Java server pages
and model view controller architecture

Knower, Known, and Process of Knowing

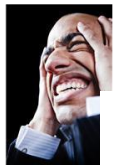
Main point 1 Preview

The web container generates a servlet from a JSP file the first time the JSP is requested from a web application. Since a JSP is essentially a servlet, one should understand servlets to effectively deal with JSPs.

Science of Consciousness: Actions in accord with fundamental levels of knowledge promote success in dealing with more expressed values. The most fundamental level of knowledge is pure knowledge, pure consciousness.

Main idea of JSP

- separate display from processing, i.e., separate html from java
  - servlet is java with some html mixed in
    - a little plain java code
    - and lots of enclosed HTML
      - out.println(“ … \” … \” … “)
    - writing and maintaining quickly becomes a headache



  - Jsp is html with a little java mixed in
    - a little java code bracketed in
      - <% …. %> , etc
    - and lots of plain HTML

- There are two types of data in a JSP page:
  - Template Data: The static part, copied directly to response
    - static HTML
  - JSP Elements: The dynamic part, translated and executed by the container
    - typically generate additional HTML portions of the page
    - can execute arbitrary Java code

Four types of JSP elements:

- **Script**: embedded Java statements <% %> <%= %> …

- **Directive**: page level operations for the translation phase.

<%@ page %>

- **EL Expression**: more convenient and powerful than a JSP expression ${ } vs <%= %>

- **Action**: JSP “functions” that encapsulate some commonly used functionality <c:foreach />

JSP scripting elements:

- **Declaration**:
  - <%! Inserts instance variable and method declarations into servlet p292
  - <%! Java declaration statements %>
  - <%! int count = 0; %>
- **Scriptlet**:
  - Inserts Java statements inside service method
    - <% Java statements %>
    - <% count = count * 10; %>
    - <% inserts into the service method p282
- **Expression**:
  - expects a Java expression, which it puts inside ‘out.print’ in service method
  - <%= Java expression %>
  - wraps it inside a print statement p286
  - <%= ++count %> becomes … out.print(++count); … in service method
- **Comment**:
  - <%-- jsp comment --%> p302
  - contrast with HTML comment
    - <!– Comment -->
    - HTML comments get sent to the browser (part of the HTML)
    - JSP elements all processed by container and do not appear in generated HTML

JSP Demo

- Inspect the JSP demo code
- Implement and run
- Insert the scripting elements shown after step 6
- Find the generated servlet
- Find the inserted JSP statements
- Find the implicit objects

```
 public void _jspService(HttpServletRequest request, HttpServletResponse response)   throws java.io.IOException, ServletException {
   PageContext pageContext = null;
   HttpSession session = null;
   ServletContext application = null;
   ServletConfig config = null;
   JspWriter out = null;
   Object page = this;
   JspWriter _jspx_out = null;
   PageContext _jspx_page_context = null;
   try {//SNIP
     application = pageContext.getServletContext();
     config = pageContext.getServletConfig();
     session = pageContext.getSession();
     out = pageContext.getOut();//SNIP
     out.write("       <title> Introduction to JSP demo – postback page </title>\n");
out.write("    </head>\n");
out.write("    <body>\n");
out.write("     <p>This is the postback message</p>\n");
     out.write("      ");
out.write("     The count is now:  \n");
out.print( ++count );
out.write("     ");
out.write("     <!-- This is an html comment inserted by the increment comment -->\n");
     out.write("      ");
 count = count * 10;
```

JSP Demo

- Inspect the JSP demo code
- Implement and run
- Insert the scripting elements shown after step 6
- Find the generated servlet
  - %userprofile%\.IntelliJIdeayyyy.m\system\tomcat\
- Find the inserted JSP statements
- Find the implicit objects

JSP Predefined Variables = *Implicit Objects*

Objects	Description
request	the HTTPServletRequest object associated with the request
response	the HTTPServletRespnse object associated with the response
out	the JspWriter used to send output to browser
session	the HttpSession object associated with the request
application	the ServletContext obtained via getServletConfig()
config	the ServletConfig object
pageContext	the PageContext object to get values from and store attributes into any of the other contexts (request, session, servletContext)
page	a synonym for this – the “this” of the jsp page’s generated servlet; page has its own scope (elements of the page)
exception	The Exception object allows the exception data to be accessed by designated JSP.

