Tecnologie di Sviluppo per il Web

Applicazioni Web J2EE
Framework per il Modello 2
it.unibas.pinco

versione 3.2

G. Mecca – mecca@unibas.it – Università della Basilicata

Sommario

- Framework per Applicazioni Web
- Un Framework Didattico per J2EE
  - ⇨Il Manuale di pinco
- Diagrammi delle Attività
- Linee Guida

Framework per Applicazioni Web

- Framework per applicazioni Web
  - ⇨infrastruttura di classi riutilizzabili in applicazioni Web diverse
  - ⇨tipicamente fornisce un controllore frontale
  - ⇨una sintassi per il file di configurazione
  - ⇨una serie di interfacce per le azioni
  - ⇨altre funzionalità di supporto allo sviluppo (es: sistema di logging)

Framework per Applicazioni Web

- Disponendo del framework
  - ⇨l'utente deve limitarsi a sviluppare i componenti mancanti dell'applicazione
- Ovvero
  - ⇨sviluppare e verificare il modello
  - ⇨scrivere il file di configurazione
  - ⇨sviluppare le azioni (classi Java)
  - ⇨sviluppare gli schermi (pagine jsp)

Framework per Applicazioni Web

- Esempi di framework per J2EE
  - Jakarta Struts (jakarta.apache.org) – leader di mercato
  - Java Server Faces
  - Spring MVC
- Tutti estremamente complessi
  - numerose classi, vari MB di codice

Un Framework Didattico per J2EE

- it.unibas.pinco
  - il primo framework didattico per lo sviluppo di applicazioni Web
- Obiettivo
  - consentire di sperimentare praticamente lo sviluppo Web basato sul Modello 2
  - accorciando il ciclo di apprendimento
  - infrastruttura estremamente semplificata

Un Framework Didattico per J2EE

- Filosofia
  - ⇨“fornire solo quello che è essenziale”
  - ⇨poche classi, poco codice
  - ⇨in modo che sia analizzabile da parte degli studenti e ne siano chiari i meccanismi di funzionamento
- Obiettivo
  - ⇨allenare gli studenti allo sviluppo con altri framework industriali

Un Framework Didattico per J2EE

- Cosa offre pinco
  - ⇨un controllore frontale
  - ⇨la gestione della mappa delle azioni
  - ⇨le interfacce per le azioni
  - ⇨alcune azioni predefinite
  - ⇨un sistema di logging
  - ⇨un supporto alla convalida
  - ⇨una libreria di tag

Un Framework Didattico per J2EE

- Sono a carico dello sviluppatore
  - ⇨la gestione delle form e la convalida dei dati
  - ⇨la gestione delle autenticazioni e delle autorizzazioni
- Esempi di applicazioni con pinco
  - ⇨indovinaM2
  - ⇨aciM2

>> indovinaM2pinco
>> aciM2pinco

Il Manuale di Pinco

- Controllore frontale
  - ⇨classe it.unibas.pinco.framework.Controllo
  - ⇨lavora utilizzando la mappa delle azioni WEB-INF/pinco-config.xml
  - ⇨intercetta tutte le richieste rivolte a URI della forma *.cmd
  - ⇨utilizza la mappa per eseguire l'azione corrispondente e inoltrare la richiesta alla pagina Jsp successiva

Il Manuale di Pinco

- Redirezione degli URI in web.xml
  - è necessario configurare il controllore in modo da intercettare tutte le richieste HTTP

```
<servlet>
    <servlet-name>Controllo</servlet-name>
    <servlet-class>it.unibas.pinco.framework.Controllo</servlet-class>
</servlet>
<servlet-mapping>
   <servlet-name>Controllo</servlet-name>
   <url-pattern>*.cmd</url-pattern>
</servlet-mapping>
```


Il Manuale di Pinco

- Mappa delle azioni: pinco-config.xml
  - ⇨file xml valido rispetto al DTD pinco.dtd
  - ⇨collezione di elementi “descrittoreAzione”
- In ogni descrittore
  - ⇨un comando
  - ⇨una classe di azione
  - ⇨uno o più esiti con i relativi schermi successivi

Il Manuale di Pinco

- Comando
  - ⇨URI della forma <nome>.cmd
- Azione
  - ⇨classe Java dell'applicazione che implementa l'interfaccia Azione
- Esito
  - ⇨stringa associata all'URI di una pagina JSP che produce lo schermo successivo

