracle.jdbc.OraclePreparedStatement;
import java.sql.*;
import org.w3c.dom.Document;
import org.w3c.dom.Node;

public class iSession
{
 private static boolean TRACE = false;              //- Trace flag
 private static String  RN    = "iSession";
 private static int     MAX_TREE_STEP = 99999;

 private long     SES=-1;     public long     getSES()    { return(this.SES); }
 private Document TREE;       public Document getTREE()   { return(this.TREE); }
 private String   CIP;        public String   getCIP()    { return(this.CIP); }
 private String   EIP;        public String   getEIP()    { return(this.EIP); }
 private boolean  IS_LOCKED;  public boolean  isLOCKED()  {
return(this.IS_LOCKED); }
 private boolean  IS_EXPIRED; public boolean  isEXPIRED() {
return(this.IS_EXPIRED); }
 private boolean  IS_INVALID; public boolean  isINVALID() {
return(this.IS_INVALID); }
 private int      UID;        public int      getUID()    { return(this.UID); }
 private int      PRC;        public int      getPRC()    { return(this.PRC); }
 private int      CUR_STEP;   public int      getSTEP()   { return(this.CUR_STEP); }
 private int      LNG;        public int      getLNG()    { return(this.LNG); }
 private int      DUR;        public int      getDUR()    { return(this.DUR); }   //-- Duration
in minutes

 private Document MAX_TREE;
 private String   TOK;

 /*-- Session ----------------*/
 /*-- Creates a new Session --*/
 /*--------------------------*/
 public iSession( int    UID
                , int    LNG
                , int    PRC
                , String EIP
                , String CIP)
 throws Exception
 {
  String rn = this.RN+".iSession";
  if(Gen.isEmpty(EIP)) throw new DBException(rn,"No EIP found");
  Connection conn = null;
  InitParams();
  try
```

```
{
 String tok = getUUID();

 conn = Gen.doConnect();                                    //- Transaction START
 long   ses = getNextNum(conn,"SEQ_SES");
 String qry = "select count(*) from SES_0 where SES_ID='"+ses+"'";
 int    rws = Gen.atoi(Gen.getOne(conn,qry));
 if(rws>0)
 {
  String sq0 = "delete from SES_0 where SES_ID='"+ses+"'";
  String sq1 = "delete from SES_1 where SES_ID='"+ses+"'";
  Gen.dbSQL(conn,sq1,rn);       //- We delete in reverse order
  Gen.dbSQL(conn,sq0,rn);
 }
 String sql = "insert into SES_0"
        + "(SES_ID,PRC_ID,LNG_ID,ACL_ID,SES_IP,TOK_ID)"
        + " values("+ses+","+PRC+","+LNG+",'"+UID+"','"+CIP+"','"+tok+"')";
 Gen.dbSQL(conn,sql,rn);
 Gen.dbSQL(conn,"commit",rn);
 Gen.doDisconnect(conn);            //- Transaction END

 this.SES       = ses;
 this.TOK       = tok;
 this.IS_INVALID = false;
 this.IS_LOCKED  = false;
 this.IS_EXPIRED = false;

 this.TREE = Gen.String2Dom("<iSES></iSES>");

 Node rt = Gen.getROOT(this.TREE);

 if(!Gen.isEmpty(LNG)) Gen.insNODE(this.TREE,rt,"LNG",LNG);
 if(!Gen.isEmpty(UID)) Gen.insNODE(this.TREE,rt,"UID",UID);
 if(!Gen.isEmpty(CIP)) Gen.insNODE(this.TREE,rt,"CIP",CIP);

 Gen.insNODE(this.TREE,rt,"PRC",PRC);
 Gen.insNODE(this.TREE,rt,"SES",this.SES);
 Gen.insNODE(this.TREE,rt,"STP",1);
 Gen.insNODE(this.TREE,rt,"EIP",EIP);
 Gen.insNODE(this.TREE,rt,"TOK",this.TOK);

 this.CUR_STEP = 1;
 }
 catch(Exception e) { throw new DBException(conn,rn,e.toString()); }
}
```

```
/*-- Session ----------------------------*/
/*-- Retrieves an Existing Session    --*/
/*-- Called upon the arrival of a form --*/
/*-----------------------------------------*/
public iSession( Connection CONN
            , String    TOK_IN)
throws Exception
{
 String rn = this.RN+".iSession";
 ResultSet rsl  = null;
 InitParams();
 try
 {
  if(Gen.isEmpty(TOK_IN))
   throw new DBException(CONN,rn,"No TOK_IN");
  else
  {
   String  qr0  = "select S_XML from SES_1 where TOK_ID='"+TOK_IN+"'";
   Document rr0[] = Gen.getXML(CONN,qr0);
   if(!Gen.isEmpty(rr0))
   {
    int     cnt  = 0;
    long    ses1 = -1;
    Document tree = rr0[0];
    String  tok0 = getTOK(tree);  if(!TOK_IN.equals(tok0)) throw new
DBException(CONN,rn,"Wrong TOK["+TOK_IN+" "+tok0+"]");
    long    ses0 = Gen.getSES(tree);
    int     stp0 = Gen.getSTEP(tree);

    String  qr1  = "select SES_ID,(sysdate-CRE_DATE)*1440"
             + " from SES_1"
             + " where TOK_ID='"+TOK_IN+"'"
             ;
    Statement stm = CONN.createStatement();
    stm.execute(qr1);
    rsl = stm.getResultSet();
    while(rsl.next())
    {
     ses1 = rsl.getLong(1);
     double dif  = rsl.getDouble(2);
     this.DUR = (int) dif;
     cnt++;
    }

    if(cnt!=1)    throw new DBException(rn,"TAG not found ["+qr1+"]");
    if(ses0!=ses1) throw new DBException(rn,"Wrong ses ["+ses0+","+ses1+"]");
```

