

Strumenti Java per lo Sviluppo di Applicazioni Web

**Corso di Progettazione e Analisi Orientata agli Oggetti
Università degli Studi di Udine – AA 2005/2006**

- **Introduzione a J2EE**
- J2EE Web Components
- Il Framework Struts
- Appendice Tecnica
- Bibliografia

- Piattaforma Java introdotta nel giugno 1999:

- J2SE – Java 2 Standard Edition

- Java per Personal Computer
- <http://java.sun.com/j2se>

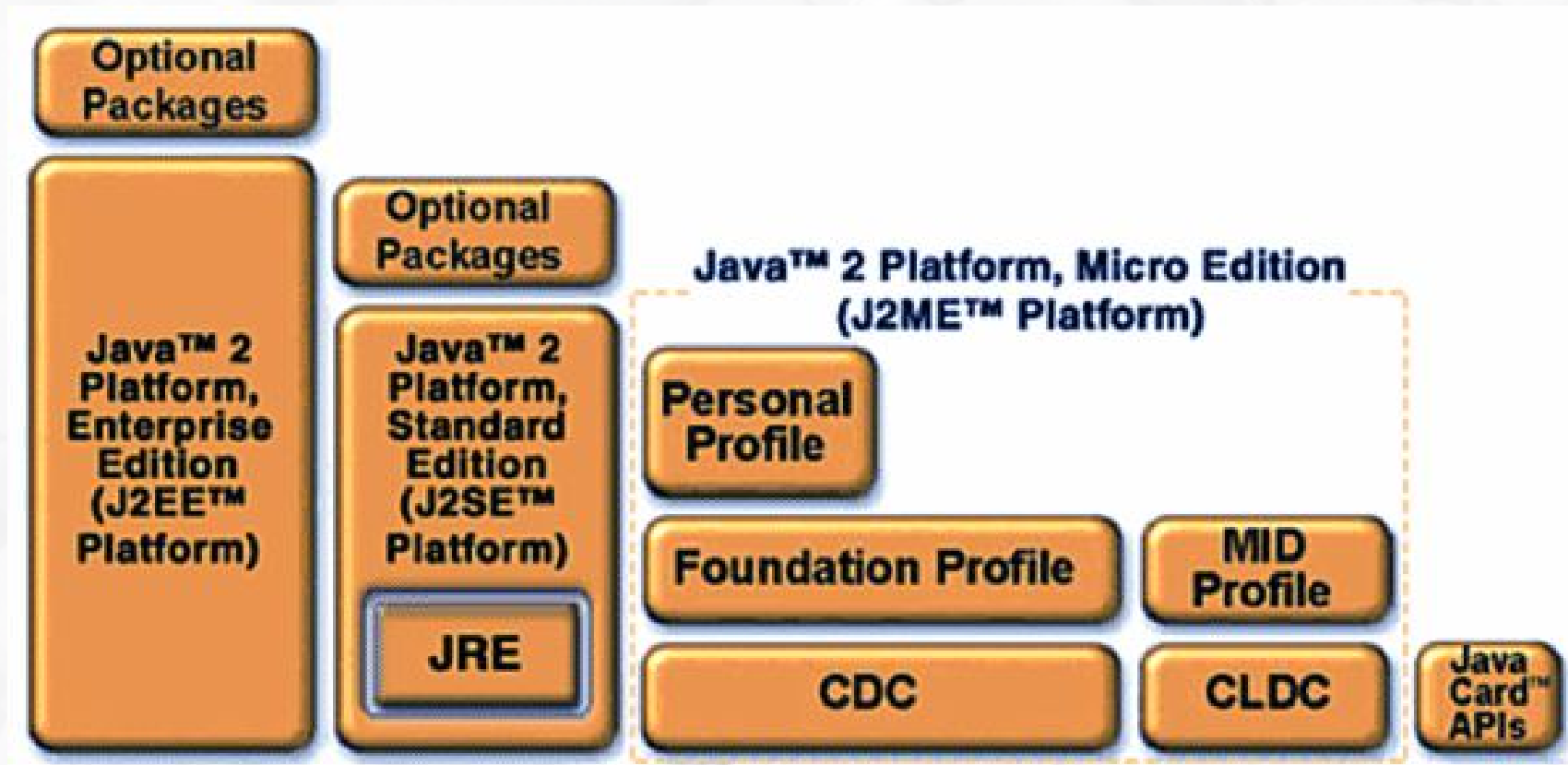
- J2ME – Java 2 Micro Edition

- Java per Dispositivi Portatili
- <http://java.sun.com/j2me>

- J2EE - Java 2 Enterprise Edition

- Java per Sistemi con Architetture Distribuite
- <http://java.sun.com/j2ee>

La Piattaforma Java



- *Una Piattaforma:* insieme delle librerie (API), dei tools di sviluppo e delle tecnologie messe a disposizione degli sviluppatori.
- *Una Specifica:* indica ai vendor i requisiti che le implementazioni dei loro prodotti devono soddisfare.
- *Compatibility Test Suite:* Usata dai vendors per verificare che la propria implementazione sia compatibile con le specifiche J2EE.
- *Design Guidelines:* destinate agli sviluppatori, in cui viene spiegato come usare al meglio le varie tecnologie.
- J2EE non è quindi nè un prodotto, nè un linguaggio, ma piuttosto una specifica per un'architettura distribuita, corredata da alcuni strumenti per lo sviluppo.

Finalità che si pone J2EE

- Superare le carenze del modello tradizionale Client-Server 2-Tier, che è solitamente facile da sviluppare e installare ma difficile da gestire, migliorare o aggiornare.
- Migliorare l'affidabilità la performance, la flessibilità, l'estensibilità la scalabilità, l'usabilità, nonché l'interoperabilità dei componenti sviluppati.
- Prendersi cura della gestione dei dettagli di “basso livello” delle applicazioni, semplificando così il lavoro del programmatore.

I componenti di un'applicazione J2EE

- Un'applicazione J2EE è formata da componenti, ovvero unità di software indipendenti (esempi di componenti sono: applets, Java Servlet, JavaServer Pages™ ed Enterprise JavaBeans™).
- Tra i software di supporto alle applicazioni, particolare importanza è assunta dall'Application Server (Web Container + EJB Container).
- Il codice scritto secondo le specifiche J2EE è (o per lo meno, dovrebbe essere) immediatamente portabile tra diversi application server conformi alle specifiche J2EE.

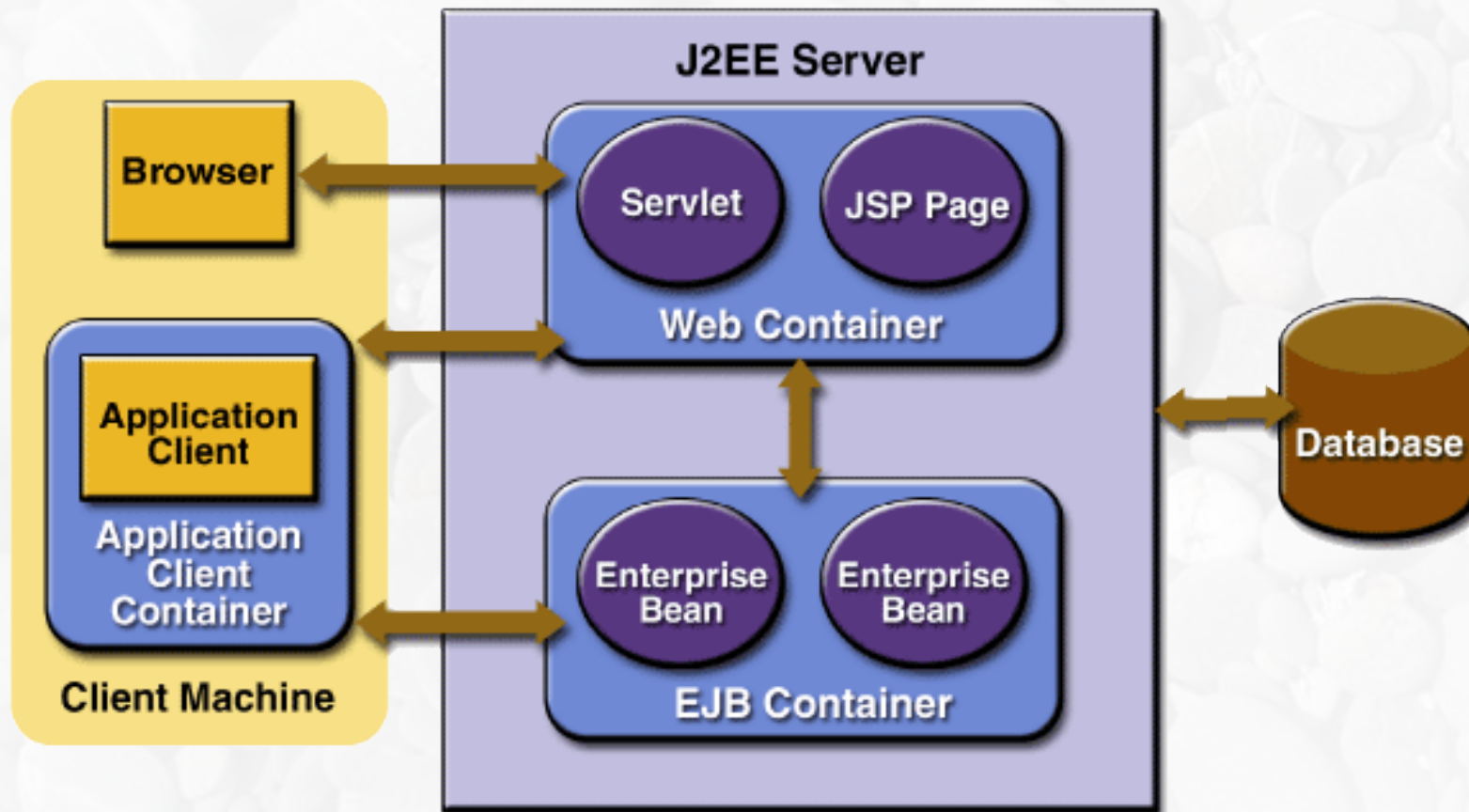
● *Components*

- Java Servlets
- JavaServer Pages (JSP)
- Enterprise JavaBeans (EJB)

