

- U

INDICE

1. INTRODUZIONE

2. ARCHITETTURA TARGET E STRATEGIA DI MIGRAZIONE

3. MIGRAZIONE DI UN PROGRAMMA SCRITTO IN ACUCOBOL-GT

- *CONFIGURAZIONE DI BASE*
- *FASE DI PRE-PROCESSING*
- *FASE DI RESTRUCTURING AND WRAPPING*
- *FASE DI GUI REENGINEERING*
- *RUN DELL'APPLICAZIONE MIGRATA*

4. APPENDICE

INTRODUZIONE

MELIS è un ambiente integrato per la migrazione di sistemi legacy (LIS) scritti in ACUCOBOL-GT verso sistemi web multi-tier è stato progettato e sviluppato come plug-in per la piattaforma ECLIPSE.

Prima di comprendere il supporto fornito allo sviluppatore dato dall'ambiente in fase di migrazione, è utile presentare le fasi del processo di migrazione che prevede di separare il layer relativo alle Graphical User Interface (GUI) dall'application layer e dal data layer in modo da wrappare il sistema legacy solo a livello di user interface.

ARCHITETTURA TARGET E STRATEGIA DI MIGRAZIONE

L'architettura target adottata è rappresentata nella Figura 1. E' stato sviluppato un *Wrapper* sottoforma di DLL, necessario alla comunicazione tra le interfacce migrate e l'application layer del LIS. In particolare il Wrapper funge da middleware in quanto permette di eseguire il sistema LIS ristrutturato e di gestire la sincronizzazione tra quest'ultimo e le interfacce reingegnerizzate. Sono due le interfacce di cui si compone il wrapper. La prima è fornita al LIS ristrutturato che è stato trasformato in un programma batch. La seconda è fornita al *GUIDeliverer*, componente che permette alle interfacce grafiche reingegnerizzate di accedere alle funzionalità del LIS. Le pagine dinamiche generate ed i relativi controlli costituiscono il componente *ReengineeredGUI*.

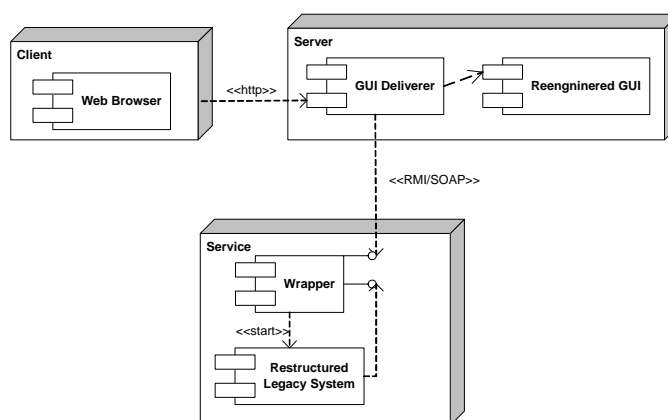


Figura 1 Architettura target

La strategia prevede le seguenti fasi: *Pre-processing*, di *Restructuring and Wrapping*, *GUI Reengineering*, *Integration* (Figura 2).

La prima fase fornisce dettagli sul tipo e sugli attributi di ogni oggetto grafico che compone la GUI del sistema legacy. In questa fase è possibile classificare le procedure associate ad ogni ENTRY-FIELD:

- procedure necessarie a validare il contenuto del campo e dunque migrabili lato client con un linguaggio di scripting quale JavaScript.
- procedure che accedono al DB ed operazioni di logica applicativa invocate dal *Wrapper*

Ogni SCREEN SECTION gestisce input ed output usando gli statement DISPLAY ed ACCEPT.

Nella fase di *Restructuring and Wrapping* ogni call relativa a questi due statement, viene rimpiazzata da istruzioni che incapsulano la comunicazione con il *Wrapper component*. Sempre in questa fase verrà deciso quale delle procedure AFTER/BEFORE saranno migrate lato client con un linguaggio di scripting.

La fase di *GUI Reengineering* permette di creare le pagine web dinamiche per ogni sottosistema migrato.

Il passo finale dell'approccio prevede l'integrazione delle nuove GUI e del sistema legacy ristrutturato in modo da produrre il sistema target.

