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

Simplest JSP page

- The following code is nearly identical to the HTML document we are already familiar with.

```
<%@ page contentType="text/html;charset=UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <title>Hello World Application</title>
    </head>
    <body>Hello, World!</body>
</html>
```

- The interesting question is, "What's actually happening behind the scenes?"

What Happens to a JSP at Run Time?

- A JSP is just a fancy Servlet. ("syntactic sugar")
  - Every JSP file is translated back to a servlet **Java Code**.
- Your JSP compiles differently based on web container you run it on.
  - Your JSPs should run same on all of them, even if the Java code they get translated into looks completely different, as long as containers you use are complaint with the specification.
- Like all other Servlets running in your web container, JSPs have a life cycle.

JSP Lifecyle

- Translation
- Compilation (same for servlet)
  - The generated java servlet file is compiled into a java servlet class.
- Loading (same servlet)
  - The java servlet class is loaded into the container.
- **Execution Phase**
  - `jspInit()`
  - `_jspService()`
  - `jspDestroy()`

Four types of JSP elements:

- **Directives:** `<%@ directive %>`
  - page level operations for the translation phase.
  - used to direct the JSP interpreter to perform an action such as setting content type, importing classes, to include tag library etc.
- **Scriptlets:** `<% scriptlet %>` `<%= scriptlet expression %>`
  - embedded Java statements
- **EL Expressions**: `${ EL Expression }`
  - more convenient and powerful than a JSP expression
- **Actions using taglibs**: `<c:foreach />`
  - JSP "functions" that encapsulate some commonly used functionality

JSP scripting elements:

- **Declaration**:
  - <%! Inserts instance variable and method declarations into servlet p292 %>
  - `<%! int count = 0; %>`
- **Scriptlet**:
  - Inserts Java statements inside service method
    - <% inserts into the service method p282%>
    - `<% count = count * 10; %>`
- **Expression**:
  - expects a Java expression, which it puts inside 'out.print' in service method
  - `<%= ++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 comments are processed by container and do not appear in generated JSP.

JSP Scripting Demo

- Inspect the JSP demo code
- Find the generated Servlet
- Find the inserted JSP statements
  - %userprofile%\.IntelliJIdeayyyy.m\system\tomcat\

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" %>
```

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` object
`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 .

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);
```

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.

EL (Expression Language)

- Use of Java code inside of JSP is not a good idea, it is discouraged.
- There must be an easier way to display data and perform simple operations without using Java code.
  - What's needed is something easily read, familiar to both Java developers and UI developers, and with a simple set of rules and operators to make data access and manipulation easier.
- Expression Language (EL) was originally developed as part of the JSP Java Standard Tag Library (JSTL), to support the rendering of data on JSP pages without the use of scriptlets, declarations, or expressions.
  - It was inspired by and largely based on the ECMAScript (the foundation of JavaScript) and XPath languages
- When the JSP compiler sees the `${expr}`, it generates code to evaluate the expression and substitues the value of expression.

EL (Expression Language)

- recall JSP "expression"

```
<%= count++ %>
```

  - evaluates the expression and writes it out to the HTML page at its location on the page

```
<%= ((User) request.getAttribute("user")).getDog().getName() %>
```

- recall that attributes are where web app stores model values
  - values in model are computed and then accessed in page for display

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

Reserved Keywords

- As with any other language, one of the first things you should know about EL is its list of reserved keywords.
  - These are words that should be used only for their prescribed purpose.

`true`	`or or ||`
`false`	`not or !`
`null`	`eq or ==`
`instanceof`	`ne or !=`
`empty`	`lt or <`
`div or /`	`gt or >`
`mod or %`	`le or <=`
`and or &&`	`ge or >=`

Operator Precedence

[ ] , .
() is Used to change the precedence of operators.
unary –, !, not, empty
*, /, div, %, mod
+, – (binary)
<, lt, >, gt, <=, le, >=, ge
==, eq, !=, ne
&&, and
||, or
?:

- Remember that operators with the same precedence are evaluated in the order they appear in an expression, from left to right.

String literals

- Unlike Java, where string literals are always surrounded by double quotes, string literals in EL can be surrounded by either double or single quotes, similar to JavaScript.
- So, both of the expressions in the following example are valid.

```
${"This string will be rendered on the user's screen."}
${'This string will also be "rendered" on the screen.'}
```

- If some string has a single quote within it, it’s probably easiest to use double quotes for the literal.
- Similarly, if the string has a double quote in it, it’s probably easiest to use a single quote literal.