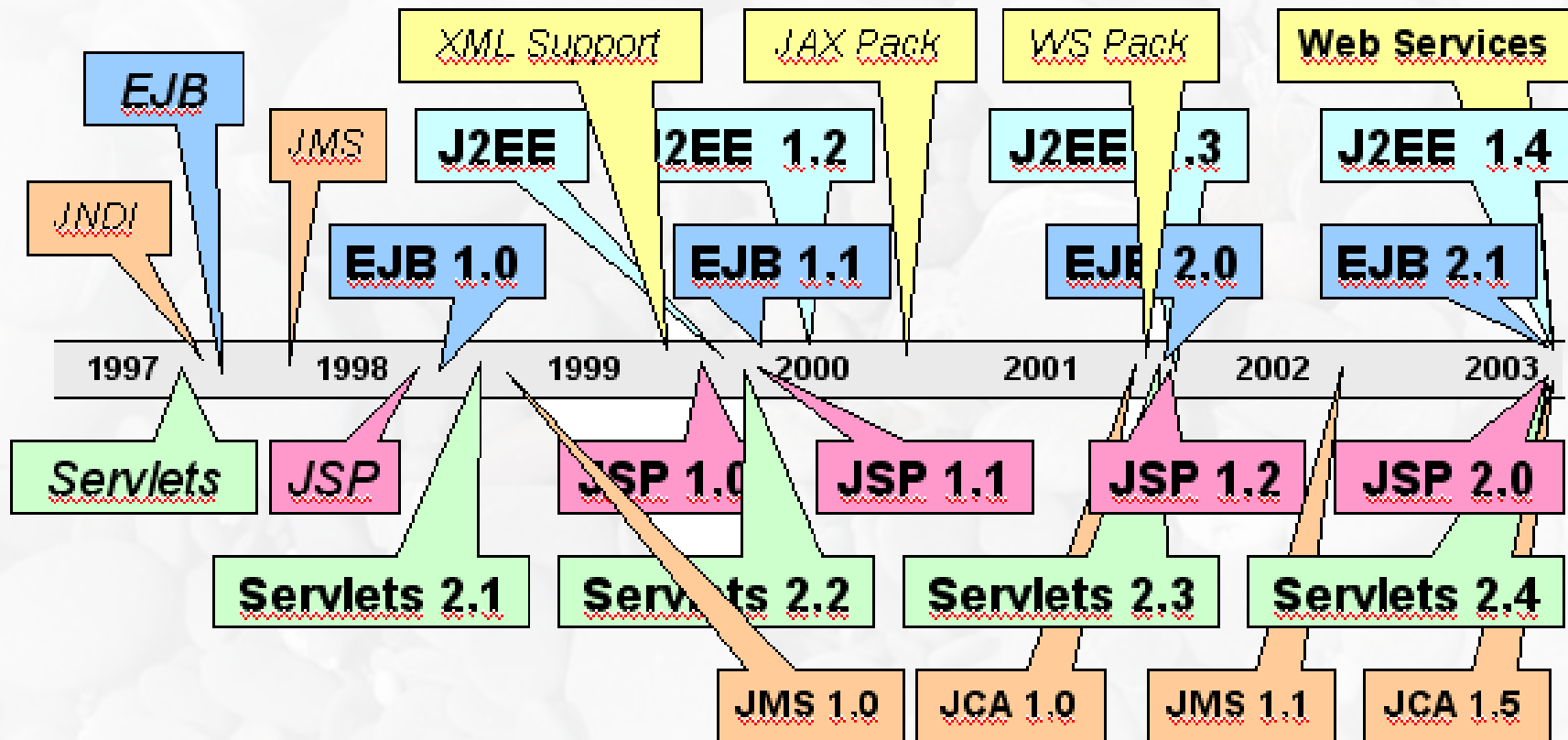
● Tecnologie di Supporto (API)

- Java database connectivity(JDBC)
- Java Messaging Service (JMS)
- Remote Method Invocations (RMI)
- Extensible Markup Languages(XML)
integrazione con Web Services
- Java Security
- JavaMail
- JavaIDL (Interface Description Language)
integrazione con CORBA

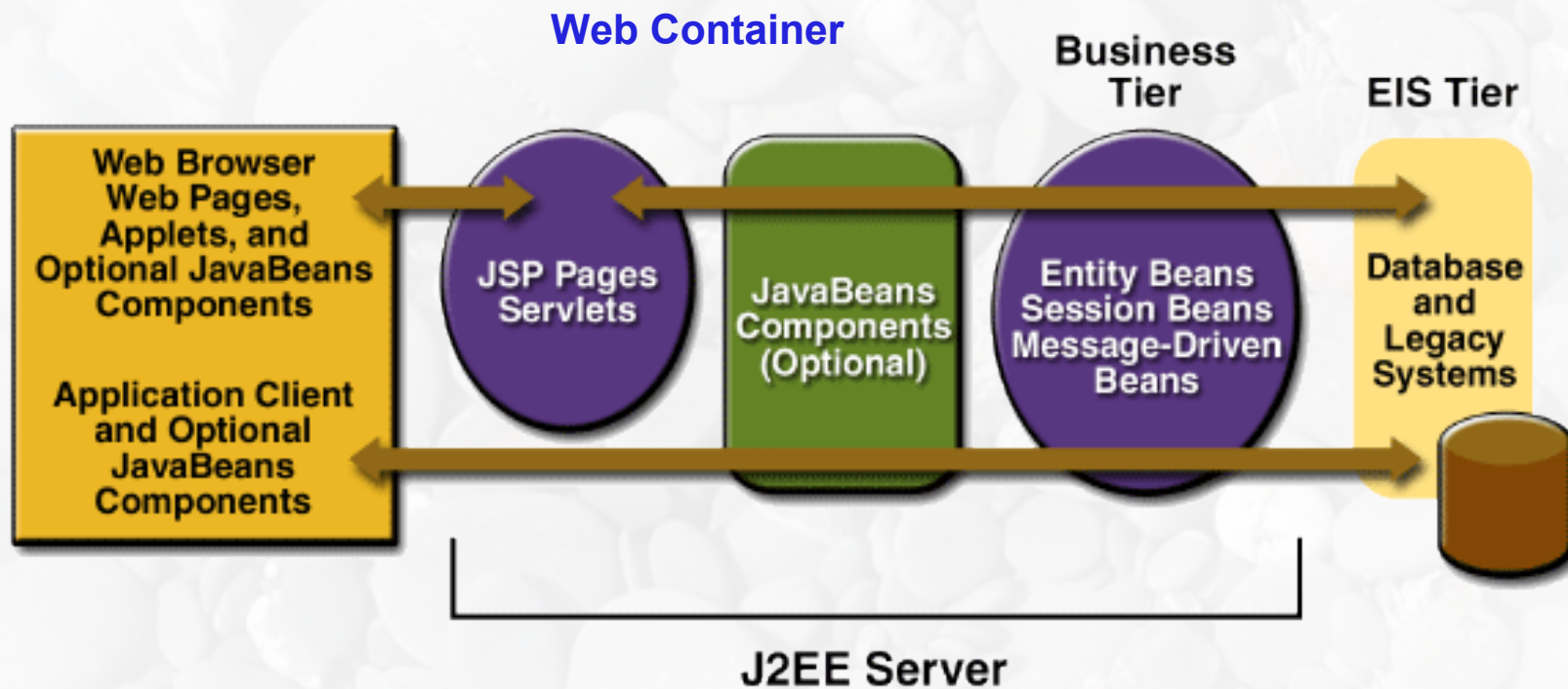
Architettura di un'ipotetica applicazione J2EE



J2EE Timeline



Possibili Interazioni tra componenti J2EE



Business Components

- Codice che svolge le mansioni tipiche di un particolare dominio applicativo ed è realizzato nel cosiddetto “*Business Tier*” (middleware), per mezzo degli “*Enterprise Beans*” (Session, Entity, Message-Driven).
- Questi sono contenuti nell'EJB Container, che fornisce, in maniera nativa, supporto per le **transazioni**, le **sessioni**, la **comunicazione**, la **sicurezza** e altri servizi. Lo sviluppatore può così concentrarsi maggiormente sui requisiti funzionali della propria applicazione.
- Gli Enterprise Beans sono componenti eseguiti dal lato Server, anche in maniera distribuita; la locazione d'accesso rimane trasparente ai Client. Gli EJB permettono l'esecuzione di un'applicazione su “thin Client”, poichè demandano a questi la sola presentazione.

Peculiarità degli EJB

- Operazioni standard per creare, distruggere, allocare e invocare istanze dei componenti. Il container si preoccupa di creare o distruggere istanze di un particolare Bean in base al carico del server usato e si preoccupa di effettuare caching, nel caso di dati provenienti da DB ove opportuno.
- Modello standard per definire un componente che intrattiene una sessione con il cliente, lasciando al container la gestione della sessione stessa.
- Modello standard per definire un componente che encapsula una fonte di dati (es. Database); il container si preoccupa di gestire il mapping “ad oggetti”-relazionale.
- Modello standard per definire le politiche di **sicurezza** sugli attributi di un componente.
- Modello standard per definire un componente che offre funzionalità di **Web Services**, con lo scambio di messaggi SOAP gestiti dal container.

- EJB sincroni che sono creati per agire direttamente sul sistema enterprise, compiere un'azione ed eventualmente ritornare un risultato al client.
- Corrispondono in pratica ai “verbi” per risolvere un particolare problema del dominio applicativo nel quale sono inseriti (es. `generaReportMensile`).
- Solitamente chiamati in causa con modalità descritte nelle specifiche J2EE ma possono anche operare direttamente come Web Services.

- EJB sincroni che vengono creati per incapsulare dati del sistema in cui sono inseriti. Questi dati possono quindi essere cercati, reperiti e distrutti dai clients.
- Hanno anche meccanismi che permettono il mapping con database relazionali e il reperimento di particolari record per mezzo di chiavi primarie (definite come classi).
- Possono essere considerati come i “soggetti” di un particolare dominio in cui sono inseriti (es. “cliente”).
- Referenze a questo tipo di Beans vengono solitamente fornite ai clients attraverso Session Beans, a seguito, ad esempio di operazioni di ricerca.