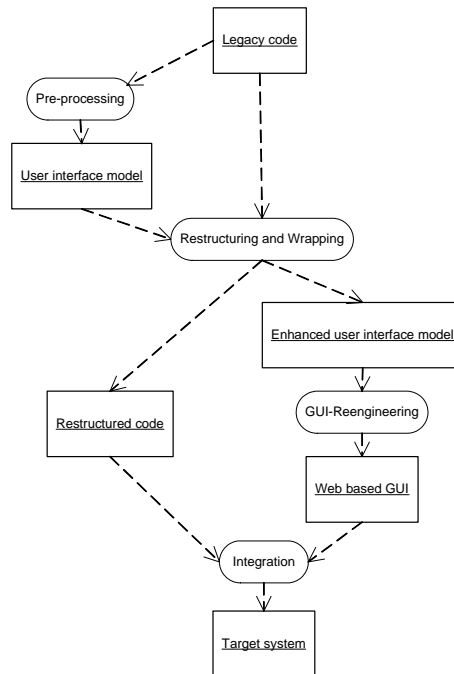


Figura 2 Strategia di migrazione

MIGRAZIONE DI UN PROGRAMMA SCRITTO IN ACUCOBOL-GT

CONFIGURAZIONE DI BASE
(già effettuata sulla macchina di test)

L'ambiente di migrazione proposto implementa ogni passo della strategia appena descritta permettendo allo sviluppatore di migrare il sistema legacy verso l'architettura target vista in precedenza.

Prima di procedere con la migrazione, è necessario creare un nuovo progetto come proposto *Figura 3*

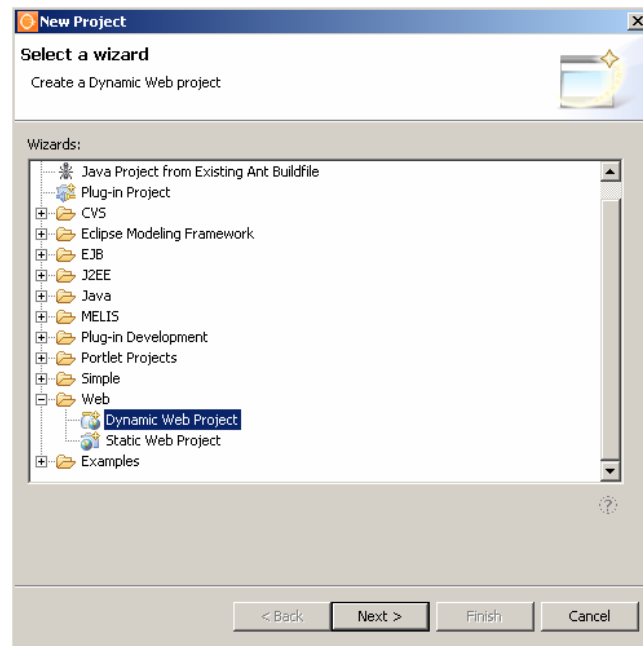


Figura 3 Creazione di un progetto vuoto

Facendo click col tasto destro del mouse sul nome dato al progetto, creare una nuova cartella in cui inserire i sorgenti del sistema cobol da migrare (*Figura 4*).

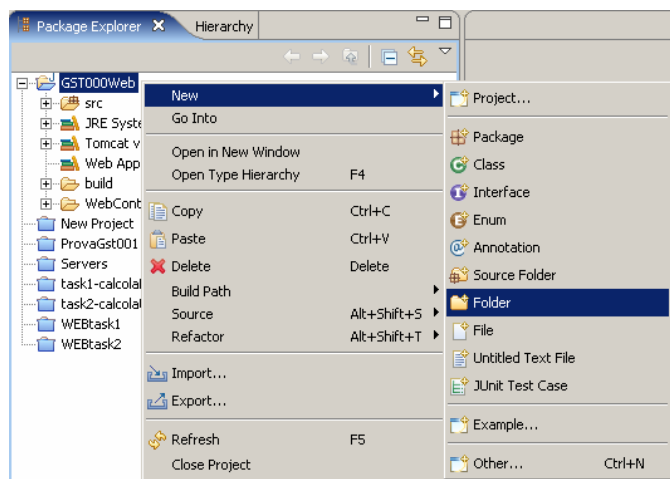


Figura 4 Creazione cartella contenente i sorgenti del LIS

Una volta copiati tutti i file contenuti nella cartella SORGENTI-COBOL presente sul desktop nella cartella appena creata, è necessario settare i parametri per la corretta configurazione dell'ambiente di compilazione e run-time di ACUCOBOL-GT. Selezionando la cartella contenente i sorgenti cobol bisogna accedere alla voce del menù *Build Path > Configure Build Path* (*Figura 5*).