```
      if(this.DUR>1500)
       this.IS_EXPIRED = true;
      else
      {
       this.IS_EXPIRED = false;
       this.SES       = ses0;
       this.CUR_STEP   = stp0;
       String qr2 = "select SES_IP,IS_LOCKED,ACL_ID,PRC_ID,LNG_ID"
              + " from SES_0"
              + " where SES_ID="+Long.toString(ses0)
              ;
       String[][] rr2  = Gen.Qry2Array(CONN,qr2);
       int       len2 = rr2.length;
       if(len2==1)
       {
        String cip = rr2[0][0];
        int    flg = Gen.atoi(rr2[0][1]);
        int    uid = Gen.atoi(rr2[0][2]);
        int    prc = Gen.atoi(rr2[0][3]);
        int    lng = Gen.atoi(rr2[0][4]);
        this.TREE      = tree;
        this.CIP       = cip;
        this.IS_LOCKED = (flg==1)?true:false;
        this.UID       = uid;
        this.PRC       = prc;
        this.LNG       = lng;
        this.TOK       = getUUID();
        Gen.updNODE(this.TREE,"TOK",this.TOK);
        updTOK(this.SES,this.TOK);
        this.IS_INVALID = false;
       }
      }
     }
    }
   }
   catch(Exception e) { throw new DBException(CONN,rn,e.toString()); }
   finally { try{ rsl.close(); } catch(Exception x) {} }
}

/*-- iSession -------------------------*/
/*-- Retrieves the Session from TREE --*/
/*-------------------------------------*/
public iSession(Document TREE)
throws Exception
{
```

```
  long ses = Gen.atol(Gen.getNODE(Gen.getROOT(TREE),"SES"));
  if(!Gen.isEmpty(ses))
  {
   this.TREE    = TREE;
   this.SES     = ses;
   this.CUR_STEP = Gen.getSTEP(TREE);
   this.UID     = Gen.atoi(Gen.getNODE(Gen.getROOT(TREE),"UID"));
   this.CIP     = Gen.getNODE(Gen.getROOT(TREE),"CIP");
   this.PRC     = Gen.getPRC(TREE);
   this.EIP     = Gen.getEIP(TREE);
   this.TOK     = getTOK(TREE);
   this.LNG     = Gen.get_LNG(TREE);
  }
 }

 public boolean isLOCKED(Connection CONN)
 throws Exception
 {
  String rn  = this.RN+".isLOCKED";
  boolean outp = false;
  try
  {
   String    qry = "select IS_LOCKED from SES_0 where
SES_ID="+Long.toString(this.SES);
   String[][] rr  = Gen.Qry2Array(CONN,qry);
   int       len = Gen.getLEN(rr);
   if(len==1) outp = (Gen.atoi(rr[0][0])==1);
  }
  catch(Exception e) { throw new DBException(rn,e.toString()); }
  return(outp);
 }

 /*-- getTAG ----------------------------------*/
 /*-- Returns the tag of a Session          --*/
 /*-- Called when a form is about to transmit --*/
 /*-----------------------------------------*/
 public String getTAG()
 throws Exception
 {
  String rn  = this.RN+".getTAG";
  String outp = "";
  try
  {
   long ses = this.SES;     if(Gen.isEmpty(ses)) throw new DBException(rn,"No SES");
   int  stp = this.CUR_STEP; if(Gen.isEmpty(stp)) throw new DBException(rn,"No
STP");
```

```
    outp = this.TOK;
    Store(ses,stp,this.TOK);
   }
   catch(Exception e) { throw new DBException(rn,e.toString()); }
   return(outp);
  }

  public boolean isEmpty()
  { return(Gen.isEmpty(this.SES)); }

  private void updTOK( long   SES_IN
                     , String TOK_IN)
  throws Exception
  {
   String     rn   = this.RN+".updTOK";
   Connection conn = null;
   try
   {
    conn = Gen.doConnect();
    String qry = "select TOK_ID"
               + " from SES_0"
               + " where SES_ID="+Long.toString(SES_IN)
               + " for update NOWAIT"
               ;
    String[][] rr   = Gen.Qry2Array(conn,qry);
    int        len1 = rr.length;
    if(len1==1)
    {
     String tok = rr[0][0];
     if(tok.equals(TOK_IN)) throw new DBException(conn,rn,"TOK not changed for
ses="+SES_IN);
     String sql = "update SES_0"
                + " set TOK_ID='"+TOK_IN+"'"
                + " where SES_ID="+Long.toString(SES_IN)
               ;
     Gen.dbSQL(conn,sql,rn);
     Gen.dbSQL(conn,"commit",rn);
    }
    Gen.doDisconnect(conn);
   }
   catch(Exception e) { throw new DBException(conn,rn,e.toString()); }
  }

  /*-- Store ----------------------*/
  /*-- Unloads TREE into this DB --*/
  /*------------------------------*/
```

```
 private void Store( long   SES
                   , int    STP_IN
                   , String TOK_IN)
 throws Exception
 {
  String rn = "iSession.Store";
  Connection conn = null;
  try
  {
   int    stp = STP_IN;
   String qr0 = "select TOK_ID from SES_0 where SES_ID="+Long.toString(SES);

   conn = Gen.doConnect();
   String tok = Gen.getOne(conn,qr0);
   if(tok.equals(TOK_IN))
   {
    String whr = "where SES_ID="+Long.toString(SES)+" and
STP_NO="+Integer.toString(stp);
    String qry = "select count(*) from SES_1 "+whr;
    int    rw  = Gen.atoi(Gen.getOne(conn,qry));
    if(rw>0)
    {
     String sq0 = "delete from SES_1 "+whr;
     Gen.dbSQL(conn,sq0,rn);
    }

    String tr = Gen.Dom2String(TREE);
    if(!Gen.isEmpty(tr))
    {
     String sq1 = "begin insSES_1("+SES+","+stp+",'"+TOK_IN+"',?);end;";
     OraclePreparedStatement pst =(OraclePreparedStatement)
conn.prepareStatement(sq1);
     pst.setStringForClob(1,tr);
     if(pst.executeUpdate()!=1) throw new DBException(conn,rn,"can not insert");
     Gen.dbSQL(conn,"commit",rn);
    }
    else
     throw new Exception("No_TREE");
   }
   else
    Gen.Report("** Ses="+SES+",stp="+STP_IN+" is GHOST");

   Gen.doDisconnect(conn);
  }
  catch(Exception e) { throw new DBException(conn,rn,e.toString()); }
 }
```