- EJB che permettono all'applicazione di gestire **messaggi** in modo asincrono.
- Fungono da JMS *message listeners*.
- Accettano messaggi da: componenti J2EE (application client, altri EJB, Web component) o anche da applicazioni che non usano tecnologia J2EE.
- Simili come funzionalità offerte a quelle dei Session Beans, ovvero effettuare operazioni nel dominio in cui sono inseriti; cambiano le modalità di invocazione (comunicazione asincrona con messaggi).

Da considerare però...

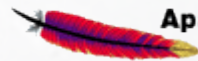
- Gli EJB hanno specifiche estese e complesse.
- Il tempo di sviluppo di un'applicazione basata su EJB è generalmente molto lungo.
- Nel caso di problemi, gli EJB sono ostici da debuggare.
- Il lungo lasso di tempo necessario per lo sviluppo di un'applicazione fa spesso sì che quando questa è pronta per il rilascio, nuove specifiche per la nuova versione degli EJB o nuove tecnologie sono maturate, rendendola di fatto obsoleta in partenza.
- Bottom Line: vanno usati con cautela e nei soli casi in cui si profili la necessità di soddisfare molte richieste di client e siano anche richieste altissima scalabilità e supporto nativo del container per transazioni.

J2EE Application Servers

- Implementano sia il Web Container che l'EJB Container



Sun Java System Application Server
Platform Edition 8



Apache Geronimo
Apache Geronimo
1.0-M5



ObjectWeb
JOnAS v4.4



Kingdee
Apusic v4.0



Trifork T4 Application Server



CAS
OnceAS2.0
Institute of Software, Chinese
Academy of Sciences



BEA
BEA WebLogic Server 9.0



Oracle
Oracle Application Server
Containers for J2EE 10g (10.1.3)



Tmax Soft
JEUS 5.0



IBM
IBM WebSphere Application Server



NEC
WebOTX 6.1



JBoss
JBoss Application Server 4.0

- Implementano solo il Web Container

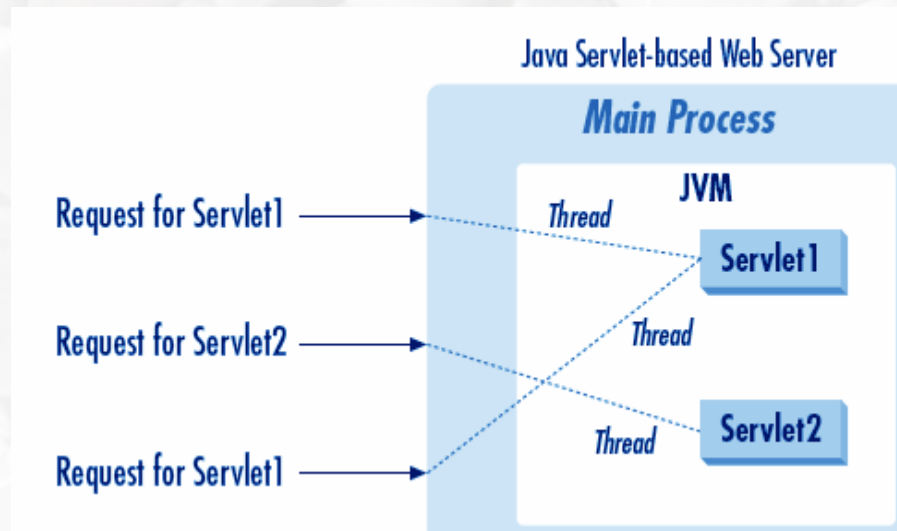


- Introduzione a J2EE
- **J2EE Web Components**
- Il Framework Struts
- Appendice Tecnica
- Bibliografia

- Negli anni '90 per permettere la generazione e presentazione di contenuti dinamici in applicazioni web veniva usato lo standard CGI (Common Gateway Interface)
 - Chiamate GET/POST del client processate e gestite da applicazioni a se stanti (es. scritte in C, Perl, Python)
 - Basse prestazioni: un processo per ogni richiesta
 - Nessun modo per gestire le sessioni
 - Programmazione a basso livello
 - Rischi di sicurezza (soprattutto in C)

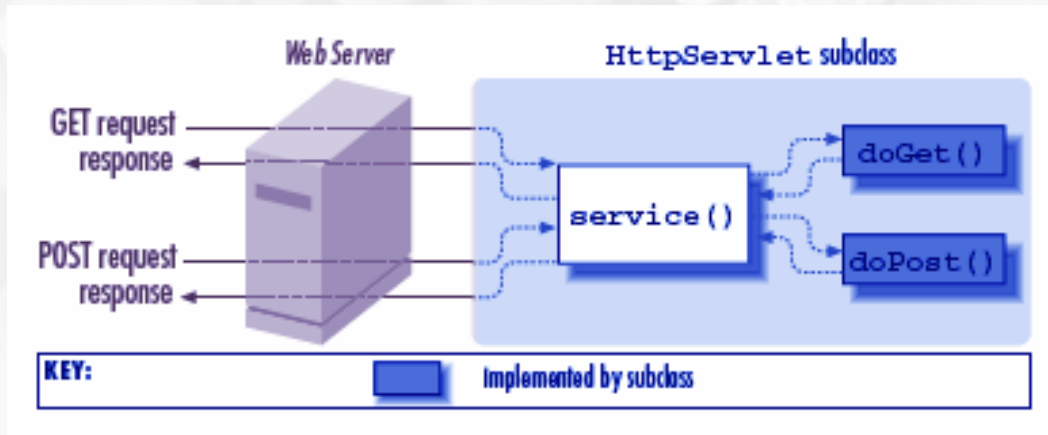
Introduzione ai Java Servlets

- I Servlet sono in pratica la versione Java di CGI
- Rispetto a questi sono tuttavia più semplici da realizzare, poichè “relativamente ad alto livello”.
- Sono altamente integrati in un server che li supporta; ciò migliora le prestazioni in quanto i Servlet vengono gestiti come threads del Server che li ospita, piuttosto che come nuovi processi separati.



Scrivere un Servlet

- Tutti i Servlet estendono la classe Servlet.
- In particolare, i Servlet http (decisamente i più usati) estendono la classe HttpServlet.
- I metodi solitamente sovrascritti di HttpServlet sono:
 - init, service, doGet/doPost, destroy (comuni)
 - doPut, doDelete, doOptions, doTrace (rari)



- NB: non c'è metodo main() nei Servlets.

I metodi di un Servlet

- `init()`: è chiamato solo all'avvio di un Servlet e generalmente serve a inizializzare i parametri/risorse comuni a tutte le richieste (es. accesso a database).
- `doGet(HttpServletRequest, HttpServletResponse)`: è chiamato ogni qual volta un Servlet riceve una richiesta GET. I due parametri rappresentano rispettivamente le informazioni presenti nella richiesta e la risposta.
- `doPost(HttpServletRequest, HttpServletResponse)`: del tutto analogo al metodo `doGet`, ma utilizzato in caso di richieste con il metodo POST.
- `destroy()`: è chiamato prima che il Servlet venga rimosso dalla memoria; per mezzo di questo metodo vengono effettuate tutte le operazioni di chiusura delle connessioni, “cleanup”, etc.

● Altri Metodi Comuni:

- `PrintWriter getWriter();`
Ottiene il `Writer` per scrivere l'output.
- `setContentType(String);`
Solitamente si imposta a `"text/html"`, per indicare che l'output è HTML da inviare in risposta al Browser.

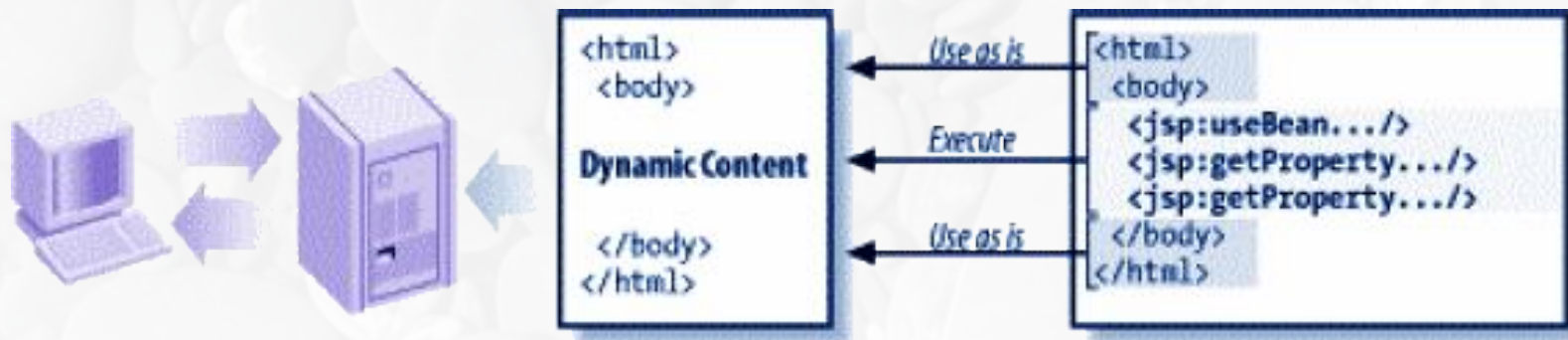
● Passaggio di Parametri:

- `String getParameter(String paramName)`
- `String[] getParameterNames()`
- `String[] getParameterValues()`
- Altri metodi per gestire immagini, files, cookies, etc.