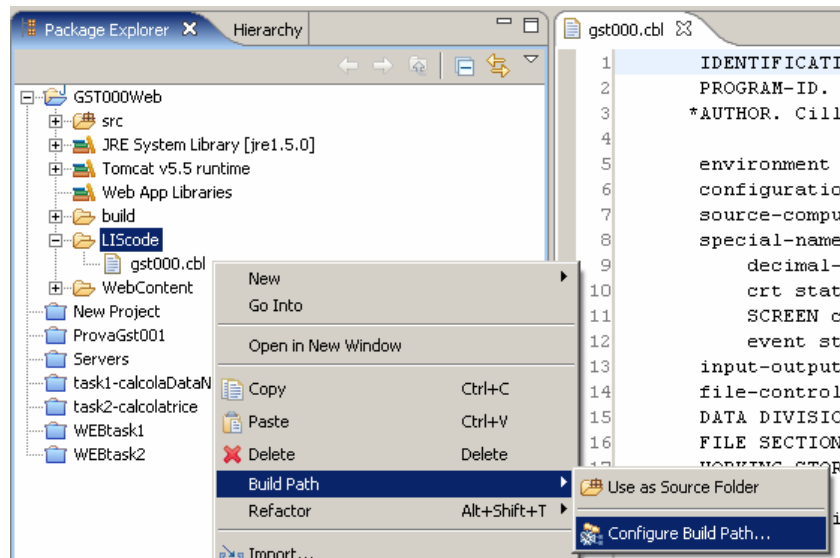


Figura 5 Configurazione ACUCOBOL-GT environment

Nella successiva finestra selezionando la voce *Environment* e cliccando su *Restore Defaults*, in maniera automatica saranno settati tutti i parametri necessari alla compilazione, al debug ed al run di un sorgente ACUCOBOL-GT (Figura 6). Sarà sempre possibile modificare i parametri con le apposite opzioni.

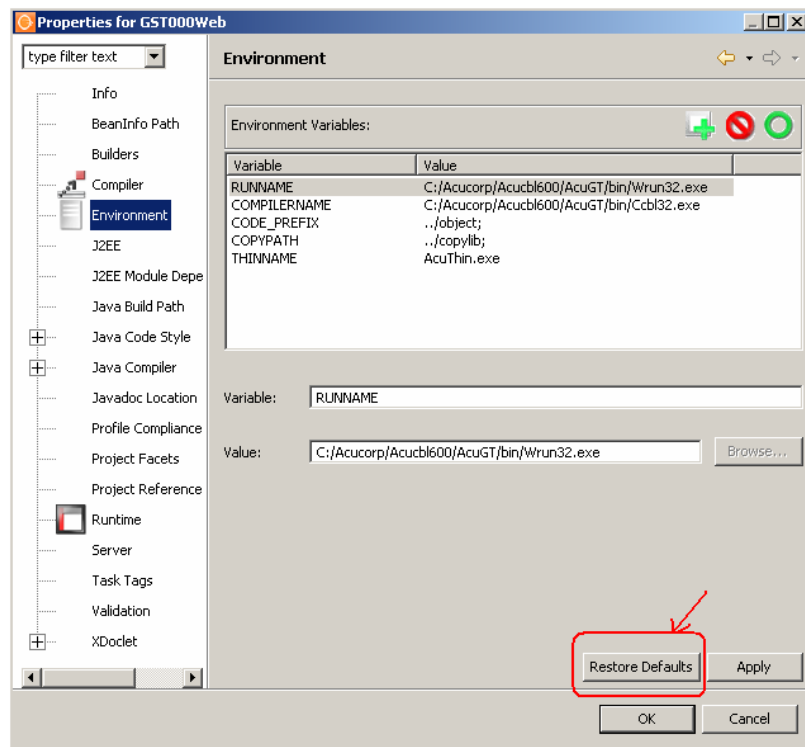


Figura 6 Configurazione parametri ACUCOBOL-GT

Per ogni file ACUCOBOL-GT sarà poi possibile effettuare una delle operazioni messe a disposizione dall'ambiente (Figura 7)

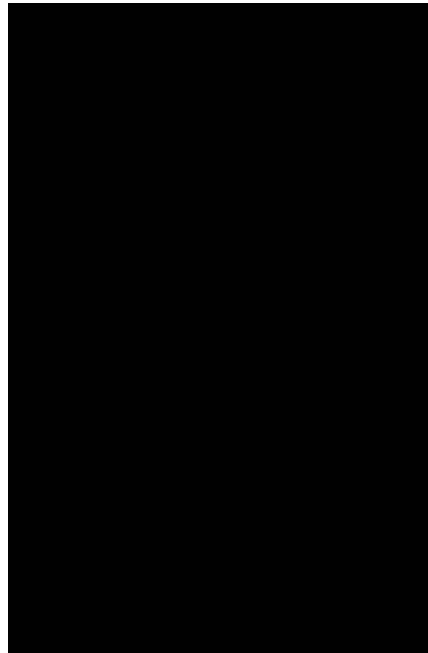


Figura 7 Opzioni per un file ACUCOBOL-GT

FASE DI PRE-PROCESSING

Nella fase di *Pre-processing* MELIS permette di generare in maniera automatica la rappresentazione della struttura di ogni SCREEN SECTION sottoforma di un file XML. Il file include inoltre tutte le azioni associate a ciascun campo contenuto nella screen e tutti i dettagli corrispondenti agli oggetti grafici. In Figura 8 è mostrato come accedere alla funzionalità messa a disposizione dall'ambiente.