```
/*-- getMaxTREE -------------------------*/
/*-- Returns MAX TREE for this Session --*/
/*-----------------------------------------*/
public Document getMaxTREE( Connection CONN
                          , String     RN_IN)
throws Exception
{
 String   rn   = this.RN+".getMaxTREE["+RN_IN+"]";
 Document outp = null;
 try
 {
  long ses = this.SES;
  int  stp = this.CUR_STEP;
  int  max = getMaxStep(CONN,ses,stp,MAX_TREE_STEP);

  if(max>0)
  {
   String qry = "select S_XML"
              + " from SES_1"
              + " where SES_ID="+Long.toString(ses)
              + " and STP_NO="+Integer.toString(max)
              ;
   Document aa[] = Gen.getXML(CONN,qry);
   if(!Gen.isEmpty(aa)) outp = aa[0];
  }
 }
 catch(Exception e) { throw new DBException(CONN,rn,e.toString()); }
 return(outp);
}

/*-- delMaxTREE -------------------------*/
/*-- Deletes MAX TREE of this Session --*/
/*-----------------------------------------*/
public void delMaxTREE( Connection CONN
                      , String     RN_IN)
throws Exception
{
 String rn = this.RN+".delMaxTREE["+RN_IN+"]";
 try
 {
  long ses = this.SES;
  int  stp = this.CUR_STEP;

  if(stp>0)
  {
```

```
    String sql = "delete from SES_1"
            + " where SES_ID="+Long.toString(ses)
            +   " and STP_NO>"+Integer.toString(stp)
            ;
    Gen.dbSQL(CONN,sql,rn);
    Gen.dbSQL(CONN,"commit",rn);
   }
  }
  catch(Exception e) { throw new DBException(CONN,rn,e.toString()); }
}

/*-- getMaxStep ------------------------------*/
/*-- Returns the position (step) of Max TREE --*/
/*------------------------------------------*/
private static int getMaxStep( Connection CONN
                     , long     SES_IN
                     , int      STP_IN
                     , int      MAX_IN)
throws Exception
{
 String   rn  = "getMaxStep";
 int      outp = -1;
 Statement stm  = null;
 ResultSet rsl  = null;
 try
 {
  String qry = "select min(STP_NO),max(STP_NO)"
            + " from SES_1"
            + " where SES_ID="+Long.toString(SES_IN)
            +   " and STP_NO>"+Integer.toString(STP_IN)
            ;
   stm = CONN.createStatement();
   stm.execute(qry);
   rsl = stm.getResultSet();
   if(rsl.next())
   {
    int mn = rsl.getInt(1);
    int mx = rsl.getInt(2);
    if(mx==MAX_IN)
      outp = mx;
    else
      outp = mn;
   }
  }
  catch(Exception e)
  { throw new DBException(CONN,rn,e.toString()); }
```

```
  finally
  {
   try { rsl.close(); } catch(Exception x) {}
   try { stm.close(); } catch(Exception x) {}
  }
  return(outp);
 }

 /*-- StoreMax ----------------------*/
 /*-- Unloads MAX TREE into this DB --*/
 /*----------------------------------*/
 public void StoreMax(Document TREE)
 throws Exception
 {
  String rn = this.RN+".StoreMax";
  try
  {
   long   ses = this.SES; if(Gen.isEmpty(ses)) throw new DBException(rn,"No SES");
   int    stp = MAX_TREE_STEP;
   String whr = "where SES_ID="+Long.toString(ses)+" and
STP_NO="+Integer.toString(stp);
   String qry = "select count(*) from SES_1 "+whr;
   Connection conn = Gen.doConnect();
   int    rw  = Gen.atoi(Gen.getOne(conn,qry));
   if(rw>0)
   {
    String sq0 = "delete from SES_1 "+whr;
    Gen.dbSQL(conn,sq0,rn);
   }

   String tr  = Gen.Dom2String(TREE);
   String sq1 = "begin insSES_1("+ses+","+stp+",'aaa',?);end;";
   OraclePreparedStatement pst = (OraclePreparedStatement)
conn.prepareStatement(sq1);
   pst.setStringForClob(1,tr);
   if(pst.executeUpdate()!=1) throw new Exception("can not insert");
   Gen.dbSQL(conn,"commit",rn);
   Gen.doDisconnect(conn);
  }
  catch(Exception e) { throw new Exception(rn+e.toString()); }
 }

 public int doStep(int STP)
 throws Exception
 {
  String rn     = "iSession.doStep";
```

```
  int    cur_stp = this.CUR_STEP;
  int    nxt_stp = cur_stp + STP;
  if(nxt_stp<1) throw new DBException(rn,"stp<1
[s="+STP+",c="+cur_stp+",n="+nxt_stp+"]");
  this.CUR_STEP = nxt_stp;
  Gen.upd1(this.TREE,"STP",nxt_stp);
  return(nxt_stp);
 }

 /*-- Lock ----------------------------------------*/
 /*-- Locks the Session so it can NOT be used again --*/
 /*------------------------------------------------*/
 /*-- Warning!!! Do not Commit here              --*/
 /*------------------------------------------------*/
 public void Lock(Connection CONN)
 throws Exception
 {
  String rn = this.RN+".Lock";
  try
  {
   String sql = "update SES_0 set IS_LOCKED=1 where SES_ID="'+this.SES+'"";
   Gen.dbSQL(CONN,sql,rn);
  }
  catch(Exception e) { throw new DBException(CONN,rn,e.toString()); }
 }

 private void InitParams()
 {
  this.SES       = -1;
  this.CIP       = "";
  this.EIP       = "";
  this.IS_LOCKED  = true;
  this.IS_INVALID = true;
  this.UID       = -1;
  this.PRC       = -1;
  this.CUR_STEP   = -1;
  this.TOK       = "";
  this.TREE      = null;
  this.MAX_TREE   = null;
 }

 /*-- get_L01 ------------------------*/
 /*-- returns the Sex in this TREE --*/
 /*-----------------------------------*/
 public int get_L01()
 throws Exception
```

```
{
 int outp = Gen.atoi(Gen.getNODE(Gen.getROOT(this.TREE),"L01"));
 return(outp);
}

/*-- get_L02 ----------------------------*/
/*-- returns the Age Unit of the Human  --*/
/*-----------------------------------------*/
public int get_L02()
throws Exception
{
 int outp = Gen.atoi(Gen.getNODE(Gen.getROOT(this.TREE),"L02"));
 return(outp);
}