Semplice Esempio di Servlet

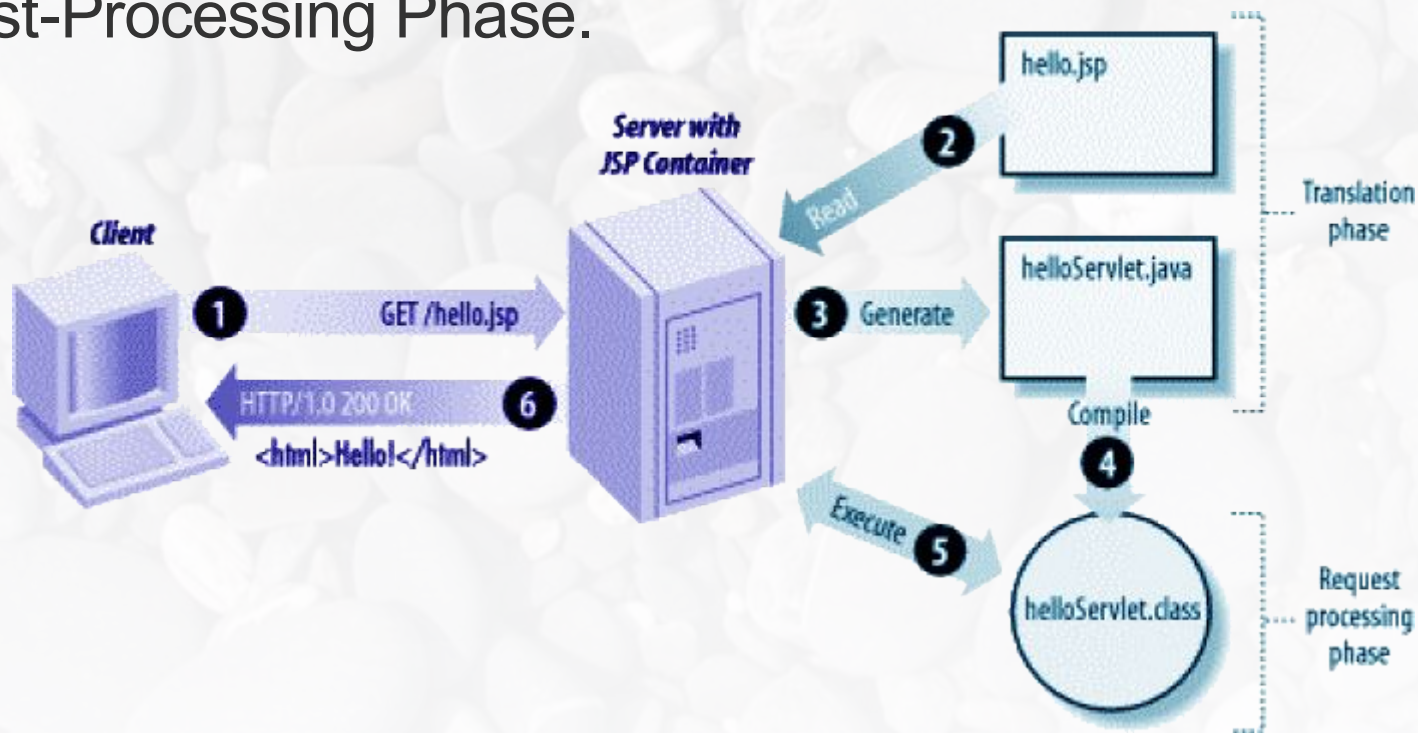
```
import java.io.*; //Apache Tomcat sample code
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter(); out.println("<html>");
        out.println("<body>");
        out.println("<head>");
        out.println("<title>Hello World!</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Hello World!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

- JavaServer Pages è una tecnologia per realizzare pagine web che includono contenuto generato dinamicamente.
- Una pagina JSP può contenere tag standard HTML, proprio come una pagina web comune; la differenza è che però questa può contenere anche alcuni elementi speciali che permettono l'inserimento di contenuto dinamico, ovvero che può variare a seconda di un particolare contesto.



Ciclo di Presentazione di una pagina JSP

- Una pagina JSP prima di poter essere fornita ad un client che ne faccia richiesta passa attraverso una cosiddetta Translation Phase, ovvero una fase dove viene convertita, dietro le quinte e, potenzialmente all'insaputa di chi l'ha realizzata, in una vera e propria Servlet.
- A questo punto può essere fornita al client con le modalità tipiche delle Servlet, in quella che viene denominata la Request-Processing Phase.



Elementi di una pagina JSP

● Directive

```
<%@ page import="java.util.", MVCApp.Cart, MVCApp.CartItem" %>
```

● Declaration

```
<%! Iterator it = null; CartItem ci = null; Vector cpi = null;%>
```

● Raw HTML

```
<html><head><title>Modulo di Autenticazione</title></head></html>
```

● Action

```
<jsp:usebean id = "Cart" scope = "session" class = "MVCApp.Cart"/>
```

● Scriptlets

```
<%  
Cpi = cart.getCartItems ( );  
it = cpi.iterator();  
While (it.hasNext()){ci= (CartItem)it.next();  
%>
```

● Expression

```
<td<% = ci.getTitle() %></td>
```

● Implicit Objects

```
<% string action = request.getParameter("action") ; %>
```

Esempio di semplice pagina JSP

```
<HTML>
  <HEAD>
    <TITLE>Prova Pagina JSP</TITLE>
  </HEAD>
  <% String bgColor = request.getParameter("COLOR"); %>
  <% if (bgColor == null) { %>
    <BODY BGCOLOR="FFFFFF" >
  <% } else { %>
    <BODY BGCOLOR="<%= bgColor %>" >
  <% } %>
  ESEMPIO
  <% if (bgColor == null) { %>
  Colore Non Specificato, userò il Bianco!
  <% } else { %>
    Ecco il colore richiesto: <%= bgColor %>
  <% } %>
</font></div>
  </BODY>
</HTML>
```

Interazione con JavaBeans – Cosa sono?

● NON Sono da Confondere con EJB!

Sono semplicemente classi che seguono le seguenti convenzioni:

- hanno un costruttore privo di parametri (zero-argument); eventualmente è possibile omettere i costruttori.
- non devono avere attributi di istanza pubblici (si usano invece i metodi getter e setter, rispettivamente `getXxx` e `setXxx`, dove *Xxx* indica il nome dell'attributo in questione)
NB Per attributi Booleani è possibile usare direttamente `isXxx` anziché `getXxx`.

- Introdotti poichè verranno usati nella maggior parte delle applicazioni e degli esempi che vedremo...

Interazione con JavaBeans – Il Bean

```
public class MyBean {
    // Initialize with random values
    int prop1 = (int)(Integer.MAX_VALUE*Math.random());
    String prop2 = ""+Math.random();

    public int getProp1() {
        return prop1;
    }
    public void setProp1(int prop1) {
        this.prop1 = prop1;
    }

    public String getProp2() {
        return prop2;
    }
    public void setProp2(String prop2) {
        this.prop2 = prop2;
    }
}
```

Interazione con Javabeans – La pagina JSP

```
<HTML><HEAD><TITLE>ESEMPIO BEAN PAO</TITLE></HEAD>
<jsp:useBean id="myBean" class="MyBean" scope="session">
  <!-- initialize bean properties -->
  <jsp:setProperty name="myBean" property="prop1" value="123" />
</jsp:useBean>