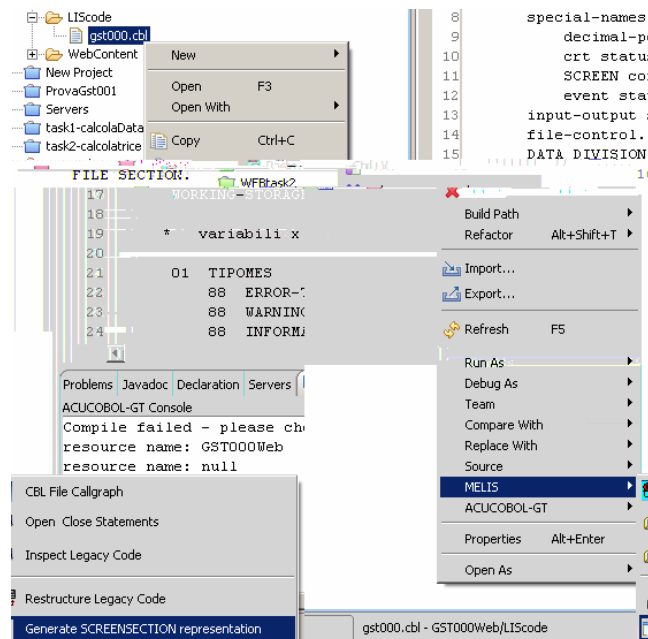


Figura 8 MELIS: fase di Pre-processing

La Figura 9 mostra l'operazione associata ad un ENTRY-FIELD¹. Questa operazione controlla che il campo contenga il valore 'S' o 'N' quindi facilmente

¹ Il codice può essere evidenziato di colore ROSSO (call ad una routine esterna), di colore GIALLO (display o refresh) e di colore GRIGIO (inizializzazione oggetti grafici). Se il codice non presenta sezioni evidenziate in rosso si può decidere se migrarlo lato client, altrimenti, è necessario controllare che la call non faccia parte della logica applicativa del LIS.

esportabile lato client sottoforma di un controllo javascript. Per inserire tale controllo, basta abilitare la scrittura nell'area di testo *JavaScript Code* e scrivere il codice necessario a riprodurre il controllo originale. L'utente può anche selezionare ed inserire automaticamente degli script di default suggeriti da MELIS nella finestra "*Suggestion Code*".

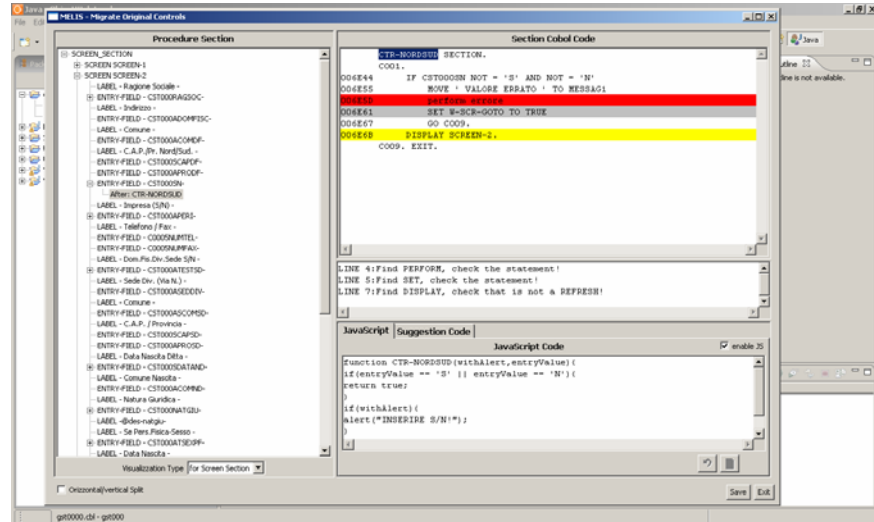


Figura 9 Migrazione controlli lato client

FASE DI *RESTRUCTURING AND WRAPPING*

Dopo aver deciso quali controlli migrare lato client e quali lato server, è necessario ristrutturare il sistema legacy in modo da wrappe il layer di interfaccia. Il file XML generato nella fase precedente sarà fornito in input a tale fase.

