Tecnologie di Sviluppo per il Web

Applicazioni Web J2EE: Conclusioni

versione 3.0

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

Sommario

- Evoluzione delle Tecnologie Web
- Test di Applicazioni Web
- Test Funzionali
- Test di Prestazioni
- Test di Affidabilità
- Operazioni di Costruzione

Evoluzione delle Tecnologie Web

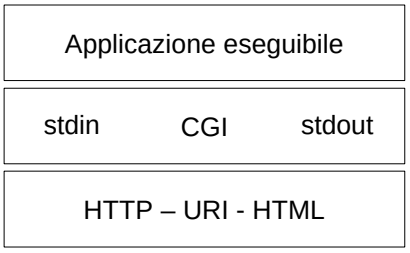
- In questi anni
  - ⇨le tecnologie per lo sviluppo Web si sono notevolmente evolute
- Il punto di partenza
  - ⇨HTTP, URI e HTML
  - ⇨protocolli non orientati allo sviluppo applicativo, ma al trasferimento di file
  - ⇨le successive evoluzioni sono state orientate a superare i limiti di questi protocolli

Evoluzione delle Tecnologie Web

- La linea evolutiva
  - ⇨introdurre un modulo intermedio detto server applicativo, capace di fornire servizi alle applicazioni lato server
  - ⇨per consentire di comunicare con i client attraverso i protocolli di base (HTTP, URI e HTML)
  - ⇨la complessità dei servizi è andata rapidamente crescendo

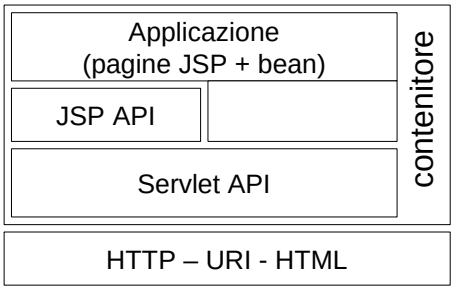
Evoluzione delle Tecnologie Web

- Il primo approccio: CGI
  - ⇨idea: server applicativo per l'esecuzione di processi sul server HTTP



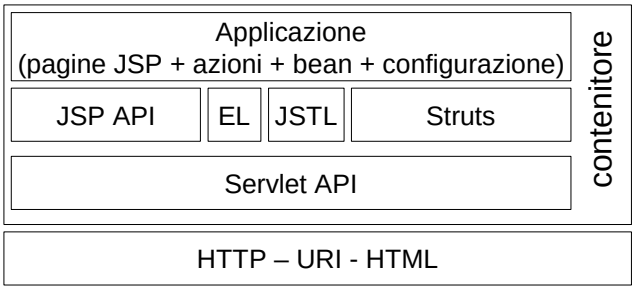
Evoluzione delle Tecnologie Web

- La tecnologia J2EE modello 1
  - ⇨servlet e pagine JSP



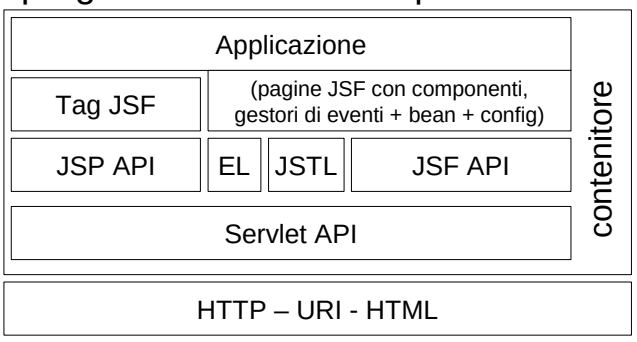
Evoluzione delle Tecnologie Web

- La tecnologia J2EE modello 2
  - ⇨separazione del codice tra vista e controllo



Evoluzione delle Tecnologie Web

- Java Server Faces
  - ⇨la programmazione a componenti



Test di Applicazioni Web

- Test e al debugging delle applicazioni Web
  - ⇨attività delicata per via della complessità dell'infrastruttura
- Molte possibili cause di errori
  - ⇨errori logici del programmatore
  - ⇨utilizzo scorretto del server applicativo (es: sessione)
  - ⇨errori nella persistenza (es: query SQL)
  - ⇨errori nella produzione della risposta HTML

Test di Applicazioni Web

- Alcune linee guida
  - ⇨verificare accuratamente il codice HTML prodotto chiedendo al browser i sorgenti
  - ⇨effettuare la convalida
  - ⇨verificare le pagine con più di un browser
  - ⇨utilizzare sistematicamente il sistema logging per avere informazioni sul funzionamento dell'applicazione