<BODY>Ecco l'Output:<BR />
<jsp:getProperty name="myBean" property="prop1" />
<jsp:getProperty name="myBean" property="prop2" />
</BODY></HTML>
```

- Output sul Browser:

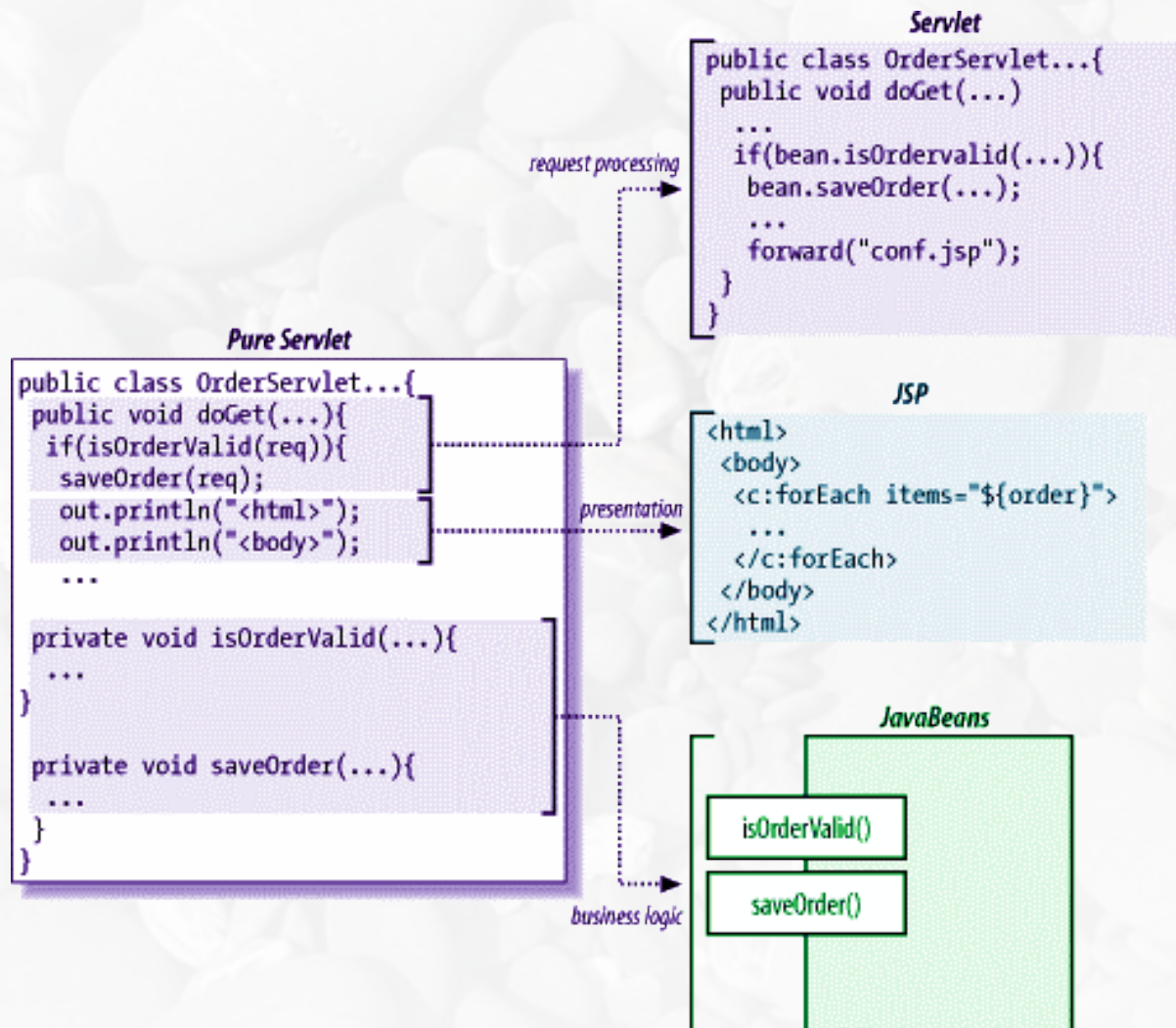
Ecco l'Output:
1234555

JSP e Servlets a Confronto

Servlets	JSPs
Java objects that subclass from HttpServlet	HTML page with embedded java code
Used for processing browser requests	Used for building web pages
Can be dynamically changed while the servlet container is running	Can be dynamically changed while the servlet container is running
Written by java programmers	Can be written by “web page folks”, not necessarily java programmers
Some configuration/setup is necessary to use (deploy) a servlet	Deployment is quick and easy
Can use “object things” like <i>methods</i> and <i>subclassing</i>	Defining methods or subclassing not easy or not possible

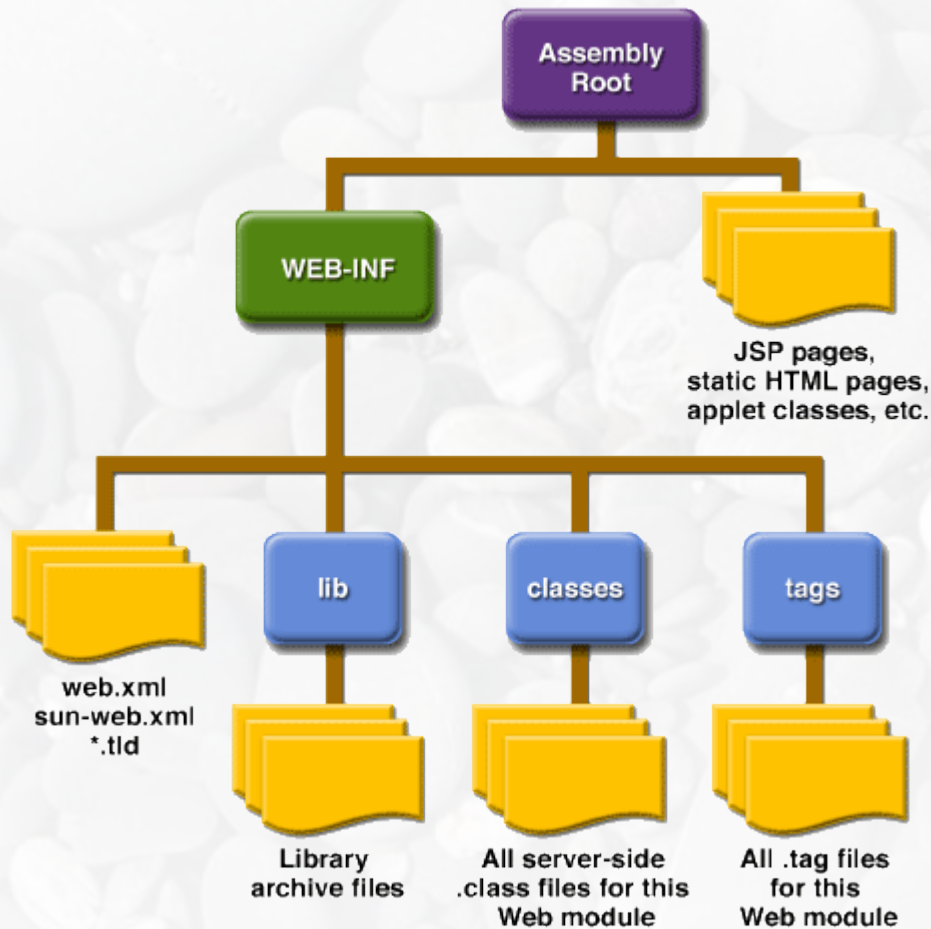
Combinare le due (tre) Cose

- Servlet e JSP possono essere utilizzati anche assieme...



Struttura di un'Applicazione Web J2EE I

- La presente struttura gerarchica può essere compressa in un file .war; il server provvederà alla decompressione e deployment.



Struttura di un'Applicazione Web J2EE II

- **Assembly Root:** è la directory principale dell'applicazione; può file html, jsp, immagini o altri elementi direttamente raggiungibili da un client tramite URI (eventualmente anche in sottodirectory).
- **WEB-INF** è una directory non direttamente raggiungibile da un client remoto, contenente i files di configurazione, nonché le classi e le librerie che compongono l'applicazione vera e propria

Altri J2EE Web Components

**JavaServer Pages
Standard Tag Library**

JavaServer Faces

JavaServer Pages

Java Servlet

- La JSP Standard Template Library (JSTL) è una componente rilasciata da Sun con l'intento di minimizzare la necessità di usare scriptlet (cosa che spesso peggiora notevolmente la leggibilità e la facilità di manutenzione) delle pagine JSP.
- JSTL è suddivisa in 4 componenti:
 - **Core Tag Library:** Contiene i tags usati nella maggior parte delle applicazioni web (ad esempio cicli e condizionali)
 - **Formatting/Internationalization Tag Library:** Contiene i tags usati per il parsing dei dati (ad esempio per l'internazionalizzazione delle date e dei formati).
 - **Database Tag Library:** Contiene tags che possono essere utilizzati per l'accesso a databases (consigliabile l'utilizzo solo durante il debug, poichè sarebbe meglio che l'accesso a database non fosse effettuato direttamente da pagine JSP).
 - **XML Tag Library:** Contiene tag per il processing di elementiXML.

Prima:

```
<html>
  <head>
    <title>Count to 10 in JSP scriptlet</title>
  </head>
  <body>
    <%
      for(int i=1;i<=10;i++)
    {%>
    <%=i%><br/>
    <%
    }
    %>
  </body>
</html>
```

Dopo:

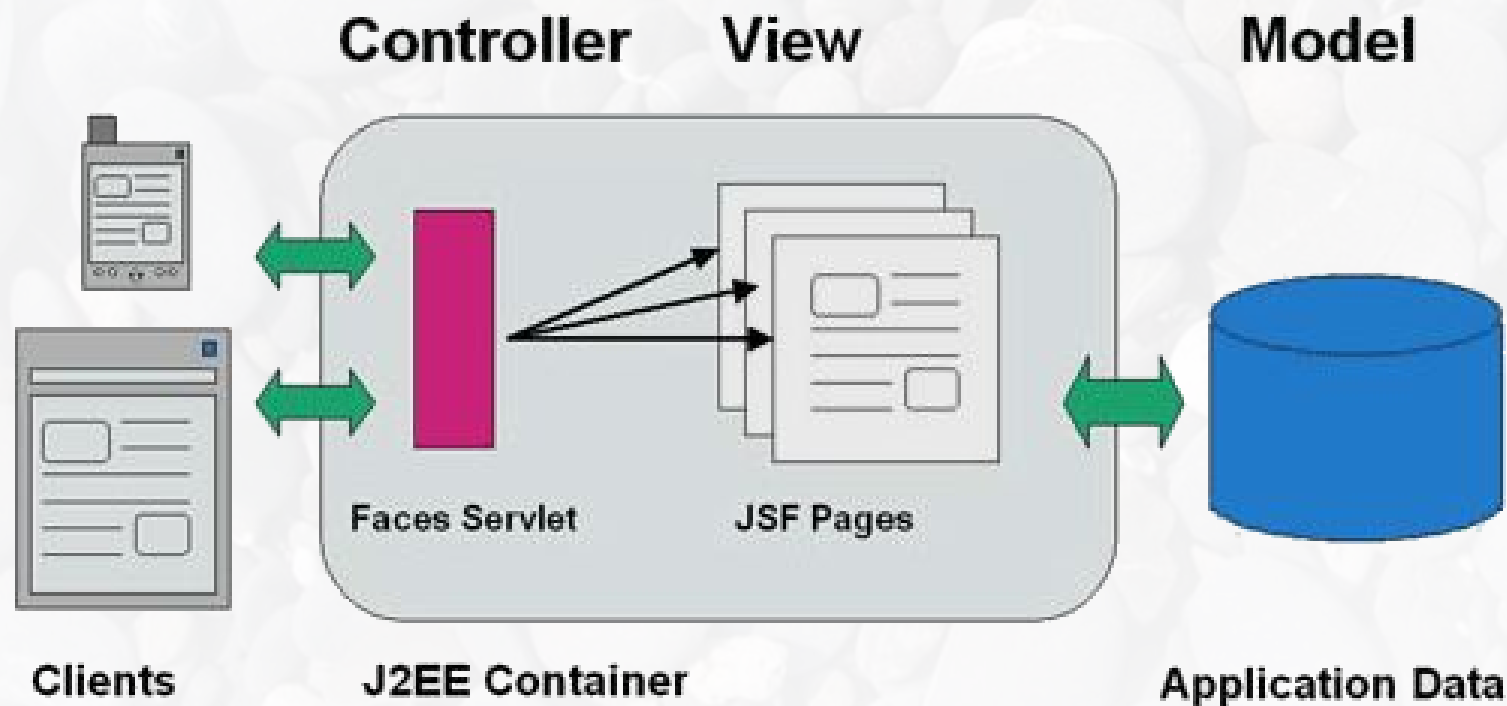
```
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
<html>
  <head>
    <title>Count to 10 Example (using JSTL)</title>
  </head>

  <body>
    <c:forEach var="i" begin="1" end="10" step="1">
      <c:out value="{i}" />

      <br />
    </c:forEach>
  </body>
</html>
```

- Lo scopo principale di JSF è quello di fornire uno strumento per la realizzazione rapida di interfacce per l'interazione con i componenti server-side di Java.
- Modello di UI, basata su componenti, permette lo sviluppo di applicazioni in grado di adattarsi automaticamente alla tipologia di client.
- Lo sviluppatore utilizza questi moduli (questi possono essere anche complessi – es. DataTable che rappresenta i dati di una tabella da db) per costruire in poco tempo interfacce grafiche per il Web.
- Libreria standard JSF; gli utenti possono anche creare e definire le loro UI Components, così da massimizzare la riusabilità e la personalizzazione dei componenti a seconda delle esigenze.

- Una servlet chiamata “**Faces Controller**” si occupa della gestione delle richieste dirette all'applicazione JSF, implementando di fatto il pattern MVC.