Ogni SCREEN SECTION gestisce input ed output usando gli statement DISPLAY ed ACCEPT. In questa fase, ogni call relativa a questi due statement, viene rimpiazzata da istruzioni che incapsulano la comunicazione con il *Wrapper component*.

In particolare la finestra riportata in Figura 10, permette allo sviluppatore di accedere alla funzionalità di ristrutturazione del codice legacy Figura 11.

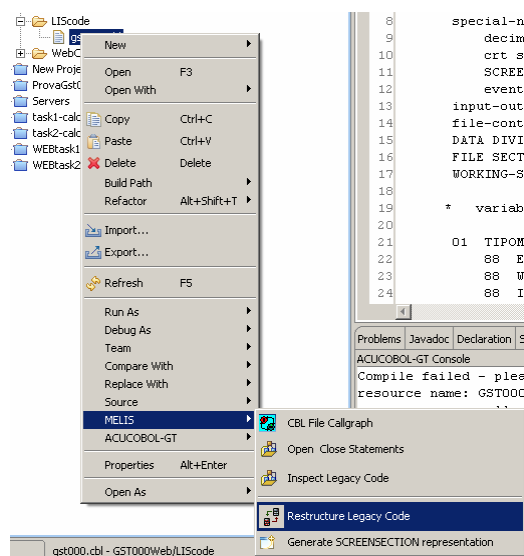


Figura 10 MELIS: fase di Restructuring and Wrapping

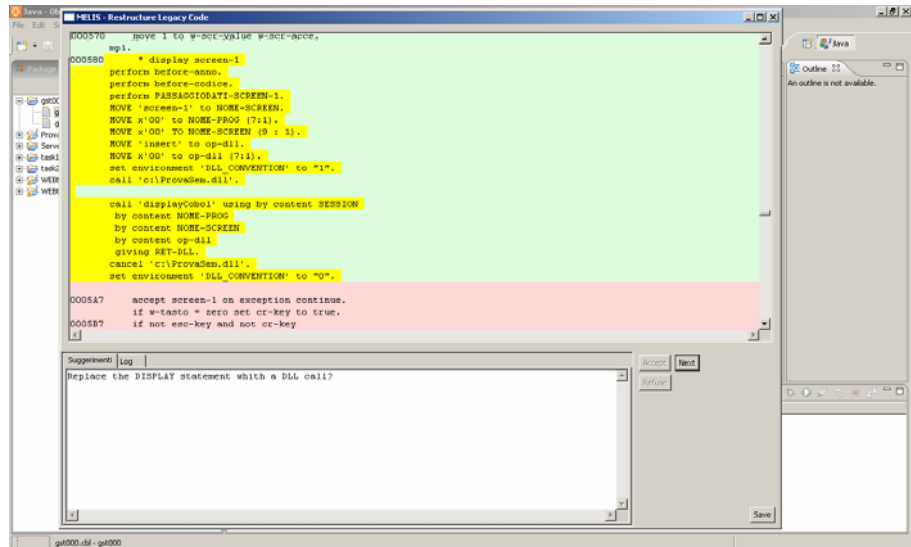


Figura 11 Ristrutturazione del codice LEGACY

In questa finestra MELIS evidenzia in giallo lo statement DISPLAY SCREEN-1 che sarà sostituito con una chiamata al Wrapper. Anche il codice inserito in modo automatico, sarà evidenziato in giallo. Tale procedura potrà essere eseguita per ogni coppia di DISPLAY/ACCEPT presenti all'interno del sorgente cobol.

FASE DI GUI REENGINEERING

Il file XML ottenuto dalla fase di Pre-processing è necessario per la generazione automatica delle interfacce grafiche reingegnerizzate. La figura Figura 12 mostra come accedere alla funzionalità messa a disposizione da MELIS per questa fase del processo. Successivamente si aprirà la finestra mostrata in Figura 13 che permetterà allo sviluppatore di creare le interfacce grafiche reingegnerizzate. L'ambiente offre opzioni che permettono all'utente di stabilire il nome del path dell'applicazione web (URL) e quali contenuti generare.