Test di Applicazioni Web

- Utilizzo dei file di log
  - nella cartella %TOMCAT_HOME%\logs
  - metodo void log(String messaggio) di ServletContext: stampa il messaggio nel file di log del server (es: localhost_log.2002-06-03.txt))
  - con Tomcat è possibile utilizzare anche System.out.println: la stampa avviene in catalina.out (stdout.log su Windows)

Test di Applicazioni Web

- Utilizzo delle pagine di errore
  - l'attributo errorPage della direttiva page
  - l'elemento error-page di web.xml
  - NON è opportuno utilizzare queste funzionalità durante la fase di sviluppo
  - perchè nascondono le eccezioni generate
  - vanno viceversa utilizzate nella versione di produzione

Test di Applicazioni Web

- Test di regressione
  - per quanto riguarda lo strato di modello e persistenza valgono le considerazioni note
  - è indispensabile sviluppare batterie di test complete sia per la logica applicativa del modello
  - sia per i componenti della persistenza
- In aggiunta
  - è opportuno sviluppare test funzionali

Test Funzionali

- Test funzionale dell'applicazione web
  - test che simula il comportamento di uno o più utenti che visitano l'applicazione
  - effettuando richieste HTTP
  - navigando i collegamenti
  - sottomettendo le form
  - e facendo assezioni sui risultati

Test Funzionali

- HttpUnit
  - un framework open source basato su JUnit per la scrittura di test funzionali su siti Web
  - consente di simulare l'attività di un utente che naviga l'applicazione usando il browser
- Nota
  - HttpUnit può essere utilizzato per effettuare test su qualsiasi tipologia di sito Web (anche siti non basati su applicazioni J2EE)

Test Funzionali

- Installazione
  - scaricare e decomprimere lo .zip da httpunit.sourceforge.net
- Per usare il framework
  - è necessario avere nel classpath alcuni jar
  - httunit.jar
  - Tidy.jar oppure nekohtml.jar
  - xerces.jar
  - per verificare il codice JavaScript: js.jar

Test Funzionali

- Un test HttpUnit
  - ⇨è un test di JUnit
  - ⇨in cui vengono usate le classi delle API di HttpUnit (com.meterware.httpunit)
- La classe essenziale: WebConversation
  - ⇨consente di simulare una sessione di lavoro tra l'applicazione e un client
  - ⇨è in grado di mantenere lo stato della sessione (es: cookie)

Test Funzionali

- Per effettuare una richiesta http
  - ⇨creare una conversazione (oggetto WebConversation)
  - ⇨creare una richiesta (oggetto WebRequest) specificando l'URI
  - ⇨inoltrare la richiesta usando la conversazione e prelevare il riferimento alla risposta (oggetto WebResponse)

Test Funzionali

- Esempio
  - WebConversation wc = new WebConversation();
  - WebRequest req = new GetMethodWebRequest("http://localhost:8080/aci");
  - WebResponse resp = wc.getResponse(req);
- In breve
  - WebConversation wc = new WebConversation();
  - WebResponse resp = wc.getResponse("http://localhost:8080/aci");

Test Funzionali

- Una volta ottenuta la risposta
  - è possibile analizzarne il contenuto in vari modi ed effettuare asserzioni
- Modalità di analisi della risposta
  - come puro testo (metodo resp.getText())
  - come DOM (metodo resp.getDOM())
  - per componenti (link, form, tabelle ecc.) utilizzando i metodi offerti dalle API

Test Funzionali

- Elementi della pagina analizzabili
  - ⇨link (ancore): oggetti di tipo WebLink ottenibili con il metodo WebLink getLinkWith(<ancora>) di WebResponse
  - ⇨form: oggetti di tipo WebForms, ottenibili con il metodo WebForms[] getForms()
  - ⇨tabelle: oggetti di tipo WebTable ottenibili con il metodo WebTable[] getTables()

Test Funzionali

- Operazioni con i link
  - ⇨il metodo fondamentale di WebLink è void click()
  - ⇨genera una richiesta http di tipo GET all'URI destinazione del link
  - ⇨per ottenere il riferimento all'oggetto WebResponse prodotto è possibile utilizzare, dopo click(), il metodo WebResponse getCurrentPage() di WebConversation

Test Funzionali

>> JavaDoc di httpUnit

- Operazioni con le form
  - due tipologie di operazioni
  - accesso e modifiche ai valori dei controlli: String getParameterValue(<nome>), void setParameter(<nome>, <valore>)
  - sottomissione della form: metodo void submit() di WebForm
  - per accedere alla risposta: wc.getCurrentPage()