/*-- get_L04 ----------------------------*/
/*-- returns the Zip code of the Human  --*/
/*-----------------------------------------*/
public int get_L04()
throws Exception
{
 int outp = Gen.atoi(Gen.getNODE(Gen.getROOT(this.TREE),"L04"));
 return(outp);
}

/*-- get_L15 ------------------------*/
/*-- returns the Name of the Human --*/
/*-----------------------------------*/
public int get_L15()
throws Exception
{
 int outp = Gen.atoi(Gen.getNODE(Gen.getROOT(this.TREE),"L15"));
 return(outp);
}

/*-- get_V01 ----------------------------*/
/*-- returns the SurName of the Human --*/
/*-------------------------------------*/
public String get_V01()
throws Exception
{
 String outp = Gen.getNODE(Gen.getROOT(this.TREE),"V01");
 return(outp);
}

/*-- get_V20 ----------------------*/
```

```
/*-- returns the Tel of the Human --*/
/*---------------------------------*/
public String get_V20()
throws Exception
{
 String outp = Gen.getNODE(Gen.getROOT(this.TREE),"V20");
 return(outp);
}

/*-- get_V22 -------------------------------*/
/*-- returns the Mobile Tel of the Human --*/
/*---------------------------------------*/
public String get_V22()
throws Exception
{
 String outp = Gen.getNODE(Gen.getROOT(this.TREE),"V22");
 return(outp);
}

/*-- get_V04 ---------------------------*/
/*-- returns the Address of the Human --*/
/*-------------------------------------*/
public String get_V04()
throws Exception
{
 String outp = Gen.getNODE(Gen.getROOT(this.TREE),"V04");
 return(outp);
}

/*-- get_V06 -------------------------*/
/*-- returns the email of the Human --*/
/*-----------------------------------*/
public String get_V06()
throws Exception
{
 String outp = Gen.getNODE(Gen.getROOT(this.TREE),"V06");
 return(outp);
}

/*-- get_NAM -----------------------------*/
/*-- returns the NickName of the Human --*/
/*---------------------------------------*/
public String get_NAM()
throws Exception
{
 String outp = Gen.getNODE(Gen.getROOT(this.TREE),"NAM");
```

```
 return(outp);
}

/*-- getNextNum ---------------------------------*/
/*-- Gets the next number from sequence TNAME --*/
/*-- Can not be Replicable                    --*/
/*---------------------------------------------*/
private static long getNextNum(Connection CONN,String NAME_IN)
throws Exception
{
 String rn   = "isession.getNextNum["+NAME_IN+"]";
 long   outp = 0;
 try
 {
  String qry = "select "+NAME_IN+".nextval from dual";
  String tmp = Gen.getOne(CONN,qry);
  outp = Gen.atol(tmp);
  if(outp<1) throw new Exception("wrong result ["+tmp+"]");
 }
 catch(Exception e) { throw new DBException(CONN,rn,e.toString()); }
 return(outp);
}

/*-- getUUID ----------------*/
/*-- Returns a UUID string --*/
/*--------------------------*/
private static String getUUID()
throws Exception
{
 String outp = "";
 UUID a1=UUID.randomUUID();
 String xx = a1.toString();
 outp = xx.replace("-","").toUpperCase();
 return(outp);
}
```

```
 /*-- getTOK(TREE) --------------------*/
 /*-- returns the Token of this tree --*/
 /*-----------------------------------*/
 private static String getTOK(Document TREE)
 throws Exception
 {
  String outp = Gen.getNODE(Gen.getROOT(TREE),"TOK");
  if(Gen.isEmpty(outp)) throw new DBException("iSession.getTOK(TREE)","No TOK
in TREE");
  return(outp);
 }
}
```

```
package FrmBeans;

import java.util.*;
import java.sql.*;
import org.w3c.dom.*;
import libs.*;

public interface FormBean
{
 String getVars(String[][] WebVars,Connection CONN,Document TREE) throws
Exception;

 Presenter work(Connection CONN,Document TREE) throws Exception;
}
```

```
package FrmBeans;

/*-- HUMANSbean ----------------------------------*/
/*-- PRC=221 Solution DownPayment via Bank deposit --*/
/*-- Solution selection                  --*/
/*-------------------------------------------------*/
import libs.*;
import java.net.*;
import java.sql.*;
import org.w3c.dom.*;

public class HUMANSbean implements FormBean
{
 final private static String PP = "HUM";
```

```
public Presenter work( Connection CONN
                     , Document   TREE)
throws Exception
{
 String    rn = "HUMANS";
 Presenter outp = null;
 try
 {
  iSession ses = new iSession(TREE);
  int      lng = ses.getLNG();
  int      prc = ses.getPRC();
  String   eip = ses.getEIP(); if(Gen.isEmpty(eip)) throw new Exception("No EIP");
  String   des = Gen.getPrcDescr(CONN,prc,lng);

  String prs = getPRS(CONN,PP,lng);
  int    cnt = Gen.count(prs,""+PP+"");
  if(cnt==0)
  {
   String e1= Gen.doProblem(rn,lng,"No Lines",des);
   outp = new Presenter(e1,null,null);
  }
  else
  {
   HtmlForm frm  = new HtmlForm(1);
   frm.LNG       = lng;
   frm.EIP       = eip;
   frm.FRV[4]    = des + " ("+cnt+")";
   frm.FRV[14]   = rn;
   frm.PRM[0][0] = PP;
   frm.PRM[0][1] = prs;
   outp = new Presenter(TREE,frm);
  }
 }
 catch(Exception e) { throw new DBException(CONN,rn,e.toString()); }
 return(outp);
}

public String getVars( String[][] WebVars
                     , Connection CONN
                     , Document   TREE)
throws Exception
{
 String rn = "HUMANS.getVars";
 try
 {
  String tmp = Gen.getParamValue(WebVars,PP);
```

```
   Gen.insNODE(TREE,PP,tmp);
  }
  catch(Exception e) { throw new DBException(CONN,rn,e.toString()); }
  return(null);
}