JSP Lifecycle

LOADING PHASE

1. JSP Page Translation

A java servlet file is generated from the JSP source file. The container validates the syntactic correctness of the JSP pages and tag files. The container interprets the standard directives and actions, and generates java code for a servlet.

2. JSP Page Compilation

The generated java servlet file is compiled into a java servlet class.

3. Class Loading

The java servlet class that was compiled from the JSP source is loaded into the container.

EXECUTION PHASE

1. Initialization

jspInit() method is called immediately after the instance is created. It is called only once during JSP life cycle.

2. _jspService()

This method is called for every request of this JSP during its life cycle.

3. jspDestroy()

The servlet completes its purpose and submits itself to servlet heaven (garbage collection).

jspInit(), _jspService() and jspDestroy() are the life cycle methods of JSP.

http://javapapers.com/jsp/jsp-life-cycle-explain/

http://www.tutorialspoint.com/jsp/jsp_life_cycle.htm

JSP Directives

message from a JSP page to the JSP container that controls processing of entire page.

```
<%@ page import="java.util.Date"%>

<% System.out.println( "Evaluating date now" );  Date date = new Date(); %>
Hello! The time is now <%= date %>

<%@ include file="hello.jsp"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

Main point 1

The web container generates a servlet from a JSP file the first time the JSP is requested from a web application. Since a JSP is essentially a servlet, one should understand servlets to effectively deal with JSPs.

Science of Consciousness: Actions in accord with fundamental levels of knowledge promote success in dealing with more expressed values. The most fundamental level of knowledge is pure knowledge, pure consciousness.

Main point 2 Preview

An EL expression is a compact expression of a systematic evaluation of the page, request, session and application scopes.

Science of Consciousness: The laws of nature are compact expressions that control the infinite diversity of the manifest universe. Our actions are more in accord with the laws of nature when we experience their basis in pure consciousness.

Why You Shouldn't Use Java in a JSP

- You can do almost everything in JSP that you can do using Java.
  - But that does not mean you should do it.
- JSP is a technology that was designed for the *presentation layer*, also known as the view.
  - In most organizations, user interface developers are responsible for creating presentation layer.
    - These developers rarely have experience writing Java code, and providing them with the ability can be dangerous.
- In a well-structured, cleanly coded application, the presentation layer is separated from the business logic, which is likewise separated from the data persistence layer.
- It's actually possible to create JSPs that display dynamic content without single line of Java inside the JSP. That's our mission!!!

Forwarding a Request from a Servlet to a JSP

- A typical pattern when combining Servlets and JSPs is to have the Servlet accept the request, do any business logic processing and data storage or retrieval necessary, prepare a model that can easily be used in JSP, and then forward request to the JSP.

