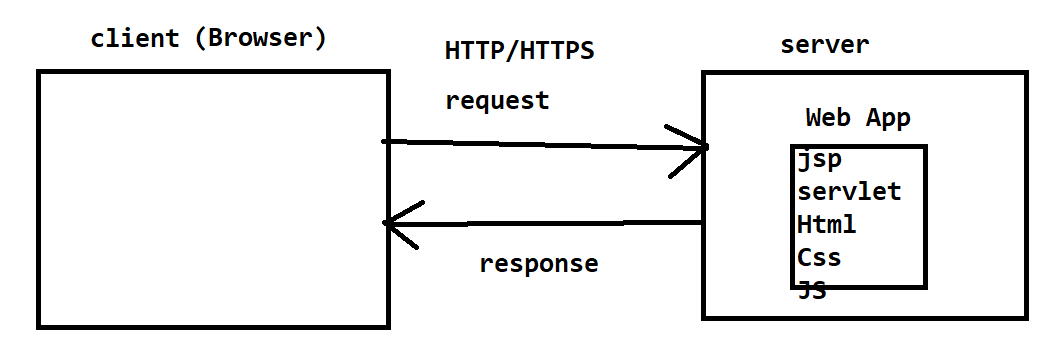
**Servlet and JSP**

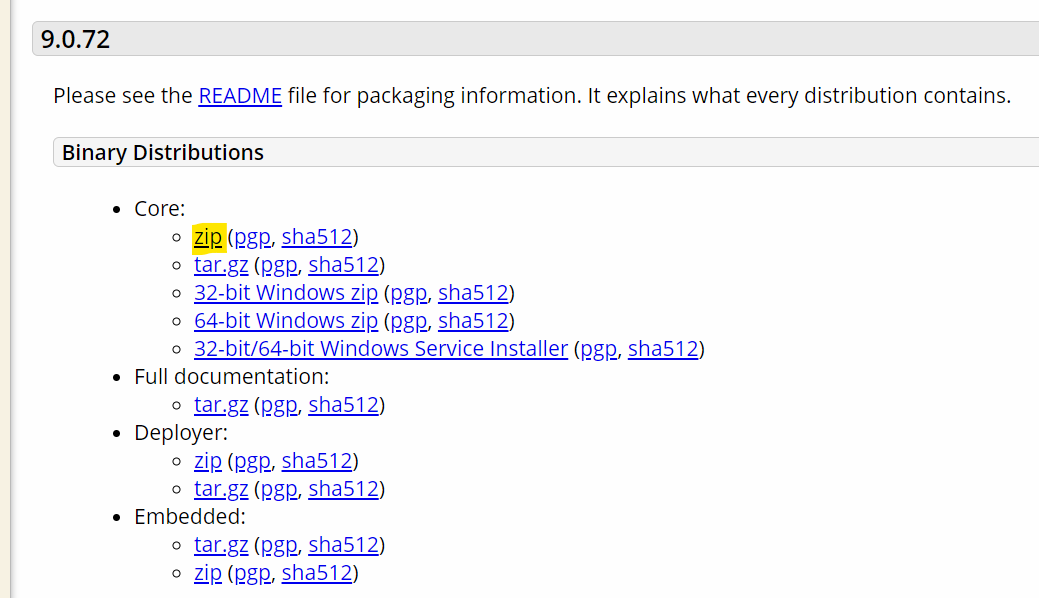
1. Using JSP servlet you can develop the web applications.
2. These applications are also known as client-server application
3. To Run type of application you will required the server along with JDK
4. In these application you will use a client and server side technologies.
5. Jsp and servlet are the server side technologies.