/*-- getPRS --------------------*/
/*-- Rerurns presentable Part --*/
/*-----------------------------*/
private static String getPRS( Connection CONN
                            , String    PRM_IN
                            , int       LNG_IN)
throws Exception
{
  String rn  = "HUMANS.getPRS";
  String outp = "";
  try
  {
    int      cnt = 0;
    String   prs = "";
    String   qry = "select ID,V01,L15,L02,V20,V22"
                 + " from OBJ_HUMAN"
                 ;
    String[][] rr  = Gen.Qry2Array(CONN,qry);
    int len = Gen.getLEN(rr);
    for(int j=0;j<len;j++)
    {
     int    id  = Gen.atoi(rr[j][0]);
     String v01 = rr[j][1];
     int    l15 = Gen.atoi(rr[j][2]);
     int    l02 = Gen.atoi(rr[j][3]);
     String v20 = rr[j][4]; if(Gen.isEmpty(v20)) v20 = " ";
     String v22 = rr[j][5]; if(Gen.isEmpty(v22)) v22 = " ";

     prs += "<TR>";
     prs += "<TD><INPUT name="'+PRM_IN+"' type=radio value="'+id+"'";
     if(cnt==0) prs += " CHECKED";
     prs += ">";
     prs += "<TD>"+v01;
     prs += "<TD>"+Gen.getNAM(CONN,l15);
     prs += "<TD>"+Gen.get_AGU(CONN,l02,LNG_IN);
     prs += "<TD>"+v20;
     prs += "<TD>"+v22;
     prs += "</TR>";
     cnt++;
    }
```

```
    if(cnt>0)
    {
     outp += "<TABLE border=1 class='sortable'>";

     outp += "<THEAD>";
     outp += "<TR>";
     outp += "<TH> ";
     outp += "<TH>Επώνυμο";    //-- Επώνυμο
     outp += "<TH>Ονομα";      //-- Ονομα
     outp += "<TH>"+Gen.get_LABEL(CONN,2016,LNG_IN);
     outp += "<TH>"+Gen.get_LABEL(CONN,2020,LNG_IN);
     outp += "<TH>"+Gen.get_LABEL(CONN,2022,LNG_IN);
     outp += "</TR>";
     outp += "</THEAD>";

     outp += "<TBODY>";
     outp += prs;
     outp += "</TBODY>";

     outp += "</TABLE>";
    }
   }
   catch(Exception e) { throw new DBException(rn,e.toString()); }
   return(outp);
  }
}
```

package FrmBeans;

```
/*-- Y201_FNbean ----------*/
/*-- Loads a new Human to DB  --*/
/*-----------------------------*/
import libs.*;
import java.util.*;
import java.sql.Connection;
import org.w3c.dom.*;

public class Y201_FNbean implements FormBean
{
 public Presenter work( Connection CONN
                , Document   TREE)
 throws Exception
 {
  String   rn   = "Y201_FN";
```

```
  Presenter outp = null;
  try
  {
   iSession ses = new iSession(TREE);
   int     prc = ses.getPRC();     if(prc!=201) throw new Exception(rn+"Wrong_PRC
[prc="+prc+"]");
   String  eip = ses.getEIP();
   int     lng = ses.getLNG();
   int     uid = ses.getUID();
   String  des = Gen.getPrcDescr(CONN,prc,lng);

   transaction(CONN,ses);       //= Transaction
   ses.Lock(CONN);
   Gen.dbSQL(CONN,"commit",rn);  //= Transaction End

   HtmlForm frm = new HtmlForm(1);
   frm.LNG      = lng;
   frm.EIP      = eip;
   frm.FRV[1]   = "M1";
   frm.FRV[4]   = des;
   frm.FRV[14]  = rn;
   frm.PRM[0][0] = "_DUM";
   frm.PRM[0][1] = "OK!"+Gen.LoopBack(CONN,uid);
   outp = new Presenter(TREE,frm);
  }
  catch(Exception e) { throw new DBException(CONN,rn,e.toString()); }
  return(outp);
 }

 public String getVars(String[][] WebVars,Connection CONN,Document TREE)
 throws Exception
 { return(null); }

 /*-- transaction ----------------------*/
 /*-------------------------------------*/
 private static void transaction( Connection CONN
                        , iSession   SES_IN)
 throws Exception
 {
  String rn = "Y201_FN.transaction";
  try
  {
   int   id  = getMAX(CONN);
   String v01 = SES_IN.get_V01(); v01 = Gen.String2ANSI(v01);
   int   l15 = SES_IN.get_L15();
   int   l02 = SES_IN.get_L02(); //-- Age unit
```

```
    int   l01 = SES_IN.get_L01(); //-- sex
    String v20 = SES_IN.get_V20();
    String v22 = SES_IN.get_V22(); //-- Mobile tel
    String v04 = SES_IN.get_V04(); v04 = Gen.String2ANSI(v04); //-- Address
    int   l04 = SES_IN.get_L04(); //-- Zip
    String v06 = SES_IN.get_V06(); //-- email

    String sql = "insert into
OBJ_HUMAN(ID,V01,L15,L02,L01,V20,V22,V04,L04,V06)"
           + "
values("+id+",'"+v01+"',"+l15+","+l02+","+l01+",'"+v20+"','"+v22+"','"+v04+"',"+l04+",
'"+v06+"')"
           ;
    Gen.dbSQL(CONN,sql,rn);
   }
   catch(Exception e) { throw new DBException(rn,e.toString()); }
  }

  private static int getMAX(Connection CONN)
  throws Exception
  {
   String rn  = "getMAX";
   int   outp = -1;
   String qry="select nvl(max(ID),0) from OBJ_HUMAN";
   outp = Gen.atoi(Gen.getOne(CONN,qry));
   if(Gen.isEmpty(outp)) throw new DBException(CONN,rn,"No outp");
   return(outp+1);
  }
}
```

```
package FrmBeans;

/*-- Y221_FNbean ----------*/
/*-- Loads a new Human to DB  --*/
/*------------------------------*/
import libs.*;
import java.net.*;
import java.util.*;
import java.sql.Connection;
import org.w3c.dom.*;

public class Y221_FNbean implements FormBean
{
 public Presenter work( Connection CONN
                , Document  TREE)
 throws Exception
```