```
request.getRequestDispatcher("/WEB-INF/jsp/welcome.jsp").forward(request, response);
```

EL (Expression Language)

- Java code inside of JSP is not a good idea, discouraged.
- easier way to display data and perform simple operations without Java
  - easily read,
  - familiar to both Java developers and UI developers,
  - simple set of rules and operators
- Expression Language (EL) developed to support rendering of data on JSP pages without scriptlets, declarations, or expressions.
  - When the JSP compiler sees `${expr}`, it generates code to evaluate the expression and inserts the value of expression at that spot on the JSP page

EL (Expression Language)

- recall JSP “expression”
  - <%= myMovieList.get(i) %>
  - evaluates the expression and writes it out to the HTML page at its location on the page
  - <%= ((Person) request.getAttribute(“pers”)).getDog().getName() %>
  - assumes request.setAttribute("pers", aPerson) in request scope;

- recall that attributes are where web app stores model values
  - values computed in model and then accessed in page for display
    - no model code on the JSP page, just value insertion

- EL simplifies JSP expression syntax
  - ${pers.dog.name}

JSP Expression vs EL

<%=((Person) request.getAttribute("pers")).getDog().getName()%>

With EL it becomes:

${pers.dog.name}

- See w3d3-expression demo
  - Why cast not needed for <%= p.getDog().getName()%> ?

High level description of EL

${something}

- container evaluates this as follows

- checks page scope for an attribute named "something",
  - if found use it.
- otherwise check request scope for an attribute named "something",
  - if found use it.
- otherwise check session scope for an attribute named "something",