**Server Download and Setup**

1. Download Tomcat Server

<https://tomcat.apache.org/download-90.cgi>



1. Server Setup
   1. Extract a ZIP file into appropriate location.
   2. Setup The Server into Eclipse
      1. Open an eclipse
      2. Set Eclipse perspective to ‘Java EE’
      3. Go To ‘Servers’ tab at the bottom of the screen
      4. Click on Create new Server Option
      5. Expand the ‘Apache’ option
      6. Select the Apache version form the list
      7. Click on Next.
      8. Set the installation directory
         1. Browse a location where you extracted you zip file.
         2. Select the parent folder of bin, config, lib etc.
      9. Click on Next and then Finish

**Web Application**

* + - 1. The application which can be execute from the browser over an network.
      2. Web Applications are of 2 types
         1. Static Web Application

The content of the page is common for all the users

These application are informative web sites.

* + - * 1. **Dynamic Web application**

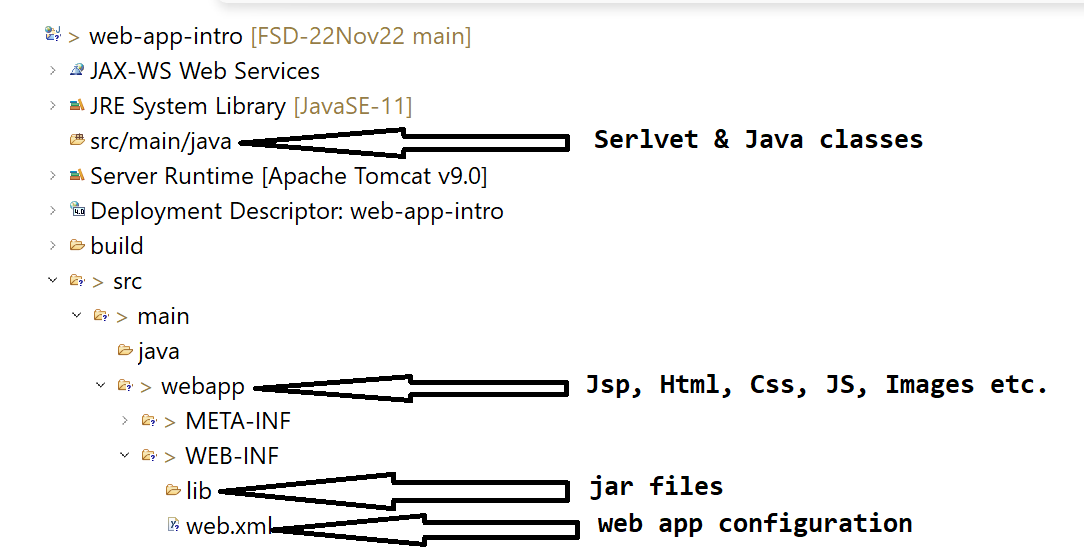
The content of the page is different for every user.

In this application you can perform the activities which is specific to the user.

**Create and Execute Dynamic Web Project**

1. Go To ‘New’ -> click on ‘Dynamic Web Project’
2. Set the name of the project.
3. **Make Sure than Target Runtime is not <NONE> and is must be selected (Tomcat server)**
4. Click on Next and Next
5. Select the check box for Deployment Descriptor.
6. Click on Finish