Test Funzionali

- Un tipico test scritto con HttpUnit
  - essendo un test funzionale, richiede di effettuare varie richieste una dopo l'altra
  - es: accesso a index, esecuzione di un caso d'uso, uscita dall'applicazione
  - normalmente il codice è di una certa lunghezza ed è opportuno che sia spezzato in vari metodi

Test Funzionali

>> TestPartita

- Un esempio di test con HttpUnit
  - ⇨test della partita con indovinam2pinco
  - ⇨una richiesta iniziale alla pagina indice, con un'asserzione sul contenuto
  - ⇨sottomissione della form con il nome, con asserzioni sul contenuto della risposta
  - ⇨sottomissione di un tentativo, con asserzioni sul contenuto della risposta
  - ⇨sottomissione della form di interruzione

Test di Prestazioni

- Una caratteristica nuova delle appl. Web
  - ⇨la loro natura distribuita e multiutente
  - ⇨questo fatto rende opportuno condurre una serie aggiuntiva di test: test di prestazioni
- Test di prestazioni
  - ⇨consiste nel verificare la capacità dell'applicazione di fornire risposte in tempi accettabili quando viene sollecitata con carichi di lavoro opportuni

Test di Prestazioni

- In concreto
  - ⇨un test di prestazioni è tipicamente un test funzionale
  - ⇨ma eseguito in modo tale da simulare l'esecuzione non di un utente unico, ma di molti utenti contemporaneamente
  - ⇨e misurando i tempi di risposta forniti dall'applicazione alle risposte a seguito del crescere degli utenti

Test di Prestazioni

- Uno strumento per i test di prestazioni
  - ⇨Apache JMeter
- Apache JMeter
  - ⇨progetto open source della Apache Software foundation scaricabile da jakarta.apache.org
  - ⇨applicazione Java Swing che consente di effettuare test di prestazioni interattivi su molti tipi di applicazioni (Java e non)

Test di Prestazioni

- Installazione di JMeter
  - ⇨scaricare lo .zip dal sito e decomprimerlo
  - ⇨eseguire il file .bat per avviare l'applicazione
- I test di JMeter
  - ⇨per eseguire un test bisogna creare un "test plan"
  - ⇨al test plan possono essere aggiunti i vari elementi che costituiscono un test di JMeter

Test di Prestazioni

- Gli elementi di un test
  - ⇨thread group
  - ⇨sampler
  - ⇨listener
  - ⇨timer
  - ⇨config element
  - ⇨pre e post processor
  - ⇨controllers

Test di Prestazioni

- La terminologia di JMeter
  - ⇨“thread group”: elemento che consente di simulare vari utenti concorrenti
  - ⇨“n. of threads”: numero di thread da utilizzare (ognuno simula un utente)
  - ⇨“ramp up period”: periodo di tempo entro il quale far salire il numero di thread da 0 fino al valore fissato
  - ⇨“loop count”: numero di ripetizioni del test

Test di Prestazioni

- La terminologia di JMeter (continua)
  - ⇨“sampler”: elemento che corrisponde ad un’operazione da effettuare nel test (per un’applicazione Web ad una richiesta http)
  - ⇨per un sampler di tipo richiesta HTTP è possibile specificare l’URI, il metodo ed eventuali parametri della query string
  - ⇨inoltre è possibile aggiungere “assertions” sul risultato fornito dal server

Test di Prestazioni

- La terminologia di JMeter (continua)
  - ⇨“listener”: componente che analizza i risultati del test e li visualizza in formato opportuno
  - ⇨i listener più utili sono il “graph results” e il “view results tree”
  - ⇨il primo costruisce un grafico dell’andamento dei tempi di risposta
  - ⇨il secondo riassume tutte le richieste e risposte HTTP eseguite durante il test

Test di Prestazioni

- Le voci del graph results listener
  - ⇨“throughput”: stima del numero di richieste che il server può servire al minuto
  - ⇨“data”: tempo di risposta in millisecondi per ciascuna richiesta
  - ⇨“average”, “median”, “deviation”: andamento del tempo medio di risposta, della mediana e della deviazione standard

Test di Prestazioni

- La terminologia di JMeter (continua)
  - ⇨“timer”: elemento che introduce ritardi tra una richiesta e l’altra di un thread
  - ⇨ce ne sono di vari tipi e servono a simulare per quanto possibile le pause di un utente
  - ⇨“http request defaults”: parametri validi per tutte le richieste (es: host e porta)