  - if found use it
- otherwise check application scope for an attribute named "something",
  - if found use it.
- otherwise ignore the expression.

More detailed description

${firstThing}

- if firstThing is not an implicit EL object search page, request, session and application scopes until attribute "firstThing" is found

  ${firstThing.secondThing}
  - if firstThing is an attributed that returns a bean then secondThing is a property of the bean
  - if firstThing is an attributed that returns map then secondThing is a key of the map

  ${firstThing[secondThing]}
  - if firstThing is a bean then secondThing is a property of the bean
  - if firstThing is a map then secondThing is a key of the map
  - if firstThing is a List then secondThing is an index into the List

Example of EL Bracket notation

- in Servlet:
  - `String[] fruit= ["Banana", "Orange", "Apple"];`
  - `request.setAttribute("myFruitList", fruit);`

- in JSP:
  - `${myFruitList[0]}` // Banana
  - Note use of myFruitList[0] versus fruit[0]

EL is “null friendly”

- scripting languages try to fail without glaring error messages
  - helpful when used by end users
  - not so good for developers, need to be aware
  - Important for careful testing and checks
- if EL cannot find a value for the attribute it ignores it
  - caution
  - no warning or error message
- in arithmetic, treats null value as 0
  - could be a surprise
  - ${100 + myBankAccount.balance} → $100?? !!
    - “What happened to the bank account?”
- in logical expressions, nulls become “false”
  - more surprises ??

Main point 2

An EL expression is a compact expression of a systematic evaluation of the page, request, session and application scopes.