**Dynamic Web Project Structure**

****

**Protocol**

TCP, IP, FTP, FTPS, SMTP, UDP, **HTTP, HTTPS**

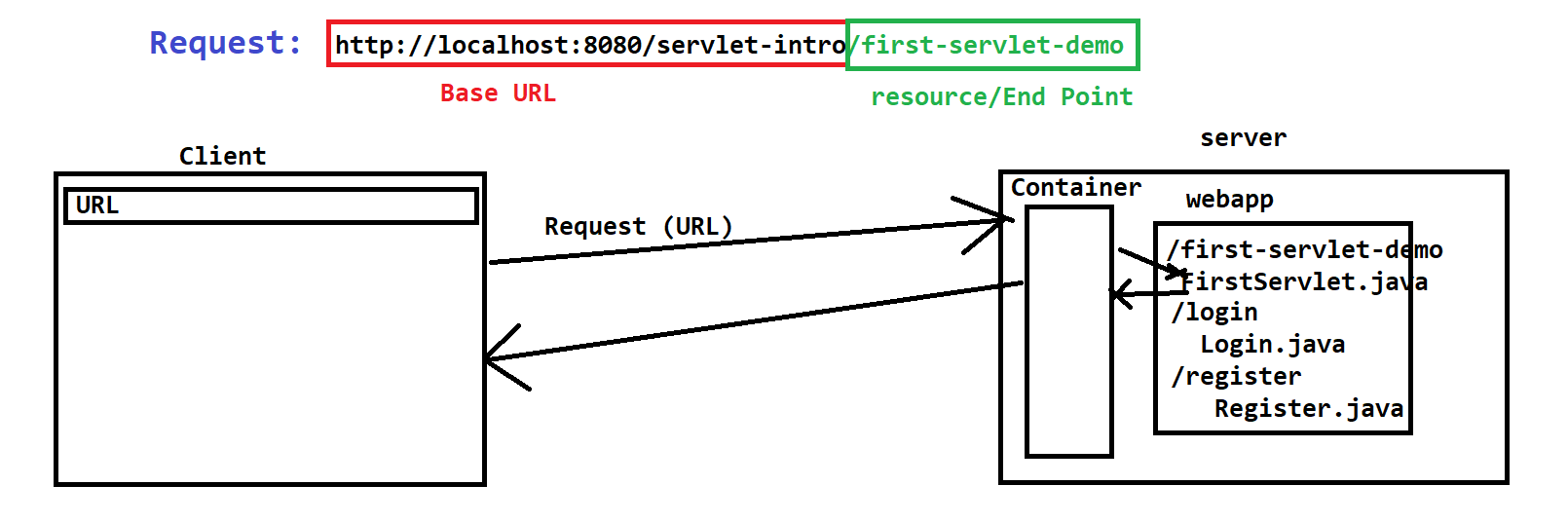
**Servlet**

1. Servlets are the java classes.
2. Servlet do not have main methods.
3. Main Purpose of using this is to get the request, process the request and generate the response.
4. Servlet is use to create dynamic web pages.
5. Inside servlet you can use the Java code as well as HTML code.
6. The HTML code has to add inside the java code and hence it is also known as HTML in JAVA
7. Every servlet will be access using an URL. There must be a unique URL provided to a servlet.
8. Servlet is maintain by the server, that is object creation and object management will be taken case by Server.
9. There is only one object of the servlet created in an application.

**How to create Servlet**

1. There 3 options to create servlet
   1. By extending **Servlet** class
   2. By Extending **GenericServlet** class
   3. By Extending **HttpServlet** class
2. Create a java class use any one of the class as a super class.
3. Override the service method. And implement your request and response code inside the service method.
4. Provide the URL for the servlet so that client can access the servlet by using URL. To provide the URL you can use **@WebServlet** annotation on the class.

**Request and response processing flow**

****

**Returning the Response to client**

1. You will required the object if HttpServletResponse
2. You have to set the type of response which is also known as MIME type.

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types>

1. MIME type can be set using the response object method ‘**setContentType(“MIME\_TYPE”)**’
2. You have to get an object of PrintWriter (text type) or InputStream (binary) to write the response.