Implicit variables in EL

Variables	Description
pageScope	Collection of all page scope variables
requestScope	Collection of all request scope variables
sessionScope	Collection of all session scope variables
applicationScope	Collection of all application scope variables
param	Collection of all request parameter as single string value per parameter
paramValue	Collection of all request parameter as string array per parameter
header	Collection of all request header values as a single string value per header
headerValues	Collection of all request header values as string array per header.
cookie	Collection of all request cookie values as javax.servlet.http.Cookie value per cookie
intiParam	Collection of all application intialization parameter as single string value per value
pageContext	Instance of javax.servlet.jsp.Pagecontext class to access various request data.

Using the implicit EL scope

- The EL defines 11 implicit variables in the scope of EL expressions. However, the implicit scope is more useful and more commonly used because of its capability to resolve an attribute in the page, request, session, or application scope.
- When an EL expression references a variable, the EL evaluator resolves the variable using the following procedure:
  - It checks if the variable is one of the 11 implicit variables.
  - If the variable is not one of the 11 implicit variables, the EL evaluator next looks for attribute starting with the page scope until application scope, if not found it ignores the expression.

High level description of EL

`${something}`

- container evaluates this as follows
  - It checks if the variable is one of the 11 implicit variables.
  - otherwise 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 a bean then secondThing is a property of the bean
- if firstThing is a 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

Object Properties and Methods

- You cannot access fields using EL expressions, even if they are public.
- Considering a is object of class A, when the EL engine sees `${a.b}` syntax, it is looking for a property b of object a, not a field.
  - Property refer combination of getter and setter (mutator) for a field, named following JavaBean conventions.
- You can call method on EL expression using method name.

Example of EL Bracket notation

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

- in JSP:
  - `${myFruitList[0]} // Banana`
  - `${myFruitList["1"]} // Orange`
  - `${myFruitList.1} // won't work`

- If the thing inside the [ ] is String literal, it can be a Map or bean or index in a list or array.
  - Array index as string will be coerced into int.

EL is "null friendly"

- 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
  - ${750 + myBankAccount.balance}
    - "Um, where’s all my money?"

- 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 manifest universe. Our actions are more in accord with the laws of nature when we experience their basis in pure consciousness.

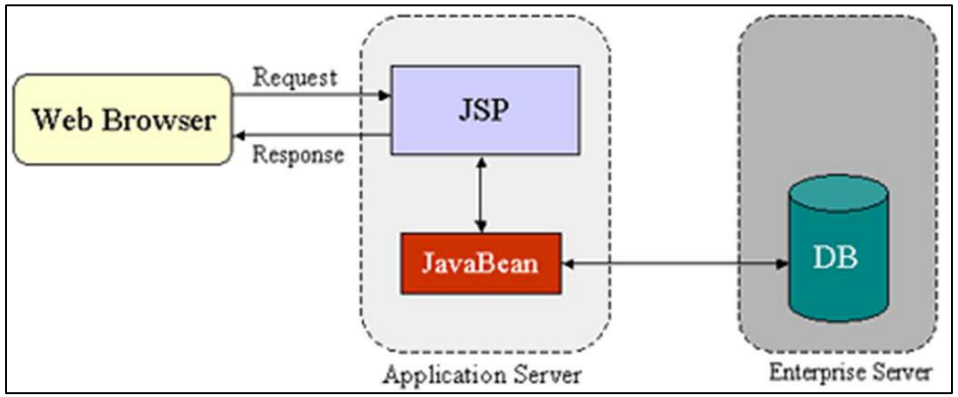
Main point 3 Preview

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.