Science of Consciousness: The laws of nature are compact expressions that control the infinite diversity of the material universe. Our actions are more in accord with the laws of nature when we experience their basis in pure consciousness.

Main point 3 Preview

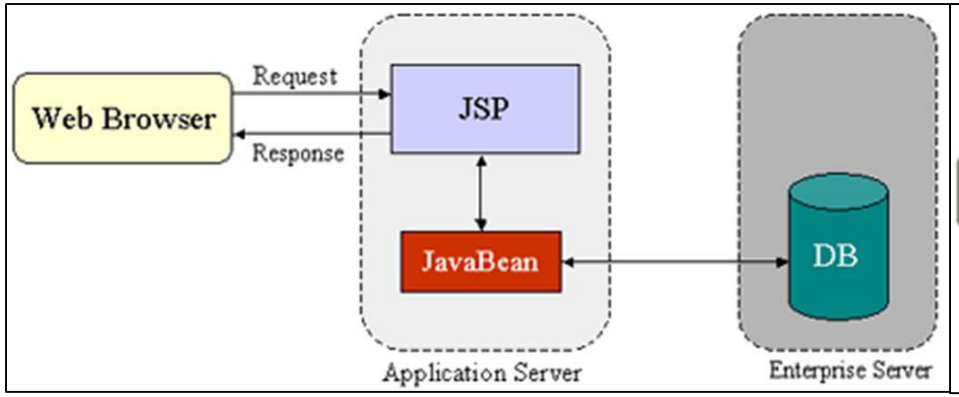
When you use JSP pages according to a Model 2 architecture, there is a servlet that acts as a controller (process of knowing) that sets attribute values based on computations and results from a business model (knower), then dispatches the request to the servlet generated by the JSP page (known). The JSP retrieves the attribute values and inserts them into the designated places in the HTML being sent to the browser.
Science of

.

Consciousness: Complete knowledge is the wholeness of knower, known, and process of knowing.

JSP Model 1 and Model 2 Architectures

Model 1



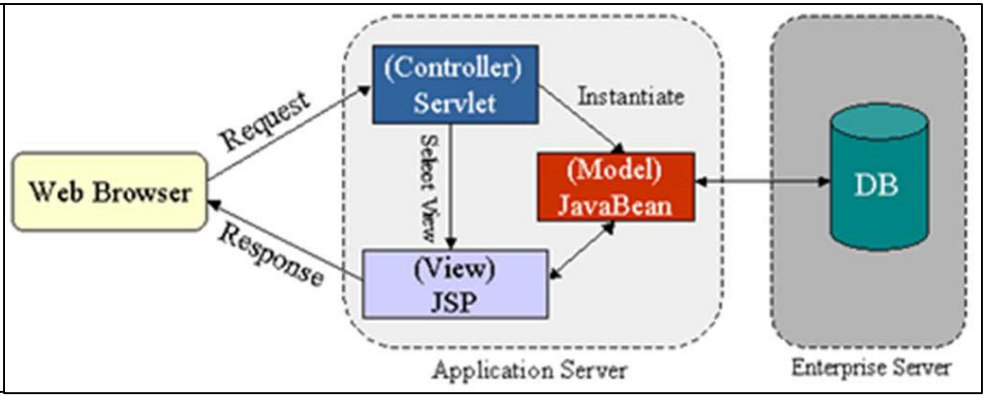
- Simple architecture
- For small applications
- Issues

  - Pages are coupled, need to know about each other.

  - Expensive to provide another presentation for same application.

  - Java code in HTML

Model 2



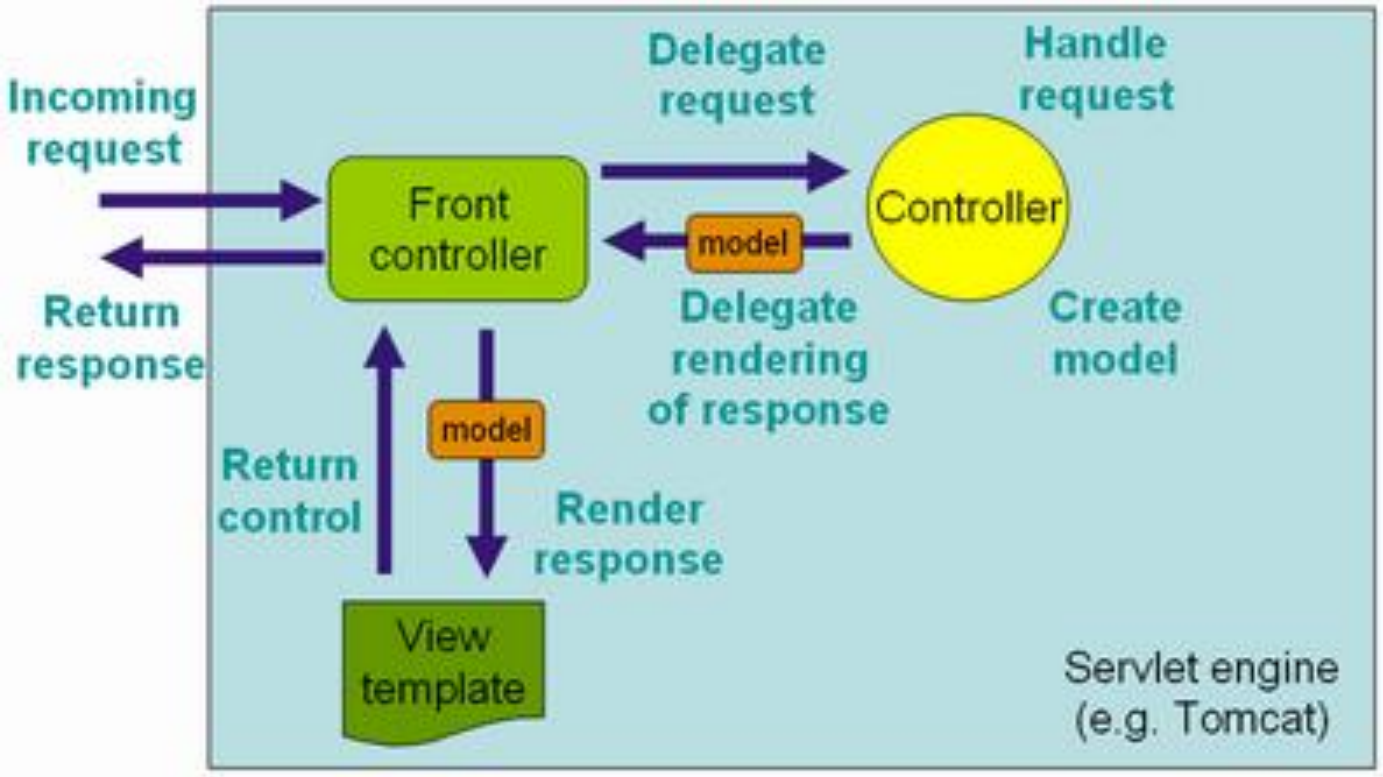
- More complex architecture
- For medium and large applications
- Advantages

  - Each team can work on different pages. Easy to understand each page.

  - Separation of presentation from control, making it easy to change one without affecting the other.

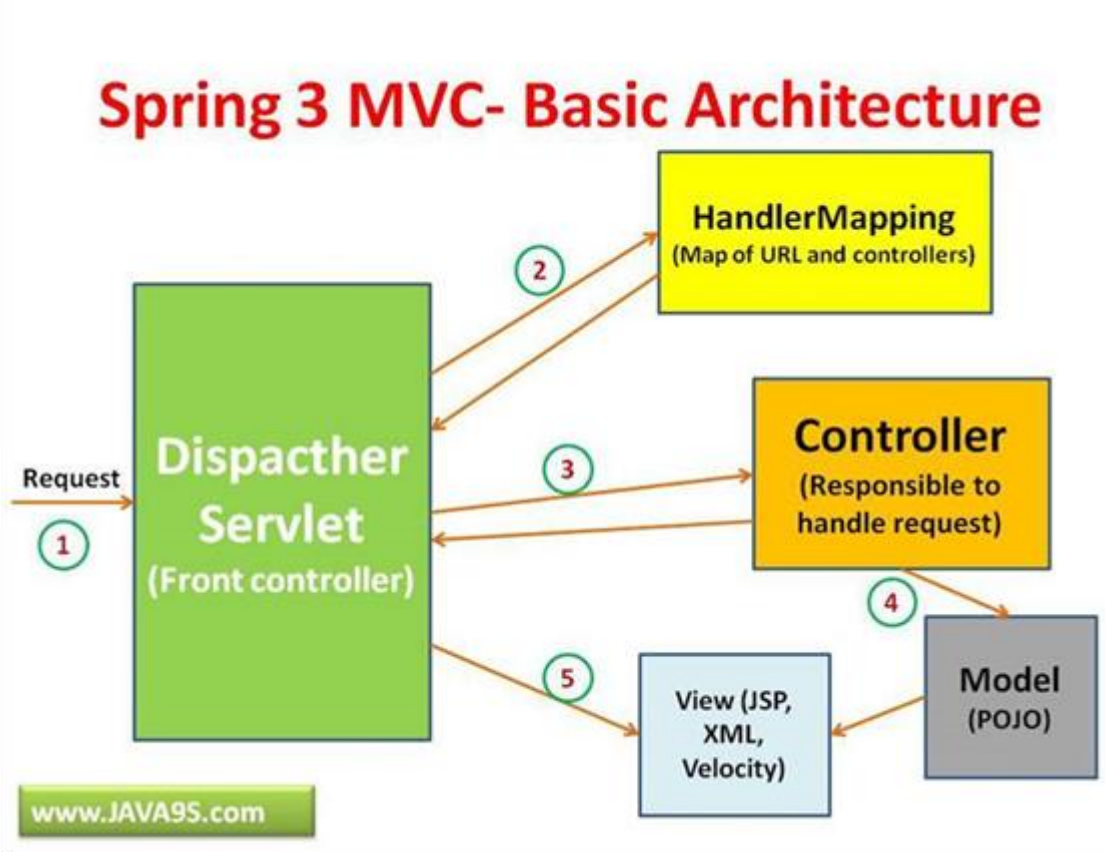
  - no Java code in HTML

SpringMVC Architecture



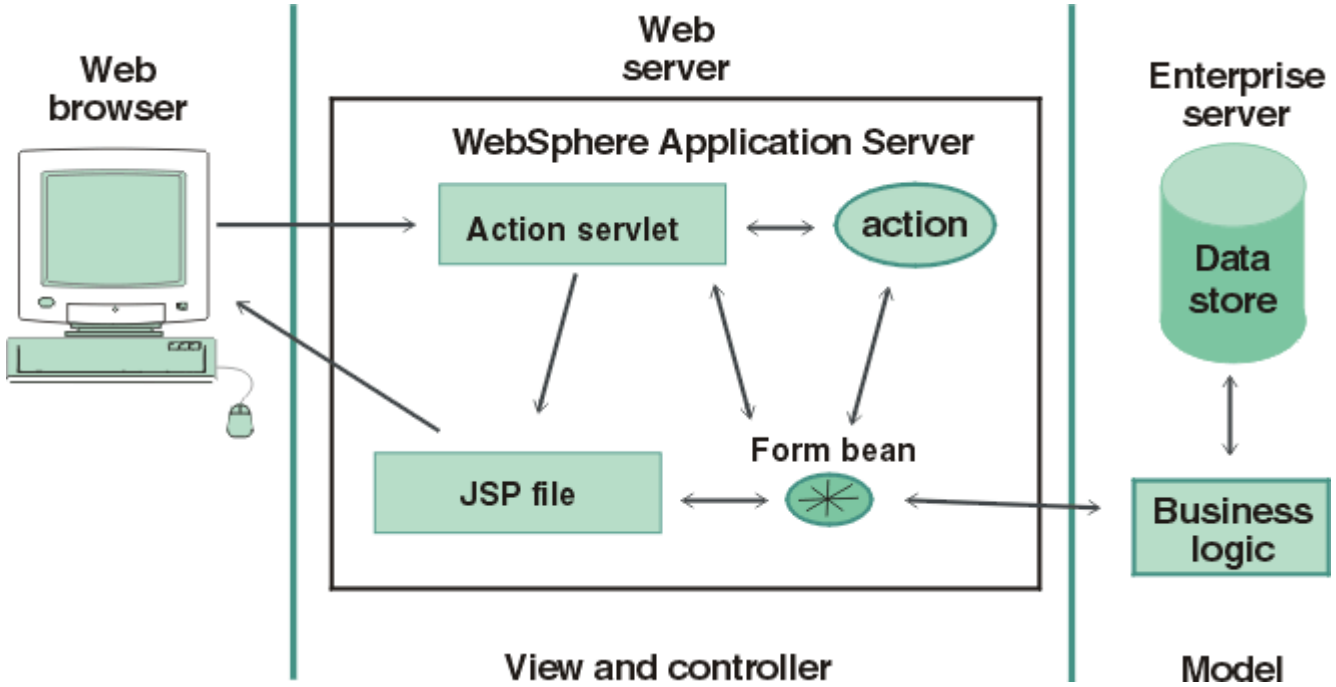
- http://stackoverflow.com/questions/20314098/spring-mvc-without-servlets
- SpringMVC site

SpringMVC Architecture



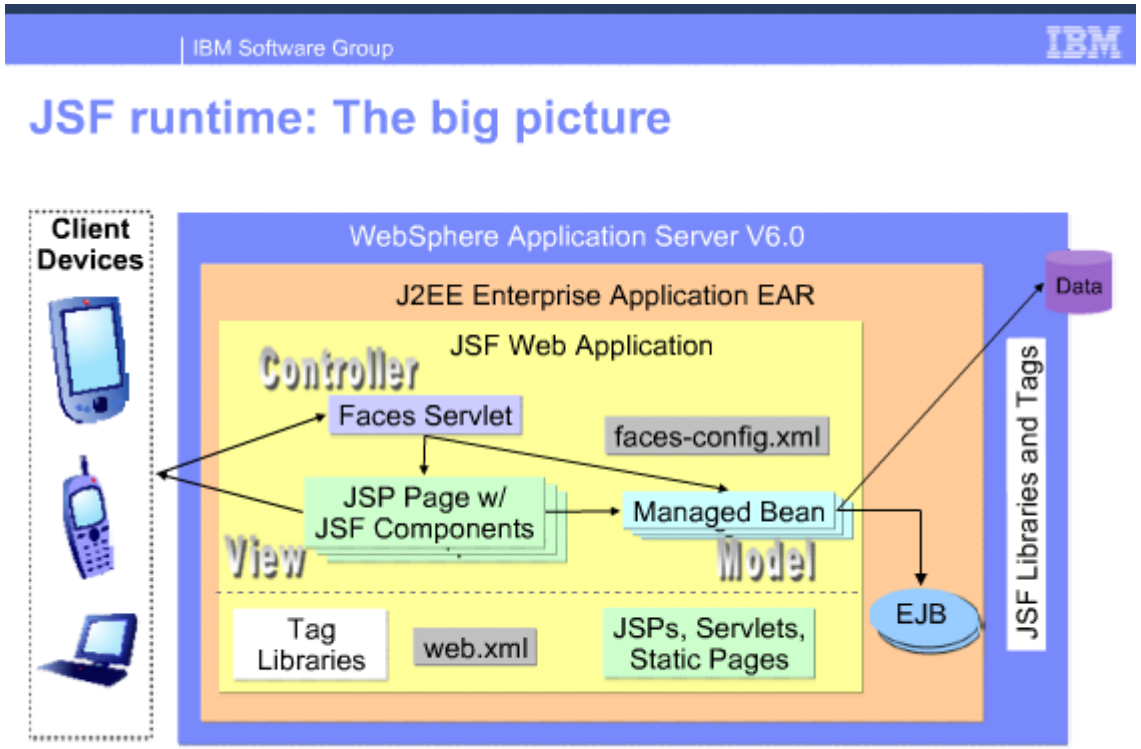