```
<%@ taglib uri="http://java.sun.com/jsp/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsp/core" prefix="f" %>
<HTML>
<HEAD>
  <title>Hello</title>
</HEAD>
<body bgcolor="white">
<h2>Enter number</h2>
<f:view>
  <h:form >
    <h:graphicImage id="logo" url="/logo.gif" />
    <h:inputText id="num" required="true" size="4">
      <f:validateLength maximum="4" minimum="1" />
    </h:inputText>
    <h:commandButton value="submit" action="success" title="Submit" />
  </h:form>
</f:view>
</body>
</HTML>
```

- Notare come non vi sia un attributo ACTION; ciò si deve al fatto che JSF ha il proprio sistema di navigazione. La destinazione dove la form viene inviata è definita dal bottone attraverso il file faces-config.xml; ciò è simile a quanto accade in Struts...

- JDBC è l'API standard per l'interconnessione di un'applicazione scritta nel linguaggio Java e la maggior parte dei DBMS attualmente disponibili.

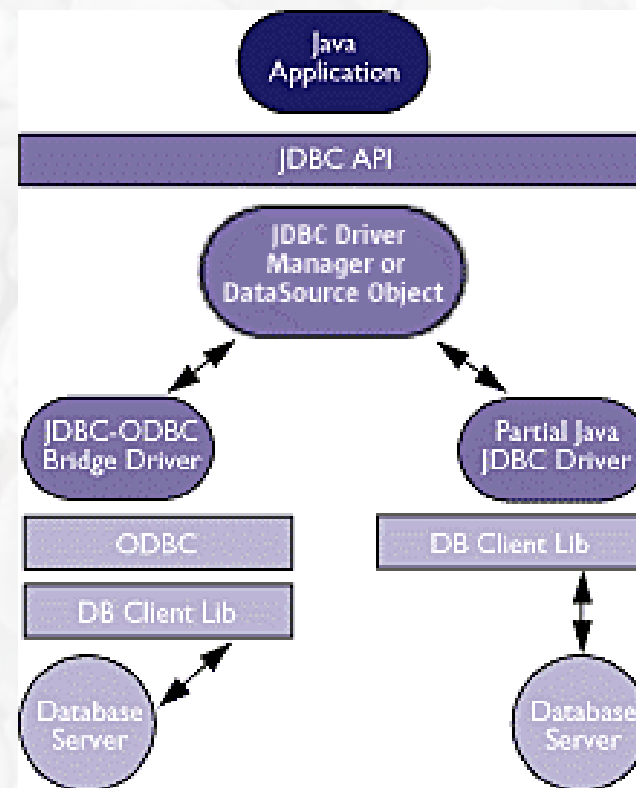
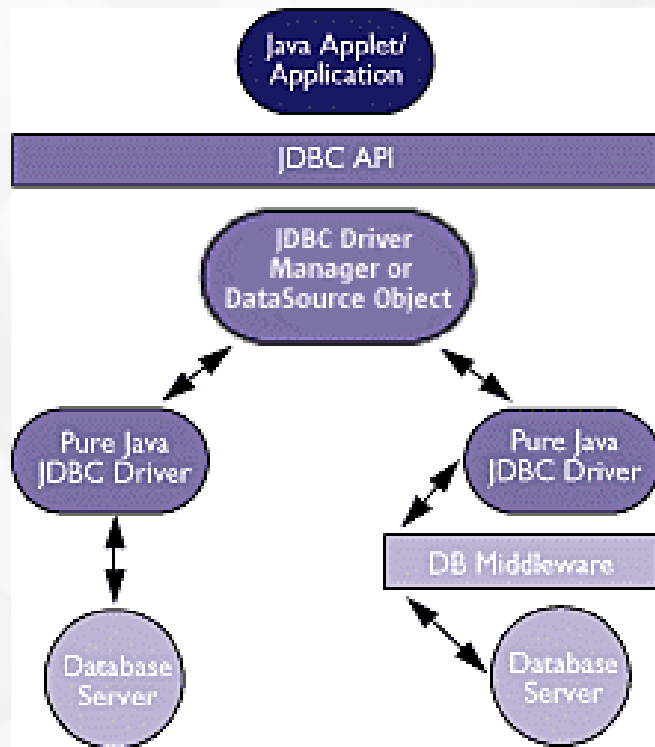
- Per mezzo di JDBC è sostanzialmente possibile:
 - Stabilire una connessione con un database
 - Eseguire Query SQL
 - Gestire i Risultati di una Query

● **Breve Esempio (Interazione con Oracle):**

```

Class.forName("oracle.jdbc.driver.OracleDriver")
Connection con = DriverManager.getConnection(
"jdbc:oracle:thin:@dbaprod1:1544:SHR1_PRD", username, passwd);
Statement stmt = con.createStatement() ;
String sqlString = "SELECT * FROM prova";
ResultSet rs = stmt.executeUpdate(sqlString);
while ( rs.next() ) { testo = rs.getString("testo"); System.out.println(testo); }
  
```

Struttura Connessioni JDBC



- Introduzione a J2EE
- J2EE Web Components
- **Il Framework Struts**
- Appendice Tecnica
- Bibliografia

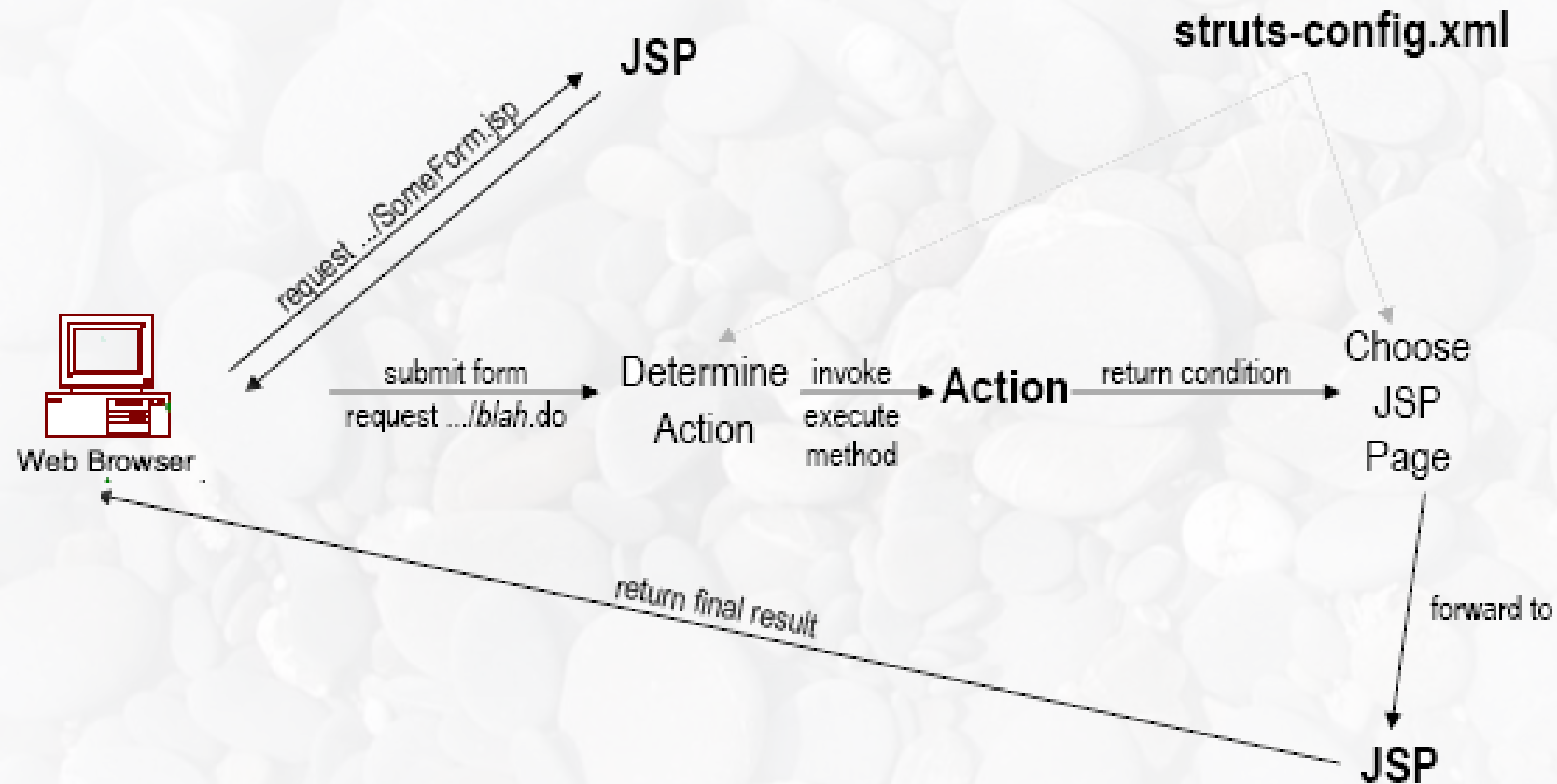
- **Un *Framework* che implementa il pattern MVC**
 - Struts offre un framework per lo sviluppo di applicazioni basate su Servlet, Beans e JSP (o altre tecnologie di presentazione), combinate così da implementare un pattern MVC.

- **Secondariamente...**
 - **Una collezione di *Utilities***

Struts offre classi che gestiscono la maggior parte dei task comuni nello sviluppo di applicazioni web
 - **Un Set di *Custom Tag Libraries* JSP**

Struts offre librerie di custom tag per visualizzare proprietà di beans, generare form HTML, iterare su vari tipi di strutture dati e gestire condizioni (es. if).

Parti in gioco e Interazioni tra loro



Dalla richiesta alla risposta

- **L'utente richiede una form**
- **Il sistema presenta la form**
- **La form fa una richiesta ad un URL (con estensione .do)**
 - Come vedremo tra poco, questo URL è mappato direttamente ad un Action class nel file struts-config.xml
- **Il metodo execute dell'Action è invocato**
 - Uno degli argomenti per l'esecuzione è un Form Bean che viene automaticamente creato, con le proprietà popolate sulla base dei dati immessi nella form.
 - L'oggetto Action invoca quindi la “business logic” ed effettua l'accesso alle basi di dati, disponendo i risultati di queste operazioni in beans.
 - Action usa quindi mapping.findForward per ritornare una condizione, la quale sarà a sua volta mappata nel file struts-config.xml ad una pagina JSP.
- **Struts inoltra la richiesta alla pagina JSP appropriata**
 - La pagina può usare i tags preposti all'interazione con i beans per mostrare i risultati (es. bean:write, bean:message)

I Passi di Costruzione di un'applicazione Struts

- **1] Creare il file struts-config.xml**
- **2] Creare un Form Bean**
- **3] Creare un Results Bean**
- **4] Definire una Action class per gestire le richieste**
- **5] Creare una form la cui action è definita in struts-config.xml**
- **6] Mostrare i risultati in una pagina JSP**

1] Creare il File struts-config.xml

- Mappare le richieste per indirizzi .do in Action classes
- Mappare le risposte in pagine JSP
- Dichiarare i form beans che verranno usati
- Eventualmente specificare le fonti per dati attinti dall'esterno (Database)
- Eventualmente specificare le risorse testuali da utilizzarsi per l'internazionalizzazione o comunque per mantenere in un unico luogo le componenti testuali di un'applicazione (buona cosa... si pensi ai vantaggi della centralizzazione quando vi è la necessità di sostituire un testo in più pagine)
- ***NB: Ricordarsi di far ripartire il web container dopo aver fatto ciò, dato che il file viene letto solo al primo avvio del server***

Esempio di semplice struts-config.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
    "http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
<!-- This is the struts configuration file for the Login example application -->
<struts-config>

    <data-sources>
        <!-- == Definizione delle Fonti di Dati (es. DB) == -->
        [...]
    </data-sources>

    <form-beans>
        <!-- Definizione dei Form Bean -->
        <form-bean name="loginForm" type="pao.struts.LoginForm"/>
    </form-beans>

    <action-mappings>
        <!-- Mappings per le Azioni -->
        <action path="/login"
            type="pao.struts.LoginAction"
            name="loginForm"
                scope="request"
            input="/login.jsp"
            validate="true">
            <forward name="failure" path="/login.jsp"/>
            <forward name="success" path="/loggedIn.jsp"/>
        </action>
    </action-mappings>
</struts-config>
```