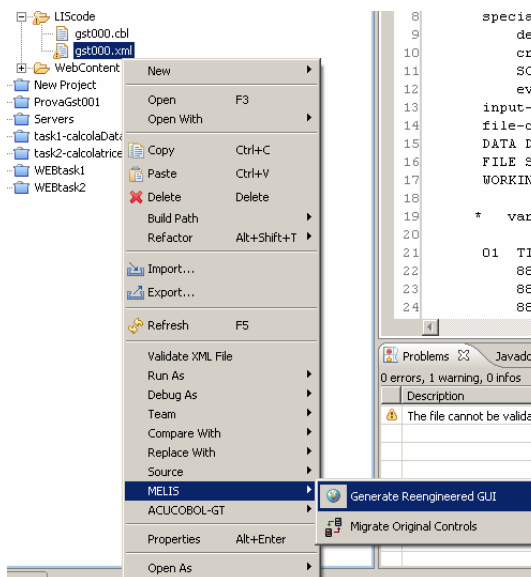


Figura 12 MELIS: fase di GUI Reengineering

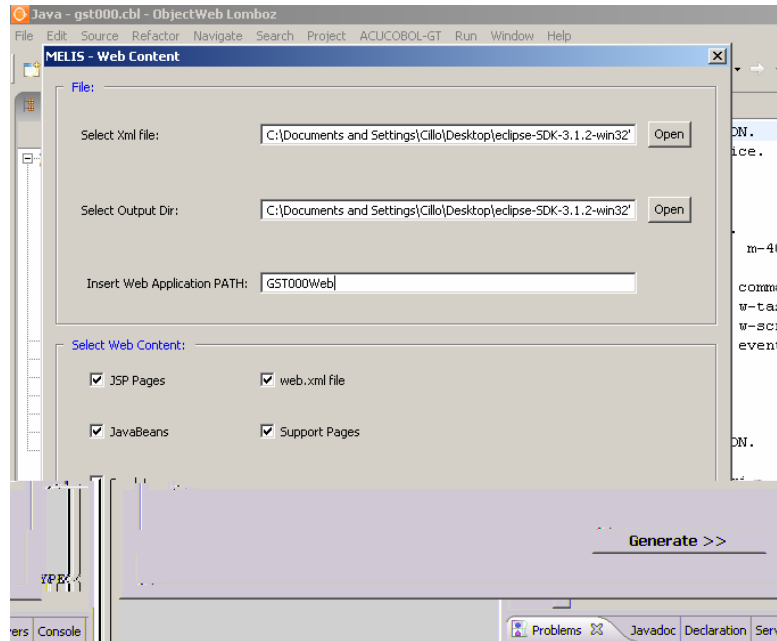


Figura 13 Generazione interfaccia grafiche Web Based

A questo punto MELIS avrà generato una cartella all'interno del progetto dal nome **MELIS-WebGUI** (Figura 14)

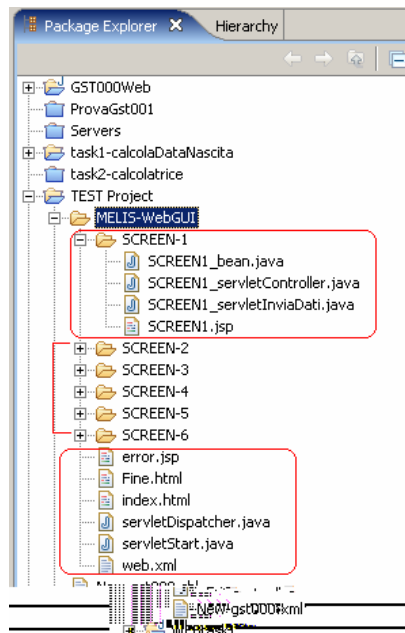


Figura 14 Output della fase di GUI Reengineering

In questa cartella sarà creata per ogni SCREEN-SECTION una cartella contenente una pagina JSP, un javabean e le due servlet, saranno inoltre create le pagine HTML di supporto, la *servletStart* e la *servletDispatcher* ed infine il file *web.xml*. A questo punto è necessario effettuare le seguenti operazioni:

1. copiare i javabean nel package *bean* della cartella *src* del progetto
2. copiare tutte le servlet nel (*default package*) della cartella *src* del progetto
3. copiare tutte la pagine JSP/HTML nella cartella *WebContent* del progetto
4. copiare il file *web.xml* nella cartella *WebContent/WEB-INF* del progetto

La **Figura 15** riporta un esempio delle operazioni compiute in precedenza:

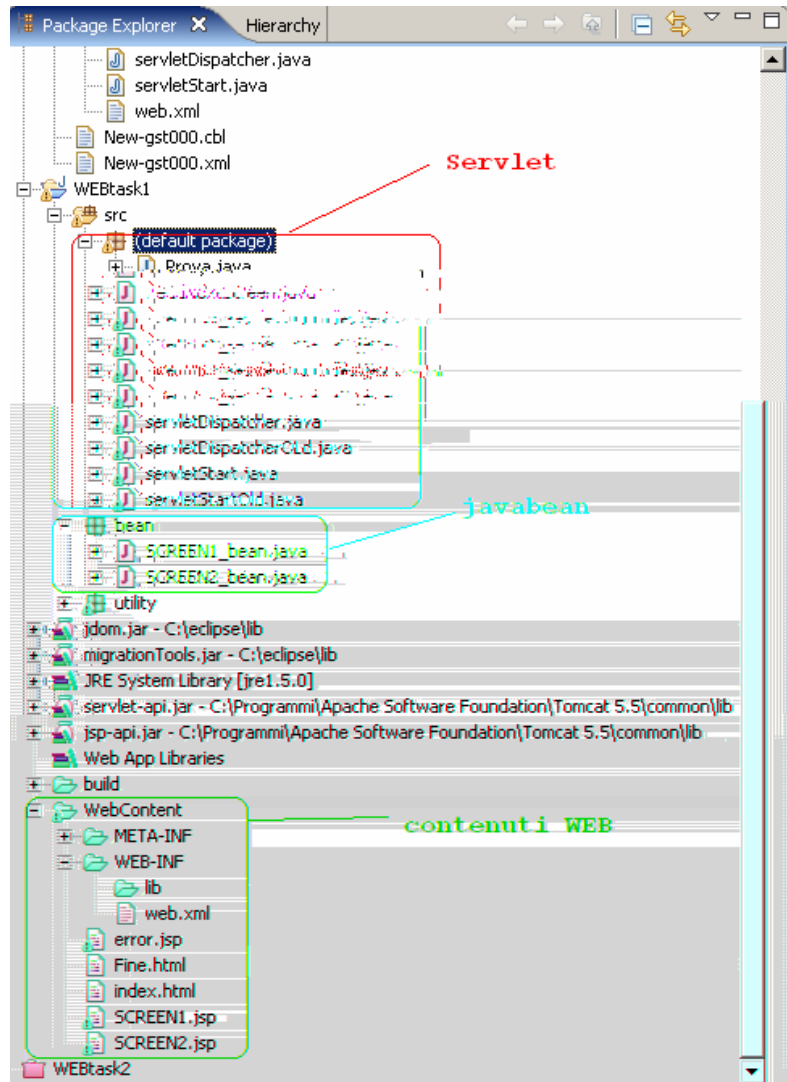


Figura 15 Deploy applicazione web

RUN DELL'APPLICAZIONE
MIGRATA

Una volta completate tutte le fasi del processo, è possibile eseguire l'applicazione migrata all'interno dell'ambiente così come mostrato in Figura 16.

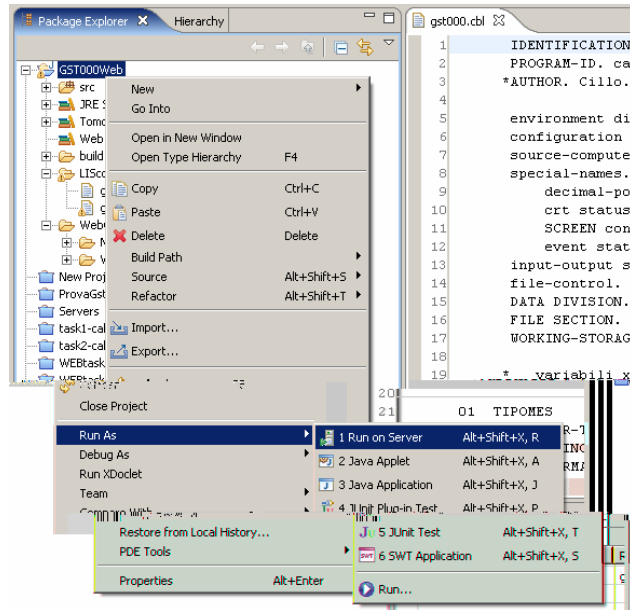


Figura 16 Run del sistema migrato

La Figura 17 riporta il sistema target in esecuzione.