- http://java9s.com/spring-3-mvc/spring-3-mvc-introduction-spring-3-mvc-architecture

Struts MVC Architecture



- http://www.ibm.com/support/knowledgecenter/SSRTLW_8.0.4/com.ibm.etools.struts.doc/topics/cstrdoc001.html

JSF MVC Architecture



http://stackoverflow.com/questions/5104094/what-components-are-mvc-in-jsf-mvc-framework

So it's basically a M(M(M(MVC)C)C)C ;) [BalusC]

'fractal' MVC

http://stackoverflow.com/questions/10111387/understanding-jsf-as-a-mvc-framework?rq=1

Main point 3

When you use JSP pages according to a Model 2 architecture, there is a servlet that acts as a controller (process of knowing) that sets attribute values based on computations and results from a business model (knower), then dispatches the request to the servlet generated by the JSP page (known). The JSP servlet then retrieves the attribute values and inserts them into the designated places in the HTML being sent to the browser.

Science of Consciousness: Complete knowledge is the wholeness of knower, known, and process of knowing.

Main point 4 Preview

The JSP Standard Tag Library provides convenient action tags for many common operations on a JSP page. JSTL combined with the EL will satisfy most JSP needs.

Science of Consciousness: The TM Technique is a simple repeatable procedure that increases our ability to act effectively and achieve our goals.

“Scripting considered harmful”

- JSP scripting originated in early days of web apps
- most JSP scripting no longer used
  - Action elements and EL concepts used in JSF and similar frameworks
- might see in legacy code
- might see something similar in other frameworks
  - PHP
  - ASP.net
  - ASPMVC.net
  - …?
- scripting part of “Model 1 JSP architecture”
