Tecnologie di Sviluppo per il Web

Applicazioni Web J2EE:
Java Servlet
Parte b

versione 3.3

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

Sommario

- Operazioni Fondamentali
  - ⇨Richiesta
  - ⇨Risposta
  - ⇨Sessione
  - ⇨Contesto dell'Applicazione
  - ⇨Inoltro delle Richieste
- Riassumendo

Operazioni Fondamentali

- Un servlet è una classe Java
  - ⇨in un servlet è consentito tutto quello che è consentito in una classe Java
- Caratteristiche particolari
  - ⇨la classe viene istanziata dal contenitore
  - ⇨è orientata alla comunicazione client-server basata su HTTP
  - ⇨da questo discendono le operazioni fondamentali tipicamente fatte in un servlet

Operazioni Fondamentali

- Nota
  - ⇨le API javax.servlet.* e javax.servlet.http.* sono particolarmente complesse
  - ⇨nel seguito vengono schematizzati i metodi principali
  - ⇨per informazioni su ulteriori metodi, fare riferimento alla documentazione delle API in questione

Operazioni Fondamentali

- Gestione della richiesta
  - ⇨API HttpServletRequest
- Gestione della risposta
  - ⇨API HttpServletResponse
- Gestione della sessione
  - ⇨API HttpSession
- Gestione del contesto di applicazione
  - ⇨API ServletContext
- Inoltro delle richieste
  - ⇨API RequestDispatcher

Gestione della Richiesta

- Richiesta
  - ⇨javax.servlet.http.HttpServletRequest
  - ⇨eredita da javax.servlet.ServletRequest
- Due funzioni principali
  - ⇨contiene una mappa dei parametri forniti dall'utente nella "query string"
  - ⇨contiene informazioni sulla richiesta http (es: metodo, URI, intestazioni http)

Gestione della Richiesta

- Mappa dei parametri
  - ⇨accesso alle coppie nome=valore della query string (tipicamente da una form HTML)
  - ⇨due metodi fondamentali
  - ⇨String getParameter(String nome)
    fornisce il valore del del parametro “nome”
  - ⇨String[] getParameterValues(String nome)
    fornisce i valori del parametro “nome” (es: SELECT e RADIO)

Gestione della Richiesta

- Esempio:
  - ⇨String nome = request.getParameter(“nome”);
- Note:
  - ⇨se il parametro non esiste, il risultato è null
  - ⇨tutti i valori sono considerati di tipo String
  - ⇨è necessario normalmente convertirli nel tipo appropriato e convalidare i valori

Gestione della Richiesta

- Convalida dell'input
  - ⇨non necessariamente gli utenti forniscono parametri dei tipi corretti
  - ⇨valgono le usuali considerazioni relative all'input
  - ⇨è necessario convalidare i valori e fornire all'utente gli opportuni messaggi di errore
  - ⇨ma la tecnica di convalida è leggermente più articolata (>>)

Gestione della Richiesta

- Analisi delle informazioni della richiesta
  - ⇨numerosi metodi per ottenere informazioni sul messaggio di richiesta HTTP
- Varie categorie
  - ⇨informazioni sul client
  - ⇨informazioni sul metodo
  - ⇨informazioni sull'URI
  - ⇨informazioni sulle intestazioni HTTP

Gestione della Richiesta

- Informazioni sul client
  - String getRemoteAddr(): numero IP del client
  - String getRemoteHost(): nome del client
- Informazioni su metodo e protocollo
  - String getMethod(): metodo HTTP
  - String getProtocol(): versione di HTTP
- Informazioni sulle intestazioni HTTP
  - String getHeader(String nome)
- Informazioni sull'URI
  - String getScheme() protocollo (http, ftp…)
  - String getServerName() host virtuale richiesto
  - int getServerPort() porta TCP richiesta

Gestione della Risposta

- Risposta
  - ⇨javax.servlet.http.HttpServletResponse
  - ⇨eredita da javax.servlet.ServletResponse
- Due funzioni principali
  - ⇨consente di specificare le intestazioni http del messaggio di risposta
  - ⇨consente di produrre il corpo del messaggio di risposta stampandone il contenuto

Gestione della Risposta

- Attenzione all'ordine delle operazioni
  - ⇨devono rispecchiare la struttura di una risposta HTTP
  - ⇨è necessario impostare prima le intestazioni
  - ⇨in particolare Content-Type
  - ⇨quindi stampare il contenuto del corpo

