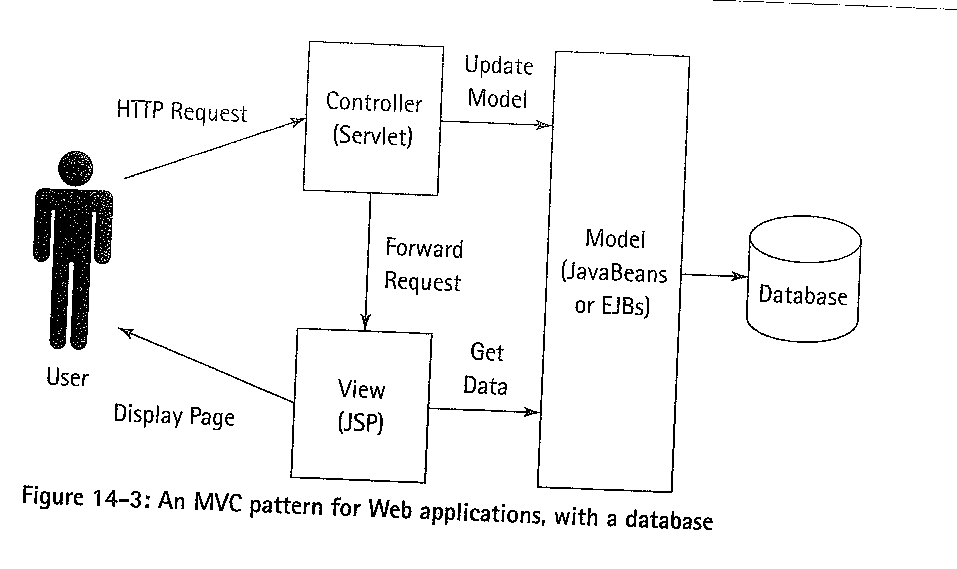
**Internet Programming**

**MVC**

**MVC Diagram**



**Roles of Java**

**Roles of Beans**

**HTTP**

**Roles of HTTP Commands**

* Basis for all **web transactions**
* Consists of client request / server response
* Request in plain English/ Response English or Binary

**How it Works**

* request for a web page ?
* Find the IP address of the machine
* Find any port numbers
* Find the name of the file requested
* Retrieve the file

**HTTP Request**

* Request has three sections
* The GET command
* The file information
* HTTP version
* followed by CR/LF

**Example of HTTP Request**

* Telnet to www.computing.dundee.ac.uk port 80
* GET /index.html HTTP/1.1
* HOST:www.computing.dundee.ac.uk

**HTTP the HEAD command**

Also have the Head command

* + Retrieves just the head of the requested document
  + Contains data about the document

**HTTP the POST command**

* + Can be used to send data to the server
  + We can use it to send data from forms

**Post**

* Name-value pairs invisible for user
* Binary upload possible
* Length limited by server settings
* Typically generated by form submission
* Usually URL Encoded
* Flash and Java (and other technologies) enable POST methods as well

**HTTP response**

First line is a version number, an error code and a response phrase : Errors are one of the following

* + Informational
  + Success
  + Redirection
  + Client error
  + Server error

**HTTP response codes**

* Informational are reserved for later use
* Redirection point the client to a new location
* Client error, the client made a mistake in the request.
* Server error, the server could not service the request
* 200 : OK
* 201 : Created
* 202 : Accepted
* 204 : No Content
* 301 : Moved Permanently
* 302 : Moved Temporarily
* 304 : Not Modified
* 400 : Bad Request
* 401 : Unauthorized
* 403 : Forbidden
* 404 : Not Found
* 500 : Internal Server Error
* 501 : Not Implemented
* 502 : Bad Gateway
* 503 : Service Unavailable

**Cookies**

* Some users disable cookies
* Limited amount of data (<4k)
* Limited number of cookies per server (<20)
* Can be persistent between pages and visits

=> session control

* Expires after a certain amount of time
* Set by server or by Javascript in browser
* Readable by Javascript in browser

**RESTful**

**What is Rest? - Representational State Transfer**

**REST - An Architectural Style, Not a Standard**

The WWW is a classic REST architecture

* + Use URL to request a resource
  + Resource is returned to you
  + You do not know the implementation of the resource

**REST request is URL**

Should always be of the form:

* http://www.parts-depot.com/parts/2465

**Error codes**

A Restful system should use error codes to indicate the success/failure of the call

**URLS - Universal Resource Locators**

**Only alphanumerics [0-9a-zA-Z], the special characters $-\_.+!\*'(), and reserved characters used for their reserved purposes may be used unencoded within a URL**

**Other characters are encoded as %xx**

**xx stands for the 2-digit hexadecimal representation of the character in the ISO-Latin character set.**

Exceptions: space can become ‘+’ as well as %20, ~ can stay unencoded.