```
 {
  String   rn  = "Y221_FN";
  Presenter outp = null;
  try
  {
   iSession ses = new iSession(TREE);
   int     prc = ses.getPRC();     if(prc!=221) throw new Exception(rn+"Wrong_PRC
[prc="+prc+"]");
   String  eip = ses.getEIP();
   int     lng = ses.getLNG();
   int     uid = ses.getUID();
   String  des = Gen.getPrcDescr(CONN,prc,lng);
   int     id  = Gen.atoi(Gen.lookForNode(TREE,"HUM"));

   if(id<0)
   {
    String e1 = Gen.doProblem(rn,lng,des,"No Selection");
    outp = new Presenter(e1,null,null);
   }
   else
   {
    String  prs = getPRS(CONN,id,lng);
    HtmlForm frm = new HtmlForm(1);
    frm.LNG      = lng;
    frm.EIP      = eip;
    frm.FRV[1]   = "M1";
    frm.FRV[4]   = des;
    frm.FRV[14]  = rn;
    frm.PRM[0][0] = "_DUM";
    frm.PRM[0][1] = prs+Gen.LoopBack(CONN,uid);
    outp = new Presenter(TREE,frm);
   }
  }
  catch(Exception e) { throw new DBException(CONN,rn,e.toString()); }
  return(outp);
 }

 public String getVars(String[][] WebVars,Connection CONN,Document TREE)
 throws Exception
 { return(null); }

 /*-- getPRS --------------------*/
 /*-- Rerurns presentable Part --*/
 /*-----------------------------*/
 private static String getPRS( Connection CONN
                     , int       HUM_IN
```

```
                    , int       LNG_IN)
throws Exception
{
 String rn   = "HUMANS.getPRS";
 String outp = "";
 try
 {
  String    prs = "";
  String    qry = "select V01,L15,L02,L01,V20,V22,V04,L04,V06"
              + " from OBJ_HUMAN"
              + " where ID="+HUM_IN
              ;
  String[][] rr  = Gen.Qry2Array(CONN,qry);
  int len = Gen.getLEN(rr);

  if(len==1)
  {
   String v01 = rr[0][0];
   int   l15 = Gen.atoi(rr[0][1]);
   int   l02 = Gen.atoi(rr[0][2]);                    //-- Age unit
   int   l01 = Gen.atoi(rr[0][3]);                    //-- sex
   String v20 = rr[0][4];       if(Gen.isEmpty(v20)) v20 = " "; //-- Tel
   String v22 = rr[0][5];       if(Gen.isEmpty(v22)) v22 = " "; //-- Mobile
   String v04 = rr[0][6];                           //-- Address
   int   l04 = Gen.atoi(rr[0][7]);                    //-- Zip
   String v06 = rr[0][8];                           //-- email

   if(Gen.isEmpty(v06))
    v06 = " ";
   else
    v06 = "<a href='mailto:"+v06+"">"+v06+"</a>";

   outp += "<TABLE border=1>";
   outp += "<TR><TD>Ονοματεπώνυμο<TD>"+v01+"
"+Gen.getNAM(CONN,l15)+"</TR>";
   outp +=
"<TR><TD>"+Gen.get_LABEL(CONN,2016,LNG_IN)+"<TD>"+Gen.get_AGU(CON
N,l02,LNG_IN)+"</TR>";
   outp +=
"<TR><TD>"+Gen.get_LABEL(CONN,2020,LNG_IN)+"<TD>"+v20+"</TR>";
   outp +=
"<TR><TD>"+Gen.get_LABEL(CONN,2022,LNG_IN)+"<TD>"+v22+"</TR>";
   outp +=
"<TR><TD>"+Gen.get_LABEL(CONN,2023,LNG_IN)+"<TD>"+v06+"</TR>";

   if(!Gen.isEmpty(v04))
```

```
      {
       String zip = "";
       if(l04<999)
        zip = " ";
       else
        zip = l04+" "+Gen.get_ZIP(CONN,l04);
       outp +=
"<TR><TD>"+Gen.get_LABEL(CONN,2028,LNG_IN)+"<TD>"+v04+"</TR>";
       outp +=
"<TR><TD>"+Gen.get_LABEL(CONN,2026,LNG_IN)+"<TD>"+zip+"</TR>";
      }
      outp += "</TABLE>";
    }
   }
   catch(Exception e) { throw new DBException(rn,e.toString()); }
   return(outp);
  }
}
```

package FrmBeans;

```
/*-- Y301_03bean ----------*/
/*-- Rendevouz presentation --*/
/*--------------------------*/
import libs.*;
import java.sql.*;
import org.w3c.dom.*;

public class Y301_03bean implements FormBean
{
 final private static String PP0 = "TIM";

 public Presenter work( Connection CONN
                      , Document   TREE)
 throws Exception
 {
  String   rn = "Y301_03";
  Presenter outp = null;
  try
  {
   iSession ses = new iSession(TREE);
   int     lng = ses.getLNG();
   int     prc = ses.getPRC();                    if(prc!=301)       throw new
Exception("[Wrong prc="+prc+"]");
```

```
    String   eip = ses.getEIP();                    if(Gen.isEmpty(eip)) throw new
Exception("No EIP");
    String   des = Gen.getPrcDescr(CONN,prc,lng);

    int      hum = Gen.atoi(Gen.lookForNode(TREE,"HUM")); if(Gen.isEmpty(hum))
throw new Exception("No HUM");
    int      usr = Gen.atoi(Gen.lookForNode(TREE,"USR")); if(Gen.isEmpty(usr)) throw
new Exception("No USR");
    String   dat = Gen.lookForNode(TREE,"DAT");          if(Gen.isEmpty(dat)) throw
new Exception("No DAT");

    String prs = getPRS(CONN,PP0,usr,dat,lng);
    if(Gen.isEmpty(prs))
    {
     String e1= Gen.doProblem(rn,lng,"No Lines",des);
     outp = new Presenter(e1,null,null);
    }
    else
    {
     HtmlForm frm   = new HtmlForm(1);
     frm.LNG        = lng;
     frm.EIP        = eip;
     frm.FRV[0]     = Gen.getUsrName(CONN,usr)+", "+dat;
     frm.FRV[4]     = des + " ("+Gen.get_HumanName(CONN,hum)+")";
     frm.FRV[14]    = rn;
     frm.PRM[0][0]  = PP0;
     frm.PRM[0][1]  = prs;
     frm.PRM[0][11] = Gen.get_LABEL(CONN,3016,lng);
     frm.PRM[0][12] = Gen.get_LABEL(CONN,3017,lng);

     outp = new Presenter(TREE,frm);
    }
   }
   catch(Exception e) { throw new DBException(CONN,rn,e.toString()); }
   return(outp);
 }

 public String getVars( String[][] WebVars
                      , Connection CONN
                      , Document   TREE)
 throws Exception
 {
  String rn = "Y301_03.getVars";
  try
  {
   String tmp = Gen.getParamValue(WebVars,PP0);
```