Gestione della Risposta

- Intestazioni HTTP
  - ⇨void setHeader(String nome, String valore)
  - ⇨void addHeader(String nome, String valore)
    per header con più di un valore
- Esempio: evitare il caching della risposta

```
response.setHeader("Cache-Control","no-cache");
//necessario per i browser HTTP 1.0
response.setHeader("Pragma", "no-cache");
//per gli agenti che ignorano le altre due
response.setHeader("Expires", "" + System.currentTimeMillis());
```

Gestione della Risposta

- Una intestazione fondamentale
  - ⇨void setContentType(String contentType)
  - ⇨es: response.setContentType(“text/html”);
  - ⇨es: response.setContentType(“text/plain”);
  - ⇨è indispensabile impostarla prima di cominciare a produrre il contenuto del corpo altrimenti si verifica un errore nella produzione della risposta

Gestione della Risposta

- Corpo del messaggio
  - ⇨è necessario ottenere un oggetto PrintWriter su cui stampare
  - ⇨metodo PrintWriter getWriter()
  - ⇨metodi print() e println()
- Esempio

```
PrintWriter out = response.getWriter();
out.println(“<html><body>”);
out.println(“…”);
```

Gestione della Sessione

- Sessione
  - ⇨oggetto di tipo javax.servlet.http.HttpSession
  - ⇨rappresenta una sessione di lavoro tra client e server
  - ⇨gestita trasparentemente dal contenitore
  - ⇨il contenitore mantiene una tabella contenente i dati delle sessioni istaurate con i client e consente di manipolarli utilizzando l'oggetto session

Gestione della Sessione

- Ciclo di vita della sessione
  - ⇨il programmatore crea una nuova sessione; in questo momento il client non si è ancora unito alla sessione
  - ⇨il server propone al client di unirsi (es: inviando un cookie)
  - ⇨il client può decidere di unirsi (es: accettando il cookie)
  - ⇨da quel momento la sessione è stabilita

Gestione della Sessione

- Oggetto sessione
  - ⇨si ottiene utilizzando il metodo getSession() di HttpServletRequest; ci sono due forme
- Prima forma
  - ⇨HttpSession getSession()
  - ⇨semantica: restituisce la sessione a cui appartiene la richiesta oppure crea una nuova sessione se non c'è n'è una esistente

Gestione della Sessione

- Seconda forma
  - ⇨HttpSession getSession(boolean modo)
  - ⇨semantica: getSession(true) è equivalente a getSession()
  - ⇨getSession(false) restituisce la sessione a cui appartiene la richiesta oppure null
  - ⇨non inizializza una nuova sessione nel caso in cui la richiesta non sia già associata ad una sessione

Gestione della Sessione

- Gestione del ciclo di vita della sessione
  - ⇨boolean isNew(): verifica se il browser si è già unito alla sessione o meno
  - ⇨void invalidate(): elimina la sessione di lavoro dalla tabella delle sessioni del contenitore
- Nota
  - ⇨tipicamente la sessione viene chiusa unilateralmente dal contenitore allo scadere del timeout fissato (normalmente 30 minuti)

Gestione della Sessione

- Per acquisire e verificare una sessione
  - ⇨due possibili strategie (intercambiabili)
  - ⇨getSession() o getSession(true): non restituisce mai null, ma poi devo verificare se la sessione è nuova con isNew()
  - ⇨getSession(false); in questo caso devo poi verificare che il riferimento ottenuto non sia null prima di utilizzarlo

Gestione della Sessione

- Manipolare gli attributi della sessione
  - ⇨alla sessione è associata una mappa persistente per la durata della sessione
  - ⇨Object getAttribute(String chiave)
  - ⇨void setAttribute(String chiave, Object ogg)
  - ⇨void removeAttribute(String chiave)
  - ⇨es: HttpSession s = request.getSession(false);
    Partita partita = (Partita)s.getAttribute("partita");