Test di Prestazioni

- La terminologia di JMeter (continua)
  - ⇨i “config element” consentono di configurare vari aspetti del test
  - ⇨es: “http cookie manager”: serve a gestire i cookie inviati dal server
  - ⇨nota: per sua natura JMeter funziona molto bene con i cookie, meno bene con gli URI riscritti

Test di Prestazioni

- La terminologia di JMeter (continua)
  - ⇨in caso di URI riscritti è necessario applicare alla risposta fornita un “post-processor”
  - ⇨elemento in grado di estrarre dalla risposta precedente il parametro JSESSIONID da utilizzare nella richiesta successiva
  - ⇨e poi applicarlo alla richiesta utilizzando un “pre-processor”

Test di Prestazioni

- Flusso di controllo nel test
  - ⇨un test ordinario è fatto semplicemente di una sequenza di richieste da effettuare in ciascun thread del thread group
  - ⇨ma JMeter consente di specificare “controllers” per rendere più sofisticato il flusso di controllo
  - ⇨esempi: controlli di loop, controlli condizionali, controlli random ecc.

Test di Prestazioni

>> test su acim2pinco
>> test su indovinam2pinco

- Sviluppo dei test
  - ⇨completamente grafico, aggiungendo gli elementi necessari al test plan usando il mouse
  - ⇨c'è uno spazio di lavoro (il "workbench") che può essere usato per "appoggiare" elementi temporanemente rimossi dal test
  - ⇨successivamente è possibile salvare e ricaricare il test sviluppato

Test di Prestazioni

- La grande utilità di JMeter
  - ⇨consente di avere un riscontro facilmente leggibile delle prestazioni dell'applicazione
  - ⇨e di stimare i tempi di risposta attesi
- Il grande svantaggio di JMeter
  - ⇨non è uno strumento orientato ai test di regressione (anche se esiste un'interfaccia a console per l'esecuzione di test da file)
  - ⇨si tratta in realtà di un profiler per appl. Web

Test di Prestazioni

- Test di prestazione di regressione
  - ⇨JUnitPerf
- JUnitPerf
  - ⇨strumento open source scaricabile da www.clarkware.com/software/JUnitPerf.html
  - ⇨consente di trasformare un test JUnit in un test di prestazioni “decorandolo” in vari modi
  - ⇨per usarlo: scaricare e decomprimere lo zip e aggiungere junitperf.jar al classpath

Test di Prestazioni

- Le decorazioni previste da JUnitPerf
  - ⇨due classi principali utilizzate per decorare un test ordinario
  - ⇨TimedTest: test su cui può essere espresso un vincolo sul tempo complessivo di esecuzione
  - ⇨LoadTest: test che viene eseguito simulando un numero di utenti fissato
  - ⇨RepeatedTest: test da eseguire varie volte

Test di Prestazioni

>> TestPartitaPrestazioni

- Per costruire il test di JUnitPerf
  - si costruisce una nuova classe di test
  - nel metodo suite() della nuova classe di test si decora un test esistente
  - in particolare, si costruisce un oggetto di tipo junit.framework.Test utilizzando la classe TestFactory
  - da quello si costruisce un TimedTest o un LoadTest o un RepeatedTest, con parametri fissati, eventualmente combinando tipologie diverse

Test di Prestazioni

- Attenzione
  - l'attività di test delle prestazioni di un'applicazione Web è un'attività essenziale
  - ma deve essere condotta con criterio e seguendo linee guida precise
  - nel seguito discutiamo alcune di queste linee guida
- Linea guida n. 1
  - prevedere test sistematici di prestazioni

Test di Prestazioni

- Linea guida n. 2
  - ⇨come stabilire le prestazioni attese ?
  - ⇨si tratta di requisiti non funzionali da decidere in fase di analisi dei requisiti
  - ⇨in particolare: numero di utenti attesi, tempi di risposta attesi
  - ⇨dipendono in grande misura dalla natura dell'applicazione Web

Test di Prestazioni

- Normalmente
  - ⇨per i tempi di risposta è opportuno non superare i 1-2 secondi
- Una buona strategia
  - ⇨effettuare profiling periodici con JMeter per stimare tempi e carichi
  - ⇨utilizzare i valori stimati con JMeter per stabilire soglie per JUnitPerf

Test di Prestazioni

- Linea guida n. 3
  - ⇨attenzione ai colli di bottiglia
  - ⇨la macchina client e la macchina server devono essere sempre diverse, per evitare di sovraccaricare il server
  - ⇨inoltre non è opportuno aumentare eccessivamente il numero di utenti concorrenti sul client, perchè in quel modo il client diventa un collo di bottiglia

Test di Prestazioni

- Infatti
  - ⇨l'obiettivo è verificare le prestazioni del server e non del client
  - ⇨è quindi opportuno verificare l'andamento della CPU sia sulla macchina client che sulla macchina server durante il test, ed accertarsi che la CPU sulla macchina client non sia satura
  - ⇨lo stesso vale per l'eventuale DBMS (accertarsi che non sia il DBMS il collo di b.)