- see “memo” p314
  - circa 2000

major disadvantages of scriptlets are:

- Reusability: you can't reuse scriptlets.
- Replaceability: you can't make scriptlets abstract.
- OO-ability: you can't make use of inheritance/composition.
- Debuggability: if scriptlet throws an exception halfway, all you get is a blank page.
- Testability: scriptlets are not unit-testable.
- Maintainability: more time is needed to maintain mingled/cluttered/duplicated code logic.
- Sun Oracle itself also recommends in the JSP coding conventions to avoid use of scriptlets whenever the same functionality is possible by (tag) classes.
- StackOverflow: how to avoid Java code in JSP files? (2010)

Why You Shouldn't Use Java in a JSP

- You can do almost everything in JSP that you can do using Java.
  - But that does not mean you should do it.
- JSP is a technology that was designed for the *presentation layer*, also known as the view.
  - In most organizations, user interface developers are responsible for creating presentation layer.
    - These developers rarely have experience writing Java code, and providing them with the ability can be dangerous.
- In a well-structured, cleanly coded application, the presentation layer is separated from the business logic, which is likewise separated from the data persistence layer.
- It's actually possible to create JSPs that display dynamic content without single line of Java inside the JSP. That's our mission!!!

Recall: Forwarding a Request from a Servlet to a JSP

- A typical pattern when combining Servlets and JSPs is
  - Servlet accepts the request,
  - does business logic and data storage or retrieval necessary
  - prepare data that can be used in JSP,
  - forward request to the JSP.