Gestione della Sessione

- Un esempio
  - ⇨due utenti utilizzano l'applicazione X
- I richiesta: client A a ServletA
  - ⇨nel codice di ServletA viene creata la sessione con getSession() (o getSession(true)); il JSESSIONID è AD23E
  - ⇨il contenitore aggiunge una voce nella tabella delle sessioni; in questo momento la sessione è "nuova" (isNew() è true); il browser non si è ancora unito
  - ⇨ServletA salva il riferimento al bean#1 nella sessione
  - ⇨con la risposta il contenitore invia il cookie JSESSIONID= AD23E

Gestione della Sessione

- II richiesta: client A a ServletB
  - ⇨con la richiesta il client si unisce alla sessione (restituisce il cookie)
  - ⇨la sessione non è più nuova
  - ⇨nel codice di ServletB viene ottenuto il riferimento alla sessione esistente con getSession(false)
  - ⇨ServletB recupera il riferimento al bean#1 dalla sessione e procede l'elaborazione

Gestione della Sessione

- III richiesta: client B a ServletA
  - ⇨nel codice di ServletA viene creata una nuova sessione con getSession() (o getSession(true)); il JSESSIONID è FFDD3
  - ⇨il contenitore aggiunge una voce nella tabella delle sessioni; in questo momento la sessione è "nuova" (isNew() è true); il browser non si è ancora unito
  - ⇨ServletA salva il riferimento al bean#2 nella sessione
  - ⇨con la risposta il contenitore invia il cookie JSESSIONID= FFDD3

Gestione della Sessione

- IV richiesta: client B a ServletB
  - il client B decide di NON unirsi alla sessione; la richiesta viene considerata come non appartenente a nessuna delle sessioni esistenti
  - nel codice di ServletB l'esecuzione di getSession(false) restituisce null
  - NOTA: eseguendo getSession(true) verrebbe fabbricata una nuova sessione (es: BB2134) che non ha niente a che vedere con la precedente
  - in ogni caso ServletB NON è in grado di recuperare bean dalla sessione

Gestione della Sessione

- Attenzione alla differenza
  - tra sessione sul client e sessione sul server
- Sessione sul client
  - sequenza di richieste effettuate dall'avvio del browser fino alla chiusura della finestra
- Sessione sul server
  - sequenza di richieste collegate ad un oggetto di tipo HttpSession presente nella tabella (con un timeout fissato)

Gestione della Sessione

- In generale
  - ⇨le due sessioni possono terminare indipendentemente
- Nel nostro esempio: caso I
  - ⇨l'utente del client A chiude la finestra del browser; la sessione termina per il client (il cookie scade), ma resta in piedi sul server fino allo scadere del timeout; successivamente scade anche per il server

Gestione della Sessione

- Nel nostro esempio, caso II
  - ⇨l'utente del client A lascia la finestra aperta per un tempo superiore al timeout; il server chiude la sessione, che però resta aperta sul client
  - ⇨l'eventuale richiesta successiva provoca al server l'invio di un cookie con un id di sessione che non esiste più
  - ⇨il server ignora il cookie e considera la richiesta come l'inizio di una nuova sessione

Gestione del Contesto dell'Appl.

- ServletContext
  - ⇨oggetto di javax.servlet.ServletContext
  - ⇨rappresenta il "contesto dell'applicazione Web" in cui il servlet viene eseguito
  - ⇨tutti i servlet hanno un ServletContext
  - ⇨si ottiene con il metodo ServletContext getServletContext() ereditato da GenericServlet

Gestione del Contesto dell'Appl.

- Due funzioni principali
  - ⇨consentire la comunicazione tra i servlet dell'applicazione e il contenitore
  - ⇨manipolare gli attributi dell'applicazione
- Comunicazione tra servlet e contenitore
  - ⇨un esempio: il metodo String getRealPath(String pathVirtuale)

Gestione del Contesto dell'Appl.

- Un ulteriore esempio
  - ⇨i metodi di logging: consentono l'accesso da parte del servlet al sistema di logging del contenitore
  - ⇨void log (String msg)
  - ⇨void log (String msg, Throwable t)
  - ⇨logging limitato: su canale fissato (file di log del contenitore) e con livello fissato

Gestione del Contesto dell'Appl.