Test di Prestazioni

- Linea guida n. 4
  - ⇨attenzione all'infrastruttura
  - ⇨bisogna tenere in conto i tempi di trasferimento sulla rete, effettuando test in rete isolata e in rete pubblica
  - ⇨inoltre bisognerebbe effettuare verifiche in rete pubblica in condizioni diverse di carico (giorni diversi e orari diversi), e considerare valori minimi, massimi e medi

Test di Affidabilità

- Finora
  - ⇨abbiamo sempre considerato l'esecuzione dei test come una attività da svolgere in fase di sviluppo
- Ma, per le applicazioni Web
  - ⇨l'affidabilità, disponibilità e prestazioni sono requisiti spesso essenziali
  - ⇨sarebbe opportuno condurre test anche sulla versione in produzione

Test di Affidabilità

- Al solito
  - ⇨sulle versioni di produzione si possono condurre solo test non intrusivi; es: letture della base di dati e non aggiornamenti
  - ⇨per evitare che gli utenti vedano i dati introdotti dal test
  - ⇨inoltre bisogna fare attenzione a non sovraccaricare l'applicazione con test di prestazione molto duri per evitare disservizi

Test di Affidabilità

- Test di affidabilità
  - ⇨un test di affidabilità è un test di regressione condotto sistematicamente sull'applicazione in produzione per verificare che stia funzionando correttamente
  - ⇨tipico funzionamento: il test viene pianificato regolarmente su una macchina client (ogni giorno, ogni 12 ore, ogni ora...); se il test fallisce, viene inviato un messaggio di segnalazione allo sviluppatore

Test di Affidabilità

- Le tecniche per farlo
  - scrivere gli ordinari test con JUnit, HttpUnit, JUnitPerf
  - predisporre un file di build di ant per eseguire i test
  - scrivere uno script della shell (file batch o script .sh) su una macchina client che esegua i test utilizzando ant
  - pianificare l'esecuzione periodica del file di ant sulla macchina client (che deve essere sempre accesa e connessa alla rete)

Test di Affidabilità

- Le tecniche per farlo (continua)
  - utlizzare il "mail logger" di ant per inviare messaggi di notifica ad indirizzi opportuni sui risultati del test
- Mail Logger di ant
  - componente di ant capace di effettuare logging inviando un messaggio di posta elettronica
  - per eseguirlo: ant –logger org.apache.tools.ant.listener.MailLogger
  - richiede varie proprietà per la configurazione
  - e opportuni jar per l'invio di messaggi

Test di Affidabilità

- Configurazione del logger
  - in un file logger.properties da importare nel file di build
  - MailLogger.mailhost=smtp.unibas.it
  - MailLogger.from=ant@unibas.it
  - MailLogger.failure.notify=true
  - MailLogger.success.notify=false
  - MailLogger.failure.to=admin@unibas.it
  - MailLogger.failure.subject=*** Build fallita

Operazioni di Costruzione

- Per le applicazioni Web
  - vari modelli di build
  - web-template-build: applicazioni web modello 1 a due strati
  - web-dbms-template-build: applicazioni web modello 1 a tre strati (importa web-template-build e dbms-template-build)
  - analoghe versioni per pinco, struts, java server faces

Operazioni di Costruzione

- Nota
  - ⇨l'esecuzione dei test funzionali e test di prestazione può richiedere un tempo significativo
  - ⇨è opportuno separare i package di questi test dai package di test di unità in modo da poterli eseguire separatamente
  - ⇨nei file di build target separati

Operazioni di Costruzione

- Esempio
  - ⇨nel file web-template-build
  - ⇨target test-modello: **/modello/Test*.*
  - ⇨target test-persistenza: **/persistenza/Test*.*
  - ⇨target test-funzionali: **/funzionali/Test*.*
  - ⇨target test-prestazioni: **/prestazioni/Test*.*
  - ⇨target test (che dipende da tutti i precedenti)
  - ⇨build dipende da test-modello
  - ⇨rebuild dipende da test

Riassumendo

- Evoluzione delle Tecnologie Web
- Test di Applicazioni Web
- Test Funzionali
- Test di Prestazioni
- Test di Affidabilità
- Operazioni di Costruzione

Termini della Licenza