PS Any char can be encoded. **E.g. “ = %22**

**Makeup of a URL**

**Protocol / Server id / File id**



**Server address**

Domain name / Machine Name / Port number

**File location**

Directory path / File name / location within the file

**When a file isn’t found**

* Default file
* A directory listing
* An error message
* A redirection

**Java Server Pages**

What are Java Server Pages?

**Completely Server Side.**

Full power of Java for Dynamic content.

* + Content based on the User.
  + Content based on a database.
  + Content based on a hardware device.

**Two Types of Server Side Java**

**Servlets (The Controllers)**

* Stand alone class files
* Written in Java and compiled by Java compiler

**JSP (The View)**

* Inside out Servlets, code is stored in web page.
* Written in JSP code compiled on the fly.

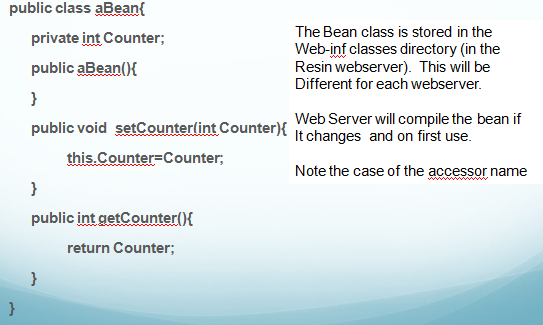
**How JSP is used**

* JSP is compiled into Byte Code
* Byte Code resides inside virtual machine
* Byte Codes has permanence.

**Beans are small class files with accessor methods**

* Java beans should not have public properties
* Any properties that need to be used should have accessor methods
* The Constructor method should have no arguments.

Bean code

****

**Including a Bean**

**<jsp:useBean i**

**d="myBean“**

**scope="page"**

**class="aBean" />**

The class tells this usebean directive which class to use from the Web-inf/classes directory. Later if we look at packages you will see how to separate beans out into logical groups.

**Passing control to servlet**

<jsp:forward page="/servlet/RSSFeedFull"/>

**Sending a Bean from Servlet to Jsp page**

Create a bean and fill with data

Add bean to request as Attribute

Forward to jsp page

**JSP Tags**

**Contain the code on a page**

* Usually are a <% %> pair.
* Pure Java is contained within these tags

**EG Counter in JSP**

<%! int counter=0;%>

<% counter++; %>

<h1>Your count is :<i> <%=counter%></h1>

* Code is reset if power is lost or memory is wiped

**HTMl Formatting**

* Standard Html is also on the page but outside the JSP tags
* <%= variable %> outputs a variable’s value to the web output.

**Including a File**

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

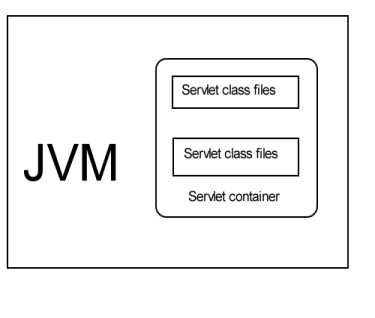
**Including a sub page**

<jsp:include page="/reset.jsp" />

**Changing Code**

* Page reflects that change on next request
* Code is compiled on the fly into class files. Will make more sense once you know more Java

**JSP (and Servlets) run in Servlet containers**

****

* **HTTP Request**
* **Web Server**
* **Servlet container**
* **Servlet Class Code**

**Deploying Servlet**

Servlet package goes in web-inf/classes

**Web.XML entry**

<servlet>

<servlet-name>HelloWorld</servlet-name>

<servlet-class>uk.ac.dundee.computing.aec.HelloWorld</servlet-class>

</servlet>

**Alternatives to doGet**

**doPost**

* + Used to handle POST requests from forms

**service**

* + Handles any request. You must determine the request type with the getMethod() method of the request type.

**doPut, doDelete**

**The Request class**

* Contains all information about the request.
* Includes data from Cookies

**The Response class**

* Sub Class of the ServletResponse class
* Allows the type of response to be set
* Allows Cookies to be added

Can add a redirection to the response:

* + res.sendRedirect(“/Outahere.html”);
  + Redirects the browser to another location
  + saves having to output HTML when you just want to go back to a set location.