```
    Gen.insNODE(TREE,PP0,tmp);
   }
   catch(Exception e) { throw new DBException(CONN,rn,e.toString()); }
   return(null);
  }

  /*-- getPRS ----------------*/
  /*-- Rerurns presentable Part --*/
  /*-----------------------------*/
  private static String getPRS( Connection CONN
                      , String    PRM_IN
                      , int       USR_IN
                      , String    DAT_IN
                      , int       LNG_IN)
  throws Exception
  {
   String rn   = "Y301_03.getPRS";
   String outp = "";
   try
   {
    String qry = "select RDZ_DAT,RDZ_DUR"
           + " from OBJ_RDZ"
           + " where USR_ID="+USR_IN
           + " and RDZ_DAT between to_date('"+DAT_IN+"','dd/mm/yyyy') and
(to_date('"+DAT_IN+"','dd/mm/yyyy')+1)"
           ;
    String[][] rr  = Gen.Qry2Array(CONN,qry);
    int len = Gen.getLEN(rr);

    outp += "<TABLE border=1>";

    outp += "<TR>";
    outp += "<TH> ";

    for(int x=0;x<12;x++)
    {
     outp += "<TH bgcolor='yellow' align=right>"+(9+x);
    }

    for(int y=0;y<6;y++)
    {
     String mm = Gen.lpad((10*y),2,"0");
     outp += "<TR>";
     outp += "<TD bgcolor='yellow' align=right>"+mm;

     for(int x=0;x<12;x++)
```

```
      {
       int hh = (9+x);
       String key = hh+":"+mm;
       outp += "<TD title='"+key+"'><INPUT name='"+PRM_IN+"' type=radio
value='"+key+"'>";
      }
      outp += "</TR>";
     }

     outp += "</TABLE>";
    }
    catch(Exception e) { throw new DBException(rn,e.toString()); }
    return(outp);
  }
}

package FrmBeans;

/*-- Y301_04bean ----------*/
/*-- Rendevouz presentation --*/
/*----------------------------*/
import libs.*;
import java.sql.*;
import org.w3c.dom.*;

public class Y301_04bean implements FormBean
{
 final private static String PP0 = "DUR";

 public Presenter work( Connection CONN
                , Document   TREE)
 throws Exception
 {
  String    rn = "Y301_04";
  Presenter outp = null;
  try
  {
   iSession ses = new iSession(TREE);
   int      lng = ses.getLNG();
   int      prc = ses.getPRC();                    if(prc!=301)        throw new
Exception("[Wrong prc="+prc+"]");
   String   eip = ses.getEIP();                    if(Gen.isEmpty(eip)) throw new
Exception("No EIP");
   String   des = Gen.getPrcDescr(CONN,prc,lng);
```

```
    int     hum = Gen.atoi(Gen.lookForNode(TREE,"HUM")); if(Gen.isEmpty(hum))
throw new Exception("No HUM");
    int     usr = Gen.atoi(Gen.lookForNode(TREE,"USR")); if(Gen.isEmpty(usr)) throw
new Exception("No USR");
    String  dat = Gen.lookForNode(TREE,"DAT");          if(Gen.isEmpty(dat)) throw
new Exception("No DAT");
    String  tim = Gen.lookForNode(TREE,"TIM");

    if(Gen.isEmpty(tim))
    {
     String e1= Gen.doProblem(rn,lng,des,"No selection");
     outp = new Presenter(e1,null,null);
    }
    else
    {
     HtmlForm frm  = new HtmlForm(1);
     frm.LNG       = lng;
     frm.EIP       = eip;
     frm.FRV[0]    = Gen.getUsrName(CONN,usr)+", "+dat+" "+tim;
     frm.FRV[4]    = des + " ("+Gen.get_HumanName(CONN,hum)+")";
     frm.FRV[14]   = rn;
     frm.PRM[0][0] = PP0;
     frm.PRM[0][1] = getPR0(CONN,PP0,usr,dat,tim,lng);
     frm.PRM[0][11] = Gen.get_LABEL(CONN,3018,lng);
     frm.PRM[0][12] = Gen.get_LABEL(CONN,3019,lng);

     outp = new Presenter(TREE,frm);
    }
   }
   catch(Exception e) { throw new DBException(CONN,rn,e.toString()); }
   return(outp);
  }

  public String getVars( String[][] WebVars
                       , Connection CONN
                       , Document  TREE)
  throws Exception
  {
   String rn = "Y301_03.getVars";
   try
   {
    String tmp = Gen.getParamValue(WebVars,PP0);
    Gen.insNODE(TREE,PP0,tmp);
   }
   catch(Exception e) { throw new DBException(CONN,rn,e.toString()); }
   return(null);
```

```
 }

 /*-- getPR0 --------------------*/
 /*-- Rerurns presentable Part --*/
 /*-----------------------------*/
 private static String getPR0( Connection CONN
                     , String    PRM_IN
                     , int       USR_IN
                     , String    DAT_IN
                     , String    TIM_IN
                     , int       LNG_IN)
 throws Exception
 {
  String rn   = "Y242_01.getPRS";
  String outp = "";
  try
  {
   String prs = "";
   prs += "<TR><TD><INPUT name="'+PRM_IN+'" type=radio value='10'
CHECKED>10 λεπτά</TR>";
   prs += "<TR><TD><INPUT name="'+PRM_IN+'" type=radio value='20'>20
λεπτά</TR>";
   prs += "<TR><TD><INPUT name="'+PRM_IN+'" type=radio value='30'>30
λεπτά</TR>";
   prs += "<TR><TD><INPUT name="'+PRM_IN+'" type=radio value='40'>40
λεπτά</TR>";
   prs += "<TR><TD><INPUT name="'+PRM_IN+'" type=radio value='50'>50
λεπτά</TR>";
   prs += "<TR><TD><INPUT name="'+PRM_IN+'" type=radio value='60'>60
λεπτά</TR>";

   outp += "<TABLE border=1>";
   outp += prs;
   outp += "</TABLE>";
  }
  catch(Exception e) { throw new DBException(rn,e.toString()); }
  return(outp);
 }
}
```