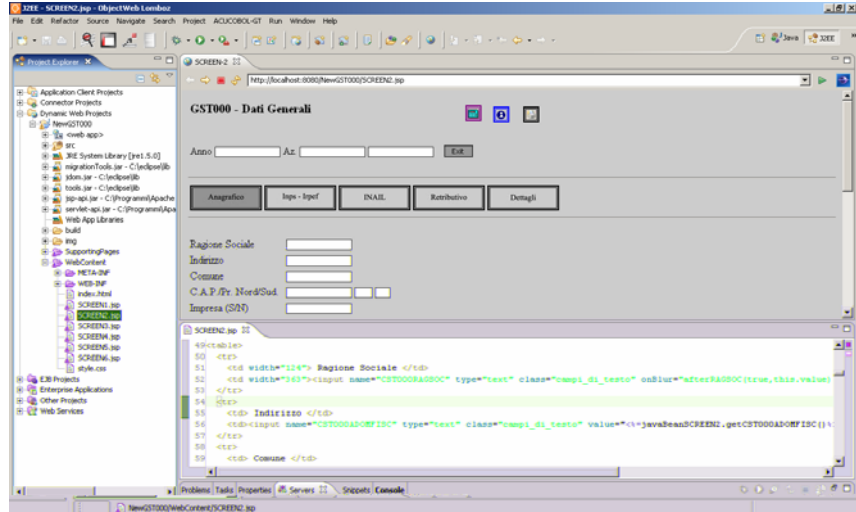


Figura 17 Sistema target in esecuzione

APPENDICE

- 1) **Usa corretto del Wrapper Component.** Come già detto nella fase di *Restructuring and Wrapping*, vanno sostituite tutte le coppie di chiamate ACCEPT/DISPLAY con una call al Wrapper e vanno individuate le AFTER/BEFORE da migrare lato client. E' inoltre necessario **inserire a mano** due funzioni addette alla corretta sincronizzazione tra i due sistemi. Per semplicità saranno chiamate WAIT ed EXIT. La WAIT permette di bloccare l'esecuzione del sistema cobol solo dopo la sua inizializzazione: così facendo, si resta in attesa mentre il sistema web sta eseguendo. La EXIT invece, serve a terminare l'esecuzione del sistema COBOL. Queste due procedure vanno inserite una volta sola. La prima va inserita, di norma, subito dopo la MAIN PROGRAM SECTION che da avvio al sistema LIS. La seconda invece, va inserita nel punto in cui si vuol far terminare l'esecuzione del LIS. Si riporta di seguito il codice per le due procedure:

```

WAIT
    set environment 'DLL_CONVENTION' to "1".
    call 'c:\ProvaSem.dll'.
    call 'Cblwait' using by content session.
    cancel 'c:\ProvaSem.dll'.
    set environment 'DLL_CONVENTION' to "0".

```

```

EXIT
    set environment 'DLL_CONVENTION' to "1".
    call 'c:\ProvaSem.dll'.
    call 'Cblexit' using by content session.
    cancel 'c:\ProvaSem.dll'.
    set environment 'DLL_CONVENTION' to "0".

```

- 2) **Passaggio corretto di parametri numerici.** Tutti i dati scambiati tra il LIS ristrutturato ed il sistema Web attraverso il *Wrapper* sono delle STRINGHE. A tal proposito è necessario **inserire a mano** l'istruzione cobol CONVERT nel momento in cui il sistema LIS si aspetta di ricevere dei dati di tipo NUMERICO dal sistema WEB. Ad ogni modo MELIS inserisce nel codice LIS ristrutturato un commento in cui si ricorda allo sviluppatore di effettuare tale controllo.

```

    move 'N' TO ret-x.
    move '1' to name-anno (5:1).
    call 'c:\leggInput.wob' using
        nome-file name-NUMERO
        value-NUMERO ret-x.
    if ret-x = 'S'
        move value-NUMERO to NUMERO CONVERT
    end-if.

```

- 3) **Controllare i SALTII CONDIZIONATI.** Quasi tutti i sorgenti COBOL sono ricchi di *GO TO*, il che rende difficile gestire in modo corretto il flusso di controllo del sistema. E' necessario dunque porre molta attenzione a questi salti condizionati e commentare i GO TO quando serve:

```

    if w-tasto not = 2 and 27 go to mp0.
    if w-tasto=27 go to mp9.

```



- 4) **COMMENTARE call non necessarie.** E' stato più volte detto che la strategia di migrazione descritta, permette di migrare il sistema LIS a livello di interfaccia utente. Per questo motivo, è possibile commentare tutte quelle CALL relative alle interfacce grafiche: gestione degli eventi sui bottoni, inizializzazione del nome delle finestre etc.
- 5) **PROBLEMI con la SESSIONE.** A causa di alcuni problemi con la sessione, che funge da ID per istanziare correttamente il Wrapper, è necessario **inserire a mano nel codice COBOL**, subito dopo la MAIN PROGRAM SECTION, il seguente codice:

```
move 'ABCDEFGHILMNOPQRSTUVWXYZ1234567890F' to SESSION.  
MOVE x'00' to SESSION(33:1).
```

Lo stesso problema si presenta **lato JAVA** per cui nelle *servletStart* e *servletDispatcher* va inserita la seguente stringa:

```
static final public String sessionID =  
"ABCDEFGHILMNOPQRSTUVWXYZ1234567890F";
```