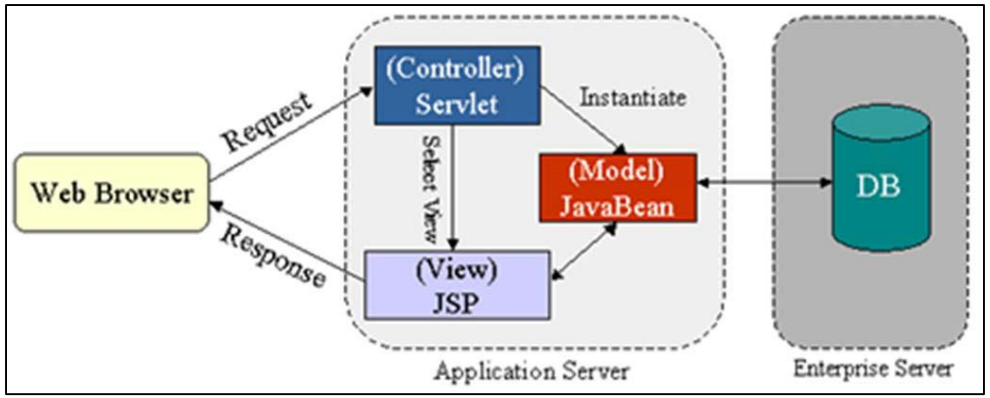
JSP Model 1 and Model 2 Architectures

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

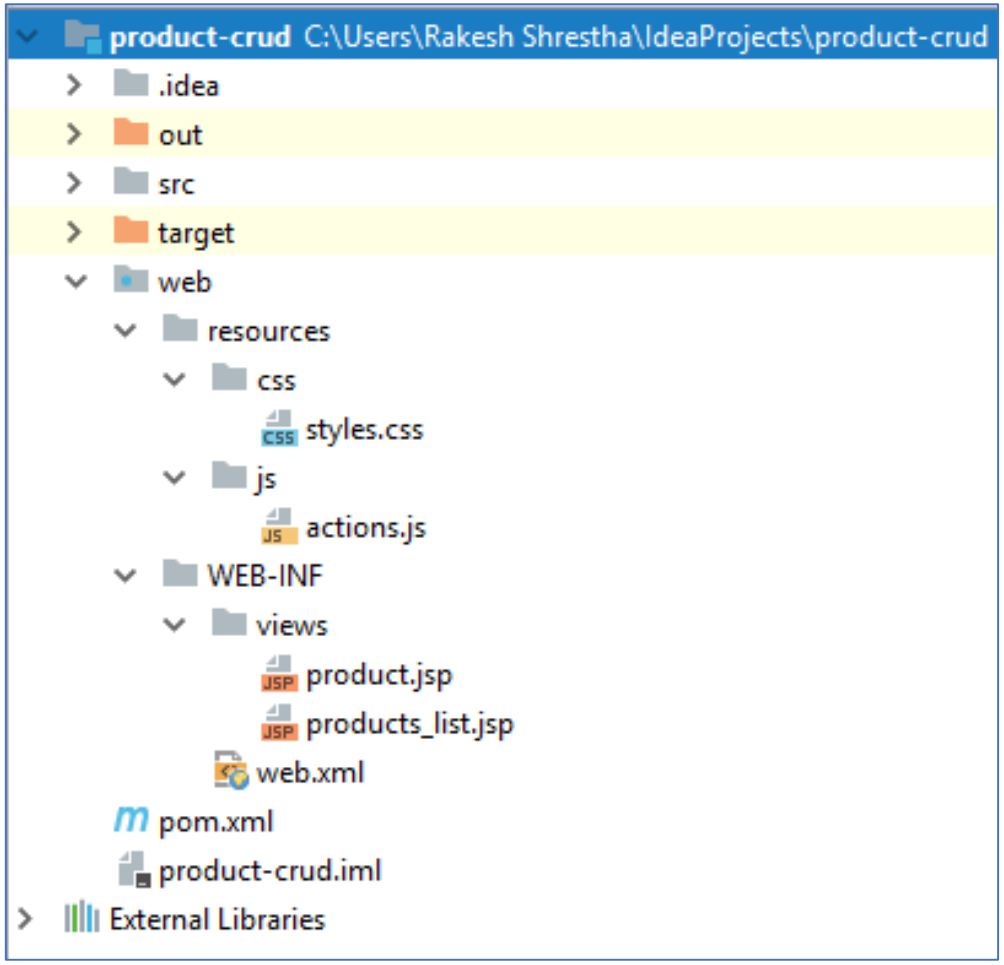


- More complex architecture
- For medium and large applications
- Advantages
  - Each team can work on different pages. Easy to understand each page.
  - Separation of presentation from control, making it easy to change one without affecting the other.
  - no Java code in HTML

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.

Linking static resources with JSP pages




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

```
<link href="<c:url value='/resources/css/styles.css'/>" rel="stylesheet"/>
```

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 itself.

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.