```
request.getRequestDispatcher("/WEB-INF/jsp/welcome.jsp").forward(request, response);
```

JSP Actions

- JSP actions are xml elements or tags that the container executes
  - scripting often not suitable for HTML content developers
  - JSP Actions are predefined HTML-like elements for common processing tasks such as iteration, conditionals, database access, etc.

- JSP Standard Actions
  - “Standard” in sense that are included with every JSP implementation—part of JSP specification
  - Examples

```
<jsp:useBean id="connection“  class="com.myco.myapp.Connection“ scope=“session”>
        <jsp:setProperty name="connection" property="timeout" value="33">
</jsp:useBean>

<jsp:forward page="error.jsp" />

<jsp:include page="hello.jsp"/>
```

- JSTL
  - Java Standard Tag Library
  - many libraries of JSP actions (“tags”)
  - JSTL is one tag library that is widely used and has this name
  - not part of the base JSP implementation, must be added to an app as a library

Using JSTL

- There are five tag libraries in the Java Standard Tag Library specification:
  - Core (`c`)
  - Formatting (`fmt`)
  - Functions (`fn`)
  - SQL (`sql`)
  - XML (`x`)
- Examples of tags from the Core JSTL Library
  - `c:set(set value of a variable)`
  - `c:out`
  - `c:if(conditional)`
  - `c:choose`
  - `c:forEach`
- Many of these make simple use of the EL.

Quick EL Review

- An expression `${person.age}` retrieves an attribute object using “person” as a key (and prints to the page) something like: xyz`.getAge()`, where xyz is an instance of a Java object having an age property

- An expression `${expr}` prints the value of `expr` to the page, where `expr` is a Java expression using Java arithmetic or logical operators

c:out to avoid XSS attack

```
<p>The person's name is <c:out value="${person.name}" /></p>
<p>The person's name is ${person.name}</p>
```

- c:out escapes HTML characters
  - if person.name = <script>alert("You are hacked!")</script>
  - the script will be executed in the second case, but not when using c:out

JSTL example with body

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html><head><title>Weather Page</title></head>
<body>
<%String[][] data = {{"Nov 6", "32", "26"},{"Nov 7", "32", "26"},{"Nov 8", "32", "26"}};
request.setAttribute(“temperatures", data);%>


<table>
<tr><th>DATE</th><th>HIGH</th><th>LOW</th></tr>
<c:forEach var=“daily" items="${temperatures}">
  <tr>
    <td>${daily[0]}</td><td>${daily[1]}</td><td>${daily[2]}</td>
  </tr>
</c:forEach>
</table></body></html>
```

DATE	HIGH	LOW
Nov 6	32°C	26°C
Nov 7	32°C	26°C
Nov 8	32°C	26°C

JSTL demo

- visually preview the 6 demo steps
- what HTTP message will step 6 generate and what will happen on the server
- implement and run the demo
- if you finish quickly, try rewriting the JSP page using scripting instead of JSTL

Using JSTL

The JSTL library provides 5 kinds of tags, each having a different (standard) prefix. You “import” a library by placing a “taglib” directive at the top of your jsp page. Here are the choices:

- Core tags: (if/else, loops, choose …)

```
<%@ taglib prefix="c"  uri="http://java.sun.com/jsp/jstl/core"
  %>
```

- Function tags: (standard Java string manipulation)

```
<%@ taglib prefix="fn"
  uri="http://java.sun.com/jsp/jstl/functions" %>
```

Using JSTL (continued)

- Format Tags (format numbers, dates..)

```
<%@ taglib prefix="fmt"
  uri="http://java.sun.com/jsp/jstl/fmt" %>
```

- SQL Tags (set data source, perform queries)

```
<%@ taglib prefix="sql"
  uri="http://java.sun.com/jsp/jstl/sql" %>
```

- XML Tags (for navigating/parsing/working with XML files)

```
<%@ taglib prefix="x"  uri="http://java.sun.com/jsp/jstl/xml"
  %>
```

Main point 4

The JSP Standard Tag Library provides convenient action tags for many common operations on a JSP page. JSTL combined with the EL will satisfy most JSP needs.

Science of Consciousness: The TM Technique is a simple repeatable procedure that increases our ability to act effectively and achieve our goals.

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE

JSP: Knower, Known, and Process of Knowing

1. Java Server Pages make it easy for HTML authors to interact with servlets.

2. The JSP Expression Language is designed to promote easy access to dynamic content contained in the MVC data model.

3.Transcendental consciousness is the knower knowing Himself or Herself.

4. **Impulses within the Transcendental Field** Model 2 architectures are successful with large systems because they maintain the integrity of the knower (model), known (view), and process of knowing (controller). Similarly, when developers maintain the integrity of their own Self then their actions will be successful even when developing large complex systems under demanding conditions.

5. **Wholeness moving within itself:** In unity consciousness, one appreciates that knower, known, and process of knowing are all expressions of the same underlying unified field of pure intelligence, one's own Self, pure bliss consciousness.