```
package FrmBeans;

/*-- Y301_FNbean ------------------*/
/*-- Loads a new Rendevouz to DB  --*/
/*---------------------------------*/
import libs.*;
import java.util.*;
import java.sql.Connection;
import org.w3c.dom.*;

public class Y301_FNbean implements FormBean
{
 public Presenter work( Connection CONN
                , Document   TREE)
 throws Exception
 {
  String   rn  = "Y301_FN";
  Presenter outp = null;
  try
  {
   iSession ses = new iSession(TREE);
   int     prc = ses.getPRC();     if(prc!=301) throw new Exception(rn+"Wrong_PRC
[prc="+prc+"]");
   String  eip = ses.getEIP();
   int     lng = ses.getLNG();
   int     uid = ses.getUID();
   String  des = Gen.getPrcDescr(CONN,prc,lng);

   int     usr = Gen.atoi(Gen.lookForNode(TREE,"USR")); if(Gen.isEmpty(usr)) throw
new Exception("No USR");
   String  dat = Gen.lookForNode(TREE,"DAT");          if(Gen.isEmpty(dat)) throw
new Exception("No DAT");
   String  tim = Gen.lookForNode(TREE,"TIM");          if(Gen.isEmpty(tim)) throw
new Exception("No TIM");
   int     dur = Gen.atoi(Gen.lookForNode(TREE,"DUR")); if(Gen.isEmpty(dur)) throw
new Exception("No DUR");
   int     hum = Gen.atoi(Gen.lookForNode(TREE,"HUM")); if(Gen.isEmpty(hum))
throw new Exception("No HUM");
   int     trt = Gen.atoi(Gen.lookForNode(TREE,"TRT"));

   transaction(CONN,usr,dat,tim,dur,hum,trt,uid); //=* Transaction
   ses.Lock(CONN);
   Gen.dbSQL(CONN,"commit",rn);                  //=* Transaction End

   HtmlForm frm = new HtmlForm(1);
```

```
    frm.LNG      = lng;
    frm.EIP      = eip;
    frm.FRV[1]   = "M1";
    frm.FRV[4]   = des;
    frm.FRV[14]  = rn;
    frm.PRM[0][0] = "_DUM";
    frm.PRM[0][1] = getTOP(CONN,usr,dat,tim,dur,hum)+Gen.LoopBack(CONN,uid);
    outp = new Presenter(TREE,frm);
   }
   catch(Exception e) { throw new DBException(CONN,rn,e.toString()); }
   return(outp);
  }

  public String getVars(String[][] WebVars,Connection CONN,Document TREE)
  throws Exception
  { return(null); }

  /*-- getTOP ---------------------*/
  /*---------------------------------*/
  private static String getTOP( Connection CONN
                      , int      USR_IN
                      , String   DAT_IN
                      , String   TIM_IN
                      , int      DUR_IN
                      , int      HUM_IN)
  throws Exception
  {
   String rn = "Y301_FN.getTOP";
   String outp = "";
   try
   {
    outp += "<TABLE>";
    outp += "<CAPTION>OK!</CAPTION>";
    outp +=
"<TR><TD>Πελάτης<TD>"+Gen.get_HumanName(CONN,HUM_IN)+"</TR>";
    outp += "<TR><TD>Χρήστης<TD>"+Gen.getUsrName(CONN,USR_IN)+"</TR>";
    outp += "<TR><TD>Ωρα Ραντεβου<TD>"+DAT_IN+" "+TIM_IN+"</TR>";
    outp += "<TR><TD>Διάρκεια Ραντεβου<TD>"+DUR_IN+" λεπτά</TR>";
    outp += "</TABLE>";
   }
   catch(Exception e) { throw new DBException(rn,e.toString()); }
   return(outp);
  }

  /*-- transaction ----------------------*/
  /*-----------------------------------*/
```

```
 private static void transaction( Connection CONN
                        , int      USR_IN
                        , String    DAT_IN
                        , String    TIM_IN
                        , int      DUR_IN
                        , int      HUM_IN
                        , int      TRT_IN
                        , int      UID_IN)
 throws Exception
 {
  String rn = "Y301_FN.transaction";
  try
  {
   String sql = "insert into
OBJ_RDZ(USR_ID,RDZ_DAT,RDZ_DUR,HUM_ID,TRT_ID,REG_USR)"
           + " values("+USR_IN+",to_date('"+DAT_IN+" "+TIM_IN+"','dd/mm/yyyy
hh24:mi'),"+DUR_IN+","+HUM_IN+","+TRT_IN+","+UID_IN+")"
           ;
    Gen.dbSQL(CONN,sql,rn);
  }
  catch(Exception e) { throw new DBException(rn,e.toString()); }
 }
}
```

ackage FrmBeans;

```
/*-- Z1bean ------------------------*/
/*-- Bean for Generic Presentation --*/
/*---------------------------------*/
import libs.*;
import java.sql.Connection;
import org.w3c.dom.Document;

public class Z1bean implements FormBean
{
 public Presenter work( Connection CONN
                , Document   TREE)
 throws Exception
 {
  if(TREE==null) throw new DBException(CONN,"Z1","No_TREE");
  HtmlForm frm = new HtmlForm(CONN,TREE,"Z1bean");
  return(new Presenter(TREE,frm));
 }

 public String getVars( String[][] WebVars
```

```
                , Connection CONN
                , Document   TREE)
  throws Exception
  {
   new WebVarLoader(WebVars,CONN,TREE);
   return("");
  }
}
```

Βιβλιογραφία

http://searchcio.techtarget.com/definition/business-process-management
http://www.w3schools.com/html
http://www.oracle.com
http://www.tutorialspoint.com/java
http://en.wikipedia.org/wiki/Session_(computer_science)
http://en.wikipedia.org/wiki/Business_process_management
http://www.youtube.com/
Εσωτερικός και Εξωτερικός Έλεγχος Ανώνυμων Εταιρειών εκδόσεις Σάκκουλας
Πληροφοριακά Συστήματα Επιχειρήσεων Ι εκδόσεις Σταμούλη