Il Manuale di Pinco

- Esempio

```
<descrittoreAzione comando="gestisciTentativo.cmd“
                   azione="it.unibas.indovina.azioni.AzioneTentativo">
    <schermoSuccessivo esito="continua">
    /schermoTentativo.jsp
    </schermoSuccessivo>
    <schermoSuccessivo esito="indovinato">
    /schermoIndovinato.jsp
 </schermoSuccessivo>
    <schermoSuccessivo esito="errore">
   /schermoErrore.jsp
    </schermoSuccessivo>
</descrittoreAzione>
```

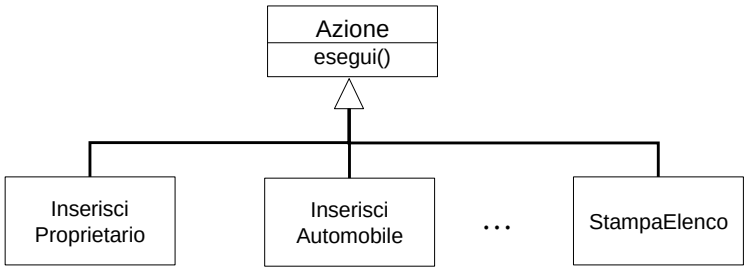
Il Manuale di Pinco

- Comunicazione tra i componenti
  - ⇨avviene attraverso l'oggetto request
  - ⇨il controllore passa la richiesta alle azioni
  - ⇨le azioni prelevano i parametri della query string dalla richiesta
  - ⇨le azioni restituiscono l'esito (una stringa)
  - ⇨le azioni salvano i riferimenti ai bean nella mappa della richiesta o della sessione o della applicazione

Il Manuale di Pinco

- L'interfaccia it.unibas.pinco.azioni.Azione

```
package it.unibas.pinco.azioni;
public interface Azione {
    public String esegui(javax.servlet.http.HttpServletRequest request);
}
```



Il Manuale di Pinco

- Attenzione
  - ⇨le azioni NON sono servlet nè pagine JSP
  - ⇨nel metodo esegui NON sono disponibili i riferimenti predefiniti delle pagine JSP
  - ⇨nè è possibile utilizzare i metodi della classe HttpServlet
  - ⇨l'unico riferimento disponibile è alla richiesta http perchè viene passato come argomento dal controllore frontale

Il Manuale di Pinco

- Come ottenere i riferimenti
  - ⇨request: argomento
  - ⇨session: request.getSession()
  - ⇨application: session.getServletContext() (oppure lavorando con la classe it.unibas.pinco.framework.Applicazione)
  - ⇨response: inutile perchè le azioni NON lavorano con la risposta HTTP

Il Manuale di Pinco

- Due azioni particolari
  - ⇨eventuale azione da eseguire all'inizializzazione del controllore (init())
  - ⇨eventuale azione da eseguire alla distruzione del controllore (destroy())
  - ⇨entrambe opzionali
- Interfacce
  - ⇨it.unibas.pinco.azioni.AzioneIniziale
  - ⇨it.unibas.pinco.azioni.AzioneFinale

Il Manuale di Pinco

- AzioneIniziale e AzioneFinale

```
package it.unibas.pinco.azioni;
public interface AzioneIniziale {
    public void esegui(ServletContext sc);
}
```

```
package it.unibas.pinco.azioni;
public interface AzioneFinale {
    public void esegui(ServletContext sc);
}
```

Il Manuale di Pinco

- Nel file di configurazione

```
<pinco-config>
    <azioneIniziale>
        it.unibas.prova.AzioneInizialeProva
    </azioneIniziale>
    <azioneFinale>
        it.unibas.prova.AzioneFinaleProva
    </azioneFinale>
    <azioni>
        // descrittori
    <azioni>
</pinco-config>
```


Il Manuale di Pinco

- Attenzione
  - ⇨gli unici URI dell'applicazione a cui è consentito fare richieste sono gli uri dei comandi (*.cmd)
  - ⇨non è possibile accedere direttamente alle pagine JSP, tranne che a index.jsp
- Lo strumento
  - ⇨il filtro degli accessi di pinco

Il Manuale di Pinco

- Filtro per gli accessi
  - ⇨protegge le pagine JSP (tranne index.jsp) per impedire l'accesso diretto
- La ragione
  - ⇨in questo modo NON è necessario effettuare verifiche sulla richiesta HTTP negli schermi
  - ⇨è sufficiente effettuare i controlli all'interno della azioni