**Packages**

* Packages are a convenient way to group servlets (and beans) and other resources.
* All files for a package need to be under the same directory

It’s conventional to start the package name with the top-level domain name e.g. com.sun.eng

**Why Packages?**

Starting with Java 1.4 the use of classes that are not in a package (sometimes called the "default" package") is deprecated. Java classes should always be in a package. In some version of Java, packageless classes are not going to be supported at all

**Roles of Ajax**

**How do we map rest to Servlets**

**Ajax**

**What is Ajax?**

**Ajax vs. normal web interaction**

**Databases**

How do we connect to them?

**How do we make them secure?**

What is Cross site scripting?

**How is cross site scripting prevented?**

**Cassandra**

**Cassandra Vs RDMS**

**Interfacing with Cassandra**

**Hector**

Astynax

**Web Applications**

**What is XML?**

**Role of XML in SOAP**

**What is SOAP**

* At it’s simplest SOAP is a way to “**wrap” method or procedure calls in a standard Internet protocol**
* At the moment, only HTTP is implemented, but some have looked at SMTP and others.
* SOAP though is only an enabler, it describes how the protocol works, it does not implement it

**Web Services are XML based**

In order to be considered a Web Service proper:

* The service must use XML as the data transmission
* XML ensures that the data is platform independent.
* The service should allow a client to make a request and get it’s value at a later date, **I.E Asynchronous**

**Loosely coupled**

* + If either the client or the service change then there is no direct need to change the other.

**Coarse Grained.**

* + We can expose individual methods in java for a client to use.
  + However to be useful as a web service we must expose high level methods that access our internal methods
  + E.G for a shopping app we wouldn’t expose “incrementtotal”, “addVAT”,”Addshipping” but “calculatetotalcost”

**Supports Remote Procedure Calls (RPC)**

* Simple Object Access Protocol SOAP
* Web Service Description language WSDL
* Universal Description, Discovery and Integration UDDI

**WSDL - Web Service Description language**

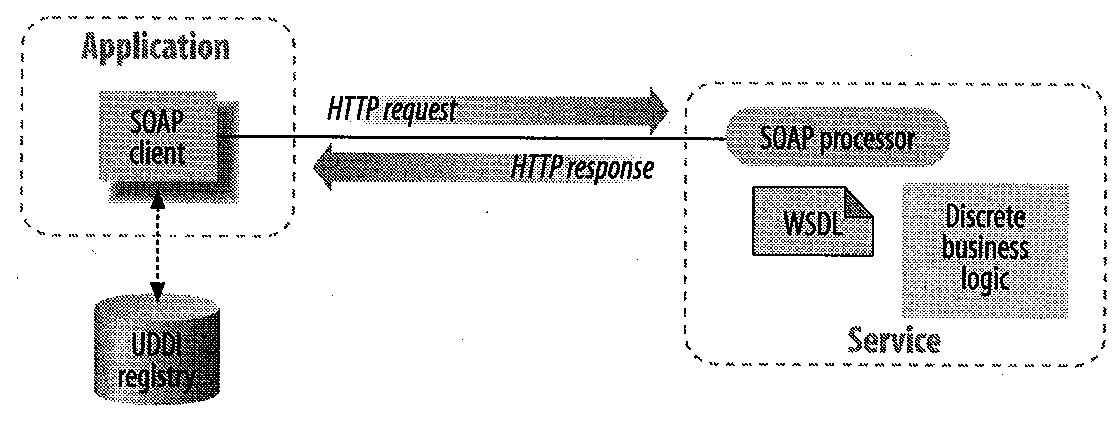
* XML technology that describes the interface of a web service in a standard way
* Allows clients to understand how to interact with a web service

**UDDI Universal Description Discovery and Integration**

* Provides a world wide registry of services
* Web Services can be “discovered” by searching for names, identifiers etc.

**Descriptor Scopes**

* + Application
  + Session
  + Request



**What is JSON?**

**What is RSS?**

Really simple syndication

JSON, RSS, XML comparison

**Server ADMIN**

**What is the importance of Log files?**

**Administer web servers**

**Types of Servers:**

* Light HTTP
* Tomcat
* Glassfish

**Client Server**

**Where is the “Intelligence”?**

**Factors we choose to use**

**Other Languages**

**What is the role of Javascript**

**Roles of Libraries**

**Java compared to Javascript**

Other Technologies

**Ruby on Rails**

**How is it different?**

**Node.js**

**What makes it special?**

**Pusher and associated techs?**