- Manipolare gli attributi dell'applicazione
  - ⇨funzione identica a quella della sessione: mappa di coppie "chiave", "oggetto"
  - ⇨Object getAttribute(String chiave)
  - ⇨void setAttribute(String chiave, Object ogg)
  - ⇨void removeAttribute(String chiave)
  - ⇨es: ServletContext sc = getServletContext();
  - ⇨Record r = (Record)sc.getAttribute("record");

Inoltro delle Richieste

- Nel caso dei servlet
  - ⇨frequentemente un servlet deve inoltrare la richiesta ad un altro servlet
- Effettuare l'inoltro
  - ⇨un servlet non dispone dei riferimenti agli altri servlet
  - ⇨non è possibile semplicemente richiamare il metodo di servizio del nuovo servlet nel metodo di servizio del primo servlet

Inoltro delle Richieste

- Due possibili soluzioni
  - ⇨redirezione della risposta
  - ⇨inoltro all'interno del contenitore
- Redirezione della risposta
  - ⇨corrisponde a produrre una risposta di tipo 301 o 304 per indicare al browser che è necessario richiedere un uri diverso
  - ⇨lo svantaggio è che richiede di innescare una nuova transazione http

Inoltro delle Richieste

- Inoltro all'interno del contenitore
  - ⇨i servlet possono chiedere al contenitore di effettuare l'inoltro
  - ⇨ovvero di chiamare, durante l'esecuzione del metodo di servizio del servlet A, il metodo di servizio del servlet B per produrre la risposta
- Vantaggio
  - ⇨in questo modo il tutto resta confinato nell'ambito di un'unica transazione http

Inoltro delle Richieste

- Request Dispatcher
  - ⇨oggetto capace di inoltrare richieste a servlet
  - ⇨si ottiene utilizzando il metodo RequestDispatcher getRequestDispatcher(String URI) di ServletContext
  - ⇨deve conoscere l'URI del servlet verso cui redirigere la richiesta
- Nota
  - ⇨non deve essere confuso con i filtri

Inoltro delle Richieste

- URI
  - ⇨deve cominciare con "/"
  - ⇨è relativo alla radice dell'applicazione Web
  - ⇨convenzione per gli URI espansi dal server
- Inoltro di richiesta e risposta
  - ⇨metodo void forward(HttpServletRequest req, HttpServletResponse res) di RequestDispatcher

Inoltro delle Richieste

- Attenzione agli URI: doppia convenzione
- URI relativi espansi dal browser
  - ⇨"/" si riferisce alla radice del server http
- URI relativi espansi dal contenitore
  - ⇨"/" si riferisce alla radice dell'applicazione
  - ⇨es: servlet-mapping
  - ⇨es: metodo getRequestDispatcher
  - ⇨es: metodo getRealPath

Inoltro delle Richieste

- Attenzione
  - ⇨è necessario eseguire l'eventuale inoltro prima di cominciare a produrre la risposta
  - ⇨altrimenti viene sollevata una eccezione
- Esempio

```
ServletContext sc = getServletContext();
RequestDispatcher rd = sc.getRequestDispatcher("/Esci");
rd.forward(request, response);
return;
```


Inoltro delle Richieste

- Una annotazione
  - ⇨durante l'inoltro, è possibile aggiungere informazioni alla richiesta in modo che siano accessibili al servlet destinatario
- Mappa di attributi della richiesta
  - ⇨una ulteriore mappa
  - ⇨void setAttribute(String nome, Object v)
  - ⇨Object getAttribute(String nome)
  - ⇨da non confondere con i parametri della query string

Inoltro delle Richieste

- Operazione di inclusione
  - l'altra operazione fornita dal RequestDispatcher
  - consente al servlet A di includere nella propria risposta il codice prodotto dal servlet B
  - si interrompe l'esecuzione del metodo di servizio di A (doGet() o doPost())
  - comincia l'esecuzione del corrispondente metodo di servizio B
  - B produce il corpo della risposta
  - il corpo della risposta prodotta da B viene incluso nel corpo della risposta prodotta da A
  - riprende l'esecuzione del metodo di A

Inoltro delle Richieste

- Differenza rispetto a forward
  - include può essere eseguito da A in qualsiasi momento
  - B non può modificare né il codice di stato né le intestazioni della risposta
  - utile per evitare di ripetere porzioni di codice comuni a più servlet/jsp
  - si definiscono in un servlet/jsp e si includono dove è necessario