Mapping per i .do con Action Classes

- Per esempio, associamo a `register.do` la Action Class `RegisterAction1`; aggiungiamo una action entry nelle `action-mappings`, con i seguenti attributi:
 - **path**: il path relativo che dev'essere mappato con l'Action senza l'estensione `.do` (implicita). Nel nostro esempio, `path="/register1"` si riferisce a `http://hostname/webAppName/register1.do`.
 - **type**: il nome dell'Action Class che dev'essere invocata quando il path della richiesta è quello indicato dal parametro `path`.
 - **name**: un bean il cui nome è quello della dichiarazione del `form-bean`.
 - **scope**: [`request` | `session`]. Indica lo scope del Form Bean eventualmente creato. `Session` è il default.

```
● <action-mappings>  
  <action path="/register1"  
          type="coreservlets.RegisterAction1"  
          name="userFormBean"  
          scope="Request">  
  </action>  
  [...]  
</action-mappings>
```

Mapping alle pagine JSP per i Risultati

- In questo esempio usiamo la pagina `confirm.jsp` quando `RegisterAction1` ritorna "success", la pagina `error.jsp` quando `RegisterAction1` ritorna "failure":

- ```
<action-mappings>
[...]
<forward name="success" path="/WEB-INF/results/confirm.jsp"/>
<forward name="failure" path="/WEB-INF/results/failure.jsp"/>
</action-mappings>
```

- Nel caso uno stesso forward sia condiviso da più Actions (o anche usato come semplice link), è possibile includere il forward tra i `global-forwards` (`before action-mappings`) al posto dell'azione.

- ```
<global-forwards>
  <forward name="bad-address"
    path="/WEB-INF/results/bad-address.jsp"/>
  <forward name="bad-password"
    path="/WEB-INF/results/bad-password.jsp"/>
  <forward name="success"
    path="/WEB-INF/results/confirm.jsp"/>
</global-forwards>
```

- Ed ecco un esempio di link in una pagina JSP:

```
<html:link forward="logoff">Logoff</html:link>
```

2] Creare un Form Bean

- Un Form Bean è un Java Bean che viene automaticamente riempito sulla base dei parametri immessi in una form e viene poi passato al metodo execute.

- Esempio:**

```
package struts.pao;
import org.apache.struts.action.*;
public class UserFormBean extends ActionForm {
private String email = "Missing address";
private String password = "Missing password";
public String getEmail() { return(email); }
public void setEmail(String email) {
this.email = email; }
public String getPassword() { return(password); }
public void setPassword(String password) {
this.password = password; }
}
```

3] Creare un Return Bean

- Servono a trasferire i risultati generati dalla business-logic alle JSP usate per la presentazione.
- Devono avere getter and setter, così come tutti i JavaBeans; non hanno altri legami con altre classi di Struts o con il file struts-config.xml (non devono essere dichiarati).
- Questi sono resi disponibili nell'ambito (scope) della richiesta, sessione o applicazione, per mezzo del metodo setAttribute con HttpServletRequest, HttpSession o ServletContext rispettivamente.

Esempio di Return Bean

```
package struts.pao;

public class SuggestionBean {
    private String email;
    private String password;

    public SuggestionBean(String email, String password) {
        this.email = email;
        this.password = password;
    }

    public String getEmail() {
        return(email);
    }

    public String getPassword() {
        return(password);
    }
}
```

Business Logic che crea un Return Bean

```
package struts.pao;

public class SuggestionUtils {

    private static String[] suggestedAddresses = {
        "president@whitehouse.gov",
        "ellison@oracle.com"
    };

    private static String chars =
        "abcdefghijklmnopqrstuvwxyz0123456789#@$%^&*?!";

    public static SuggestionBean getSuggestionBean() {
        String address = randomString(suggestedAddresses);
        String password = randomString(chars, 8);
        return(new SuggestionBean(address, password));
    }

    [...]

}
```

4] Definire una Action Class per gestire le richieste

- Il file `struts-config.xml` individua le classi Action che si prendono carico di gestire le richieste per gli URL dell'applicazione.
- Gli oggetti Action sono quelli che invocano i metodi opportuni nella business-logic, permettono l'accesso ai dati, memorizzano i risultati in beans e gestiscono la parte rimanente del flusso dei dati dell'applicazione Struts, nonché il da farsi nel caso di potenziali eccezioni (a livello concettuale, non strettamente di codice).
- Il file `struts-config.xml` deciderà la pagina JSP da mostrare all'utente a seconda di ciò che è stato deciso dall'Action.

Metodo execute di una Action Class

```
public ActionForward execute(ActionMapping mapping,
ActionForm form,... request, ... response)
throws Exception {
    UserFormBean userBean = (UserFormBean)form;
    String email = userBean.getEmail();
    String password = userBean.getPassword();
    if ((email == null) || (email.trim().length() < 3) || (email.indexOf("@") == -1)) {
        request.setAttribute("suggestionBean",
        SuggestionUtils.getSuggestionBean());
        return(mapping.findForward("bad-address"));
    } else if ((password == null) || (password.trim().length() < 6)) {
        request.setAttribute("suggestionBean",
SuggestionUtils.getSuggestionBean());
        return(mapping.findForward("bad-password"));
    } else {
        return(mapping.findForward("success"));
    }
}
```

5] Creare una form la cui action è definita in struts-config.xml

Documento HTML:

```
<!DOCTYPE ...>
<HTML>
<HEAD><TITLE>Registrazione Nuovo Account</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<CENTER>
<H1>Registrazione Nuovo Account</H1>
<FORM ACTION="formAction.do" METHOD="POST">
Email address: <INPUT TYPE="TEXT" NAME="email"><BR>
Password: <INPUT TYPE="PASSWORD" NAME="password"><BR>
<INPUT TYPE="SUBMIT" VALUE="Procedi">
</FORM>
</CENTER>
</BODY></HTML>
```

6] Mostrare i Risultati in una pagina JSP

- ❖ Dato che Struts è costruito sul pattern MVC, questa pagina JSP dovrebbe evitare la comparsa di elementi di scripting
 - Nella maggior parte dei casi, le richieste di pagine JSP hanno origine da un'Action; le pagine possono quindi essere collocate anche in WEB-INF; se si usano pagine JSP indipendenti da un'Action, allora queste devono essere incluse in una sottodirectory dell'applicazione; le entry nel file struts-config.xml devono indicare in questo caso `<forward ... redirect="true"/>`.
- ❖ Diverse alternative per mostrare l'output:
 - Usare **JSP scripting elements**: approccio da **NON usare** ...è infatti proprio quello che Struts vorrebbe evitare!).
 - Usare **jsp:useBean** e **jsp:getProperty**: approccio possibile, ma i tags sono parecchio fatiscenti,
 - Usare il **tag JSTL c:out** non una cattiva idea... ma è forse eccessivo usare JSTL per alcune situazioni (ad esempio solo per mostrare il contenuto di un bean); è invece consigliabile se si usa JSTL anche altrove nell'applicazione.
 - Usare l'**expression language** di JSP 2.0: forse la migliore opzione, ma bisogna accertarsi che il server la supporti.

Usare il tag `bean:write` di Struts

- Questo è l'approccio decisamente più comune con Struts
 - automaticamente **`bean:write`** filtra i caratteri speciali HTML, sostituendo `<` con `<` e `>` con `>`; per disabilitare ciò, si può usare `filter="false"`, come nell'esempio)

```
<!DOCTYPE ...>
<HTML>
<HEAD><TITLE>Illegal Email Address</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<CENTER>
<H1>Illegal Email Address</H1>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
The address
"<bean:write name="userFormBean" property="email"/>"
is not of the form username@hostname (e.g.,
<bean:write name="suggestionBean" property="email" filter=false/>).
<P>
Please <A HREF="register1.jsp">try again</A>.
</CENTER>
</BODY></HTML>
```

■ **struts-config.xml:**

```
<data-sources>
  <data-source key="paodb"
    type="org.apache.commons.dbcp.BasicDataSource">
    <set-property property="description" value="PAO Test Application"/>
    <set-property property="driverClassName"
      value="org.postgresql.Driver"/>
    <set-property property="username" value="steve"/>
    <set-property property="password" value="*****"/>
    <set-property property="url"
      value="jdbc:postgresql://192.168.0.3:5432/paodb"/>
  </data-source>
</data-sources>
```

■ **Generica Action Class:**

```
LoginBean lb = new LoginBean(getDataSource(request, "paodb"));
// Qui viene creato automaticamente un bean
```

- Estratto da struts-config.xml:

- `<message-resources null="false" parameter="ApplicationResources"/>`

- Estratto da ApplicationResources.properties:

- `#login page text`

- `login.title=PAO Test Application - Prego Autenticarsi`

- `login.message=Prego Eseguire il Log-In`

- `login.username=username:`

- `login.password=password:`

- `login.button.signon=Log In`

- Generica Pagina JSP:

- `<H1><fmt:message key="login.message"/></H1>`

- Per creare una versione diversa dell'applicazione, in un'altra lingua, basta creare un file denominato `ApplicationResources_XX.properties`, dove `XX` è un codice lingua ISO (es. de, it, es...)