>> web.xml

Il Manuale di Pinco

- Azioni predefinite
  - ⇨it.unibas.pinco.azioni.AzioneInoltro
  - ⇨restituisce sempre "forward"
  - ⇨utilizzabile in tutti i casi in cui bisogna navigare verso uno schermo direttamente
- Esempio

```
<descrittoreAzione comando="help.cmd"
                   azione="it.unibas.pinco.azioni.AzioneInoltro">
  <schermoSuccessivo esito="forward">help.jsp</schermoSuccessivo>
</descrittoreAzione>
```

Il Manuale di Pinco

- Azioni predefinite
  - ⇨it.unibas.pinco.azioni.AzioneInoltroSessione
  - ⇨restituisce “forward” se c’è una sessione in piedi
  - ⇨altrimenti restituisce “errore”
  - ⇨utilizzabile per l’inoltro diretto all’interno di una sessione già stabilita

Il Manuale di Pinco

- La classe Applicazione
  - ⇨incapsula il ServletContext (fornisce accesso alla mappa dell’applicazione)
  - ⇨fornisce accesso al sistema di logging
- Accesso al ServletContext
  - ⇨getAttribute(), setAttribute(), removeAttribute()
  - ⇨agiscono sulla mappa dell’applicazione

Il Manuale di Pinco

- Il logger di pinco
  - ⇨oggetto di tipo it.unibas.pinco.utilita.Logger inizializzato dal framework
  - ⇨ottenibile con Logger getPincoLogger() di Applicazione
  - ⇨può effettuare logging a livelli diversi: FINER, FINE, INFO, SEVERE
  - ⇨può effettuare logging su file o sullo standard output

Il Manuale di Pinco

- Nel caso di logging su standard output
  - ⇨tipicamente lo standard output del contenitore è rediretto ad un file
  - ⇨es: Tomcat: stdoutXXX.txt
- Configurazione del logger
  - ⇨attraverso il file /risorse/logger.properties
  - ⇨deve essere accessibile attraverso il classpath (es: in /WEB-INF/classes/risorse)

Il Manuale di Pinco

>> javadoc di Applicazione

- Nota
  - ⇨la classe Applicazione fornisce una serie di metodi statici di convenienza per effettuare il logging: logFiner() logFine(), logInfo(), logSevere()
  - ⇨inoltre, nell'applicazione sono disponibili due sistemi diversi di logging: quello predefinito del contenitore (metodo log() di ServletContext) e quello fornito da pinco

Il Manuale di Pinco

>> javadoc di Convalidatore

- Convalida dei dati
  - ⇨il framework fornisce supporto parziale alla convalida attraverso la classe it.unibas.pinco.utilita.Convalidatore
  - ⇨che fornisce metodi da utilizzare nei formBean per effettuare alcune convalide diffuse

Il Manuale di Pinco

- La libreria di tag
  - ⇨alcuni tag di utilizzo frequente
  - ⇨<pinco:noCache />
  - ⇨<pinco:errori />
  - ⇨<pinco:logoPinco />
- Per l'utilizzo della libreria

```
<%@ taglib uri="http://www.db.unibas.it/users/mecca/diogene/pinco"
            prefix="pinco" %>
```


Il Manuale di Pinco

>> indovinam2pinco
>> acim2pinco

- A questo punto
  - ⇨è utile discutere il codice delle applicazioni
- Aspetti interessanti
  - ⇨file di configurazione
  - ⇨deployment descriptor
  - ⇨sequenza schermo-comando-azione-esito-schermo
  - ⇨convalida dei dati
  - ⇨utilizzo delle azioni predefinite

Diagrammi delle Attività

- La metodologia di sviluppo
  - ⇨resta sostanzialmente la stessa
  - ⇨analisi dei requisiti
  - ⇨modello concettuale
  - ⇨sviluppo e test dei componenti di modello e persistenza
  - ⇨sviluppo e test dei componenti di interfaccia e controllo

Diagrammi delle Attività

- Il passo che cambia
  - ⇨è relativo allo sviluppo dei componenti di interfaccia e controllo
- Ovvero
  - ⇨selezione degli schermi
  - ⇨selezione delle azioni
  - ⇨costituzione della mappa comandi-azioni-esiti-schermi