Riassumendo

- Riassumendo
  - ⇨i servlet vengono gestiti dal contenitore
  - ⇨sono utilizzati per gestire gli eventi corrispondenti alle richieste http
- Gli oggetti utilizzati per il servizio
  - ⇨un pool di thread per la gestione delle risorse
  - ⇨un pool di istanze dei servlet per l'esecuzione dei metodi di servizio

Riassumendo

- Gli oggetti creati dal contenitore
  - ⇨un oggetto di tipo HTTPServletRequest per ciascuna richiesta HTTP
  - ⇨un oggetto di tipo HTTPServletResponse per ciascuna risposta HTTP
  - ⇨un oggetto di tipo HTTPSession per ciascuna sessione di lavoro con un client
  - ⇨un oggetto di tipo ServletContext unico per tutta l'applicazione

Riassumendo

- Comunicazione tra servlet
  - ⇨il contenitore mette a disposizione varie modalità di comunicazione
  - ⇨inoltro e l'inclusione di richieste
  - ⇨condivisione di risorse

Riassumendo

- Risorse condivise
  - ⇨i servlet condividono risorse attraverso le varie mappe di riferimenti disponibili
- Mappe di riferimenti
  - ⇨mappa della richiesta
  - ⇨mappa della sessione
  - ⇨mappa dell'applicazione

Riassumendo

- Mappa della sessione
  - ⇨è la più importante
  - ⇨sopravvive finchè dura la sessione di lavoro tra client e server
  - ⇨i dati sono accessibili a tutti i servlet che servono richieste appartenenti alla sessione
  - ⇨non ci sono problemi di sincronizzazione (un thread per volta viene eseguito nella sessione)

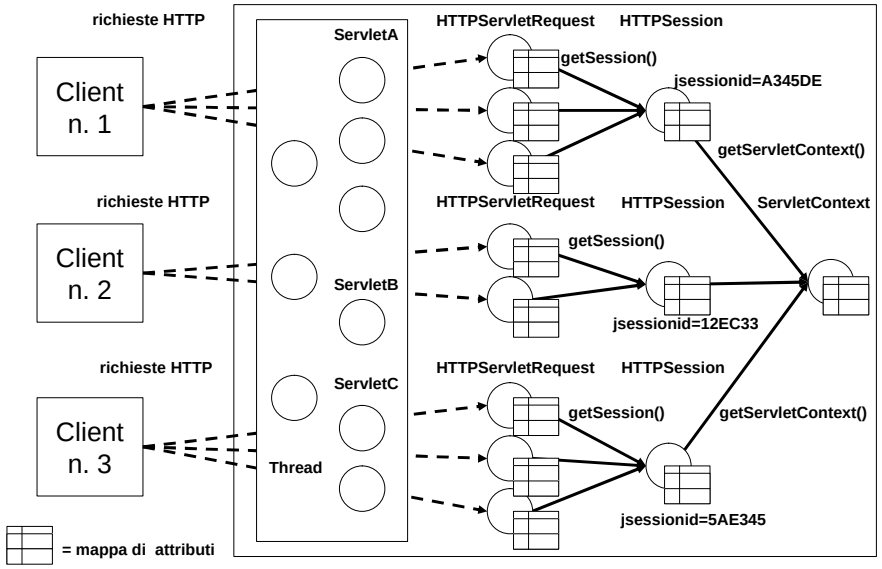
Riassumendo

- Mappa dell'applicazione
  - ⇨sopravvive finchè l'applicazione è installata nel contenitore
  - ⇨i dati sono accessibili a tutti i servlet che appartengono all'applicazione (servlet che servono richieste di sessioni diverse)
  - ⇨è necessaria la sincronizzazione per via della possibilità di accesso concorrente

Riassumendo

- Mappa della richiesta
  - ⇨è quella di più breve durata
  - ⇨sopravvive finchè vive la richiesta (cioè fino al termine della transazione http)
  - ⇨i dati sono accessibili a tutti i servlet si inoltrano la stessa richiesta
  - ⇨utile per trasmettere informazioni specifiche della richiesta durante l'inoltro
  - ⇨non pone problemi di sincronizzazione

>>> discussione codice - indovinaM1servlet



Riassumendo

- Operazioni Fondamentali
  - ⇨Richiesta
  - ⇨Risposta
  - ⇨Sessione
  - ⇨Contesto dell'Applicazione
  - ⇨Inoltro delle Richieste
- Riassumendo

Termini della Licenza