● Separazione della Logica dai Contenuti

- Struts spinge ad utilizzare il pattern MVC nella propria applicazione in maniera piuttosto decisa, permettendo così la separazione della logica dalla presentazione.

● Configurazione Centralizzata basata su file

- Anzichè cablare i parametri di configurazione di un'applicazione in file Java, molte variabili usate dalle applicazioni Struts sono rappresentate in file XML o property files. Ciò significa che cambiamenti possono essere apportati in un unico file, anche se si tratta di informazioni presentate poi su diverse pagine JSP.
- Affinchè queste abbiano effetto non è necessario ricompilare l'applicazione.
- Questo approccio (così come la precedente caratteristica vista) facilita il lavoro in team, anche se portato avanti da persone con competenze diverse (es. Il programmatore lavora sui files Java, l'esperto di comunicazione modifica il file con i testi da presentare sul sito).

● Gestione dei Forms

- Struts è in grado di validare l'input da form e inviare automaticamente un messaggio di errore all'utente, qualora questo sia non presente o in un formato non previsto dall'applicazione
 - Questa validazione può avvenire sul server (Java) o congiuntamente su Server e Client (JavaScript).

● Tag HTML

- Apache Struts offre custom tags per creare form HTML associate con JavaBeans. Ciò ha due vantaggi:
 - Lascia ricevere valori dei campi di un form direttamente da oggetti Java.
 - Lascia mostrare nuovamente form con valori già immessi intatti in modo automatico.

Documentazione non sempre chiara

- Nonostante negli ultimi anni le cose siano migliorate, Struts ha ancora meno fonti di apprendimento, sia online, sia in termini di libri, rispetto ad esempio a JSP e Servlet. La documentazione fornita sul sito ufficiale (Apache) potrebbe essere organizzata decisamente meglio e non è sempre facile da comprendere per uno che si avvicina per la prima volta a questo framework.

Meno Trasparente

- Le applicazioni basate su struts hanno un flusso dei dati più contorto rispetto alle comuni applicazioni basate su Java. La conseguenza (negativa di ciò) è che queste saranno:
 - Più difficili da comprendere
 - Più difficili da testare e (per alcuni aspetti) ottimizzare

Learning Curve più accentuata

- Per utilizzare il pattern MVC con il RequestDispatcher standard è sufficiente masticare un po di JSP e conoscere le API delle Servlet. Per implementare il pattern MVC con Struts è necessario, oltre a queste, **conoscere un framework elaborato ed esteso**. Le conseguenze di ciò sono particolarmente evidenti nei sistemi di dimensioni ridotte, con deadlines incombenti nell'immediato e senza sviluppatori esperti a disposizione: nel tempo in cui questi imparano ad usare Struts, sarebbe possibile creare l'intera applicazione.

Dove viene usato?

The image displays a screenshot of a web browser window showing the HP.com website. The browser's address bar shows the URL <http://www.shopping.hp.com/webapp/shopping/home.do>. The HP.com website features a navigation menu with links for Home, Desktops, Notebooks, Handhelds, Printers, and Ink, Toner & Paper. The main content area includes a "Home & Home Office Store" banner with a "SUPER TAG SALE" on HP Pavilion notebooks and Media Center PCs. A "spring break\$ captureprint&share" banner for digital cameras is also visible. The left sidebar contains sections for "Home & Home Office Store", "Shopping Cart", "Sign up for e-mail updates", "How to get...", "Shopping info", and "Product support".

Overlaid on the HP.com page is a smaller browser window showing an Air France flight search page. The search results indicate "There are no flights available. Please modify your travel dates." The flight search details include:

- Departing from: New York, John F. Kennedy (JFK) - USA
- Arriving at: France, Marco Polo (NCE) - ITALY
- Flight type: Round trip
- Departure date: Tuesday, April 2006
- Return date: Tuesday, April 2006

The Air France page also features a "24 minutes * 24 gift card PICKUP GUARANTEE" banner and a "Click & Learn" section with links to "Choosing a digital camera", "Choosing a camcorder", and "Storing digital pictures".

Alternative a JSP per la Presentazione

● Velocity

- Un **template engine** basato su Java, utile per la generazione di informazioni in formato testo di qualsiasi tipo (codice sorgente, HTML, XML, SQL, reports).

● FreeMaker

- Altro **template engine** in grado di creare qualsiasi output testuale (da HTML a PDF); i templates non contengono “application logic”.

● Cocoon

- Progetto di Web application Framework che offre, attraverso cosiddette **pipelines** meccanismi per la trasformazione dei formati di presentazione (tra i formati disponibili troviamo: JSP, Velocity, Freemaker e PHP).

● Struts CX - stxx

- Con questi due Framework il **JSP** è **sostituito da XML** che può poi essere facilmente trasformato virtualmente in qualsiasi altro formato attraverso XSL.

- Espresso: in realtà è un framework completo, che include Struts; offre però anche altre funzionalità (in particolare **autenticazione**, scheduling ed esecuzione in background, **testing**).
- Struts Workflow Extension: offre meccanismi che cercano di minimizzare le situazioni non previste, dovute a comportamenti anomali da parte degli utenti (es. doppio “submit” di una form, da una serie di azioni sequenziali previste)
- Easy Struts Plug-in: plugin per Eclipse e Jbuilder si compone di vari tools e wizards per la modifica dei file di configurazione delle applicazioni Struts.

- Tapestry: Framework Jakarta dal maggio 2003 è, come Struts, uno strumento pensato per siti Internet particolarmente grandi e complessi. Punta, grazie ad una buona flessibilità, ad adattarsi a markup languages anche diversi dall'HTML (es. Wml), oltre che separare la logica dalla presentazione. Le applicazioni sono basate su componenti riusabili e **templates, testabili anche a livello statico** (ovvero senza la necessità di aver già sviluppato le parti dinamiche del sito).
- WebWork: Altro framework, sviluppato per offrire supporto a template e componenti di UI (controlli per le form), internazionalizzazione, mapping dinamico di parametri di form a JavaBeans e validazione dell'input.

- Introduzione a J2EE
- J2EE Web Components
- Il Framework Struts
- **Appendice Tecnica**
- Bibliografia

Appendice – Installazione di Apache Tomcat

- Un buon tutorial per l'installazione del Web Container Apache Tomcat su Linux (Debian) si trova presso questo link:

http://www.howtoforge.com/apache2_tomcat5_mod_jk

- Se desiderate eseguire l'installazione su di una distribuzione derivata da RedHat (Fedora, CentOS), sono necessari i seguenti passi, prima di procedere e durante l'esecuzione delle istruzioni del link di cui sopra:
 - 1] per poter utilizzare yum per il recupero dei pacchetti e soddisfare le dipendenze, eseguire il comando:
rpm --import http://fedora.redhat.com/about/security/4F2A6FD2.txt
 - 2] dopodichè, utilizzare yum con la sintassi “***yum install NOME_PACCHETTO***” per installare i seguenti pacchetti: apr-util, aprapr-util-devel, apr-devel, automake, autoconf, m4, libtool-libs, libtool, pcre, pcre-devel.
 - 3] aggiungere nel file /etc/profile le seguenti due righe:
PATH=/usr/lib/jdk/bin:\$PATH
export PATH
 - 4] rispetto alle istruzioni riportate nel link di cui sopra, dove questi parla di apxs2, è sufficiente linkare il proprio apxs (di solito in /usr/sbin).
 - 5] il tutorial propone di installare Tomcat in /usr/lib/. Dopo averlo fatto consiglio un link a /etc/tomcat per i files di configurazione e a /var/logs/tomcat per i logs.

- Introduzione a J2EE
- J2EE Web Components
- Il Framework Struts
- Appendice Tecnica
- **Bibliografia**

- The J2EE 1.4 Tutorial – E. Armstrong, J. Ball, S. Bodoff, D.B. Carson, I. Evans, D. Green, K. Haase, E. Kendrock – Sun Microsystems – 2004
- Art of Java Web Development – N. Ford – Manning - 2004
- J2EE Developer's Handbook - P.J. Perrone, S. Venkata, R. “Krishna”, R. Chaganti, T. Schwenk – Sams – 2003
- J2EE Bible 1.4 - James McGovern, Rahim Adatia, Yakov Fain, Jason Gordon, Ethan Henry, Walter Hurst, Ashish Jain, Mark Little, Vaidyanathan Nagarajan, Harshad Oak, Lee Anne Phillips - Wiley – 2003
- IBM Websphere Online Documentation
- The Struts Framework – Practical Guide for Java Programmers – S. Spielman – Morgan Kaufmann Publishers - 2003
- Struts Kick Start – J. Turner, K. Bedell – Sams – 2002
- JavaServer Pages – H. Bergsten - O'Reilly - 2002

- Introduction to the J2EE 5:
http://java.sun.com/developer/technicalArticles/J2EE/intro_ee5/
- M. Hall Tutorials: <http://www.coreservlets.com/>
- Apache Struts Project: <http://struts.apache.org/>
- Apache Tomcat: <http://tomcat.apache.org/>
- Velocity: <http://jakarta.apache.org/velocity/>
- Jboss Application Server:
<http://www.jboss.org/products/jbossas>
- End-To-End Internationalization of Web Applications:
<http://www.javaworld.com/javaworld/jw-05-2004/jw-0524-i18n.html>
- Macromedia - EJB Overview:
http://livedocs.macromedia.com/jrun/4/Getting_Started_with_JRun/introejb2.htm
- Introduction to EJB:
http://java.about.com/library/weekly/aa_ejbintro1_1.htm