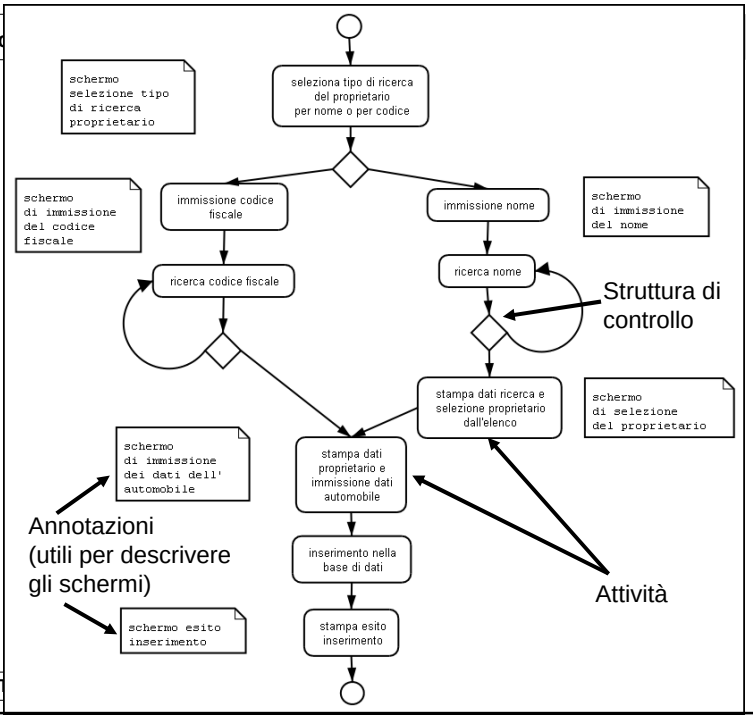
Diagrammi delle Attività

- Uno strumento molto utile
  - ⇨i diagrammi delle attività di UML
- Diagramma delle attività
  - ⇨diagramma dinamico di UML
  - ⇨che descrive l'evoluzione di schermi e azioni applicative all'interno di un caso d'uso
  - ⇨mettendo in evidenza il flusso di esecuzione e i diversi scenari

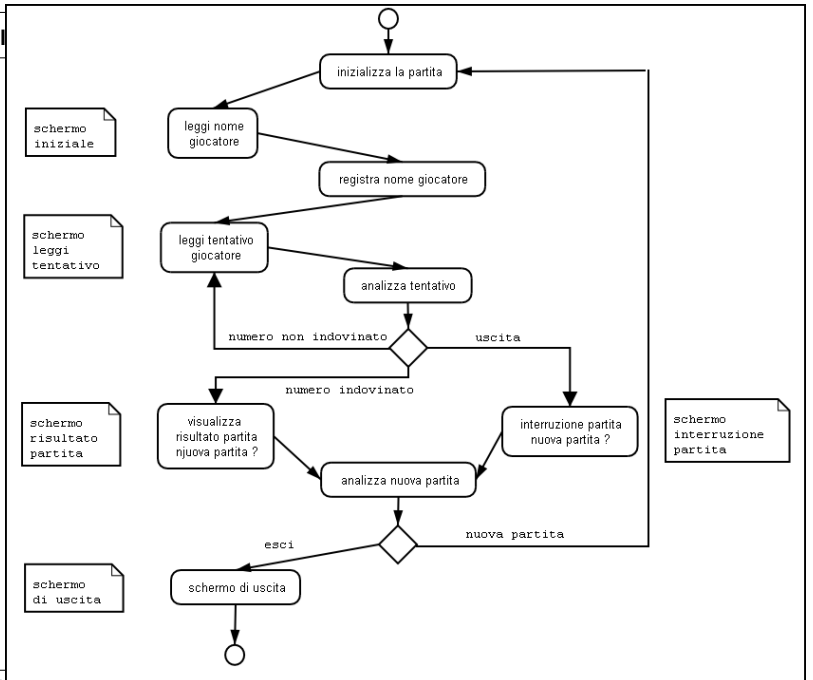
Diagrammi delle Attività

- Filosofia
  - ⇨è simile ad un diagramma di flusso tradizionale, ma il livello di granularità è molto meno fine
  - ⇨è un grafo in cui ciascun nodo corrisponde ad uno o più passi di un caso d'uso, e non ad una semplice istruzione
  - ⇨spesso le attività sono disposte in modo da distinguere attività che producono schermi da attività "interne" al sistema

Esempio: Inserisci Automobile



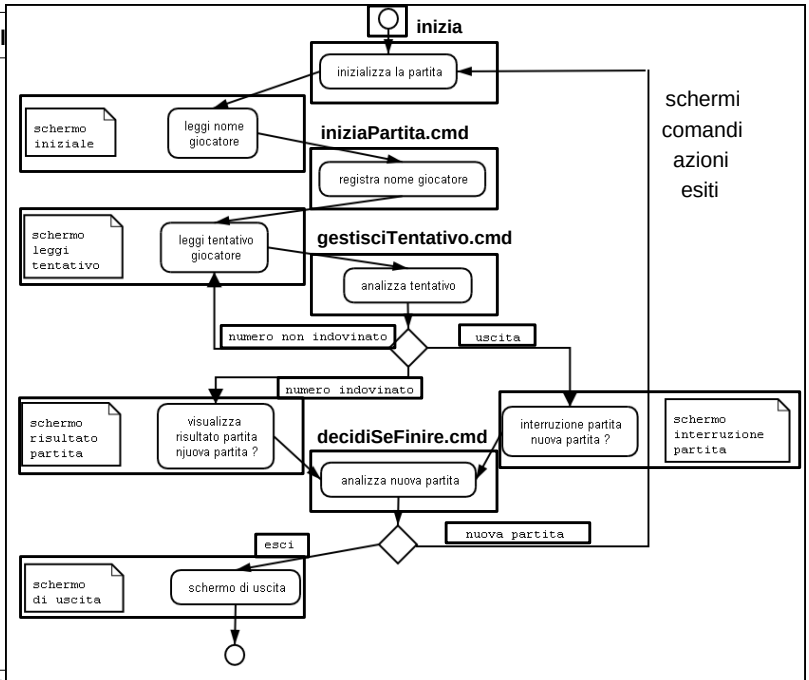
Esempio: Indovina il Numero



Diagrammi delle Attività

- Una caratteristica importante
  - ⇨a causa del meccanismo di produzione degli schermi (per cui gli schermi vengono rigenerati ex novo ogni volta) la mappa delle azioni discende immediatamente dal diagramma delle attività
  - ⇨è necessario individuare schermi e azioni
  - ⇨etichettare opportunamente i comandi (richieste provenienti dagli schermi)
  - ⇨e gli esiti (ovvero le diramazioni dalle strutture di controllo)

Esempio: Indovina il Numero



Diagrammi delle Attività

- Tipicamente
  - ⇨le azioni hanno esiti “standard” che si riferiscono ad errori di convalida o ad errori nella verifica della risposta
  - ⇨questi esiti tipicamente NON vengono descritti nel diagramma delle attività per evitare di appesantirlo
  - ⇨e vengono considerati impliciti

Diagrammi delle Attività

- Nota
  - ⇨per la descrizione degli schermi è opportuno utilizzare anche ulteriori strumenti
  - ⇨per esempio bozzetti che descrivano la disposizione grafica degli elementi nella pagina, le form e i controlli relativi
  - ⇨in questa fase bisogna rispettare le linee guida precedentemente discusse su accessibilità e usabilità

Linee Guida

- Nello sviluppo con il framework
  - ⇨è opportuno tenere presente alcune linee guida
- Linea guida n.1
  - ⇨lo sviluppo basato sul framework non è adatto ad applicazioni di piccola dimensione
  - ⇨in quel caso la maggiore complessità dell'infrastruttura non è compensata

Linee Guida

- Linea guida n.2
  - ⇨nello sviluppo dei casi d'uso è fondamentale il diagramma delle attività
  - ⇨il file azioni.xml deve essere una trasposizione dei vari diagrammi
  - ⇨serve ad orientarsi e a documentare il flusso

Linee Guida

- Linea guida n.3
  - ⇨attenzione al rapporto tra azioni e modello
  - ⇨le azioni sono parte del controllo
  - ⇨non devono implementare la logica applicativa
  - ⇨la logica applicativa deve essere incapsulata per quanto possibile nei componenti

Linee Guida

- Linea guida n.4
  - ⇨la comunicazione tra gli strati è basata sullo scambio di stringhe
  - ⇨la correttezza delle stringhe non è controllabile a tempo di compilazione
  - ⇨sono una fonte di errore frequente
  - ⇨è opportuno fare molta attenzione

Linee Guida

- Linea guida n.5
  - ⇨per la verifica dell'applicazione è opportuno usare pagine di errore il più possibile esplicative
  - ⇨è necessario utilizzare il sistema di logging per documentare le operazioni effettuate dal controllore e dalle azioni

Linee Guida

- Linea guida n.6
  - ⇨è indispensabile limitare il codice nelle pagine jsp al minimo
  - ⇨il codice deve essere confinato nelle azioni
  - ⇨l'unico codice consentito è quello necessario per per produrre la risposta
  - ⇨è possibile usare librerie di tag Jsp per ridurre al minimo questo codice

Riassumendo

- Framework per Applicazioni Web
- Un Framework Didattico per J2EE
  - ⇨Il Manuale di pinco
- Diagrammi delle Attività
- Linee Guida

Termini della Licenza