**PrintWriter out = response.getWriter();**

1. Write use response into the print() method.

**Request handling and Processing**

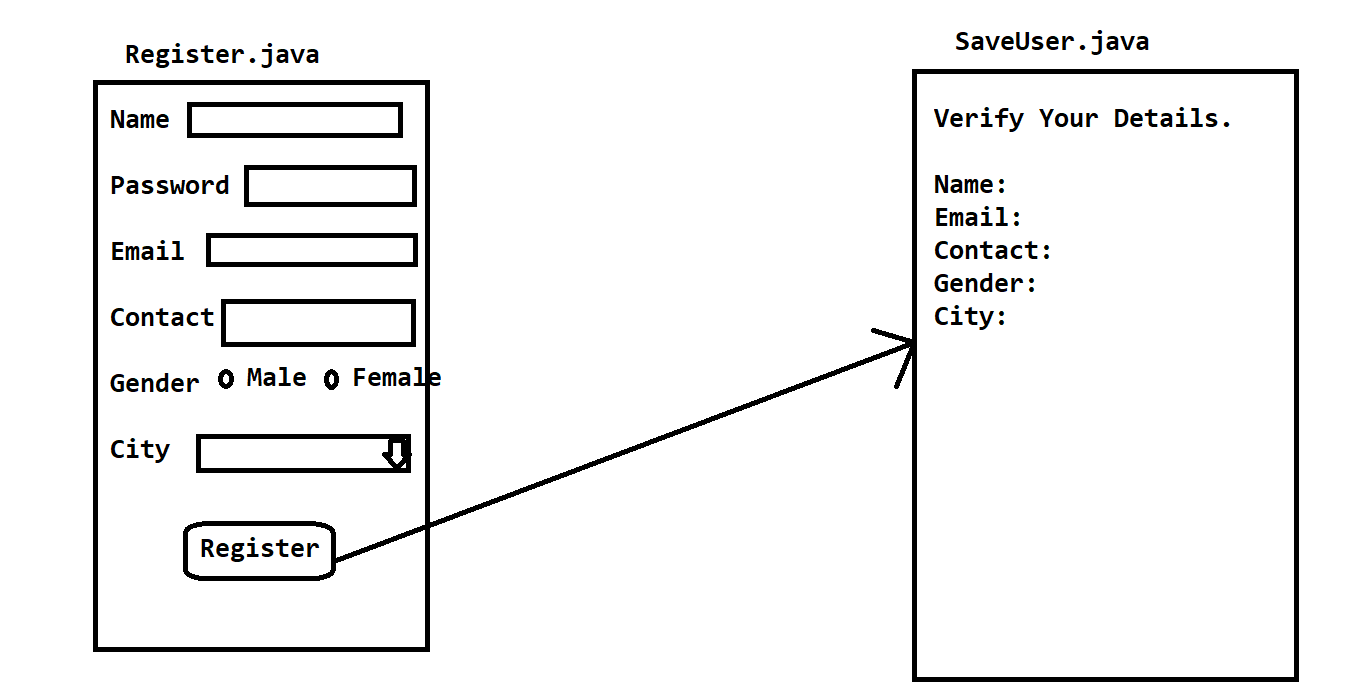
1. To Handle a request you can use the HttpServletRequest Object.
2. From the request Object you can get the data which is in the form of parameter or attribute.
3. Request can also be consider as a URL.

**Parameter**

1. Parameter is a user data.
2. Parameter pass from the request or from the request body.
3. Every parameter has a key and value.
4. Parameters are visible inside URL after ‘?’
5. These parameters can be access by request object.
6. Every Parameter is in the form of String only.
7. There can be a multiple parameters in the URL which must be separated by ‘&’
8. To Get the parameter from the request you can use a request object and getParameter method.
9. Syntax:

Request.getParameter(“Key”) : String

Task-1



**Add Jar file in web application**

1. Copy a jar file form the folder location.
2. Paste the jar file inside the application at location src/main/webapp/WEB-INF/lib

**JSP (Java Server Pages)**

1. JSP is used to create dynamic web pages.
2. JSP is mainly used for designing a web pages.
3. On JSP you can use HTML code directly.
4. Along with HTML code you can also use CSS, JS, JAVA code as well.
5. You can write a java code inside HTML and hence it is also known as JAVA in HTML.
6. JSP page extension is .jsp
7. Every JSP page converted into servlet first at the time of execution.
8. The by default URL of the JSP page is /pagename.jsp
9. JSP has to be created inside webapp

**Scripting Elements/Tags**

1. These tags are used to write a java code on the JSP page.
2. The java code is required on the JSP page to make it dynamic page.
3. There are 3 types of scripting tags
   1. Scriptlet Tag
      1. This tag is use to added a java code on JSP page.
      2. The code added in side this tag is a local code , which gets added inside service method after conversion.
      3. The variables created inside this are the local variables.
      4. Syntax:

<% Java Code %>

* 1. Expression Tag
     1. You can execute and display expression result on the browser using this tag.
     2. The expression added in side this will included inside the service method.
     3. The code added inside this tag will be a part of out.print(expression).
     4. Syntax:

<%= Java Expression %>

* 1. Declaration Tag
     1. This tag is use to write the instance level code.
     2. Using this tag you can create a methods, static variable.
     3. You can also create static methods using this tag.
     4. Syntax:

<%! Java Code %>

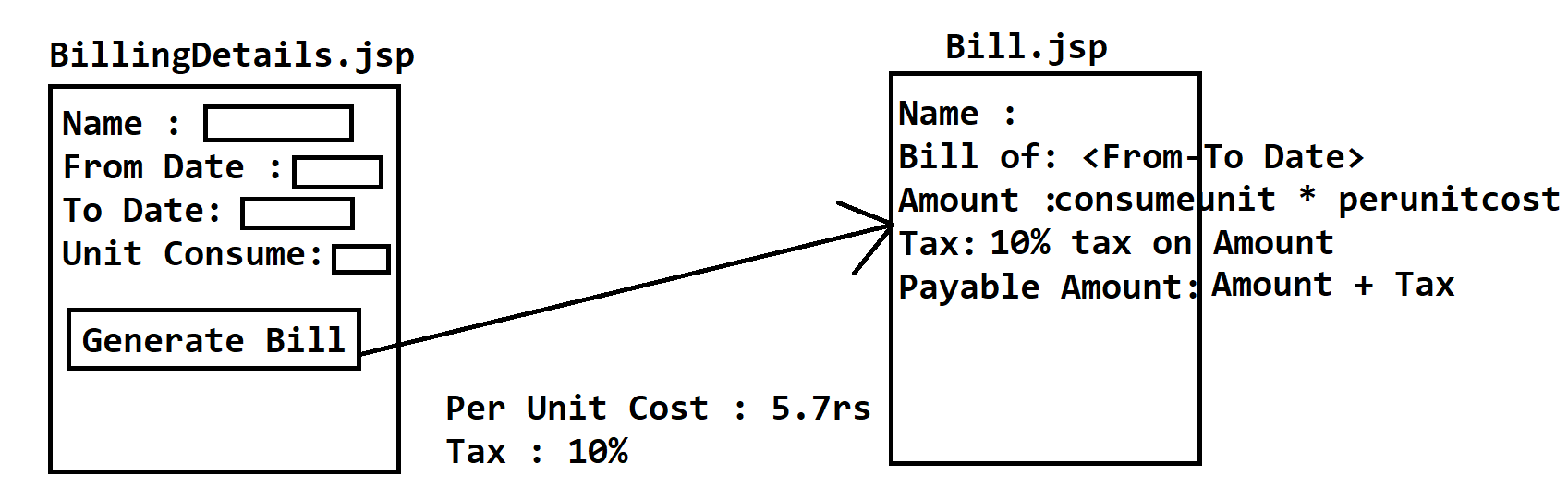
**Implicit Object**

1. Every JSP page has some object which are internally provided by the server.
2. There are total 9 implicit object present on every JSP page

|  |  |
| --- | --- |
| **Object Name** | **Class/Interface Name** |
| request | HttpServletRequest |
| response | HttpServletResponse |
| session | HttpSession |
| application | ServletCotext |
| out | JspWriter (PrintWriter) |
| page | this (in java) |
| pageContext | PageContext |
| config | ServletConfig |
| exception | Throwable |

Note: **These implicit Object are only accessible inside Scriptlet and Expression tag it is not allowed in Declaration tag**

**Task-1**



**Redirection techniques**

1. It is use to redirect user from one page to another without any user action.
2. There are 2 types of redirection techniques
   1. **Request Dispatcher**
      1. Is an interface
      2. It is use to go from one page to another without any user action.
      3. Request Dispatcher is use to redirect from one page to another using same/existing request.
      4. So the current/same request details will be also carried as it is to the next page.
      5. RequestDispatcher has 2 methods

**forward(request, response)**

**include(request, response)**

* + 1. **Syntax**

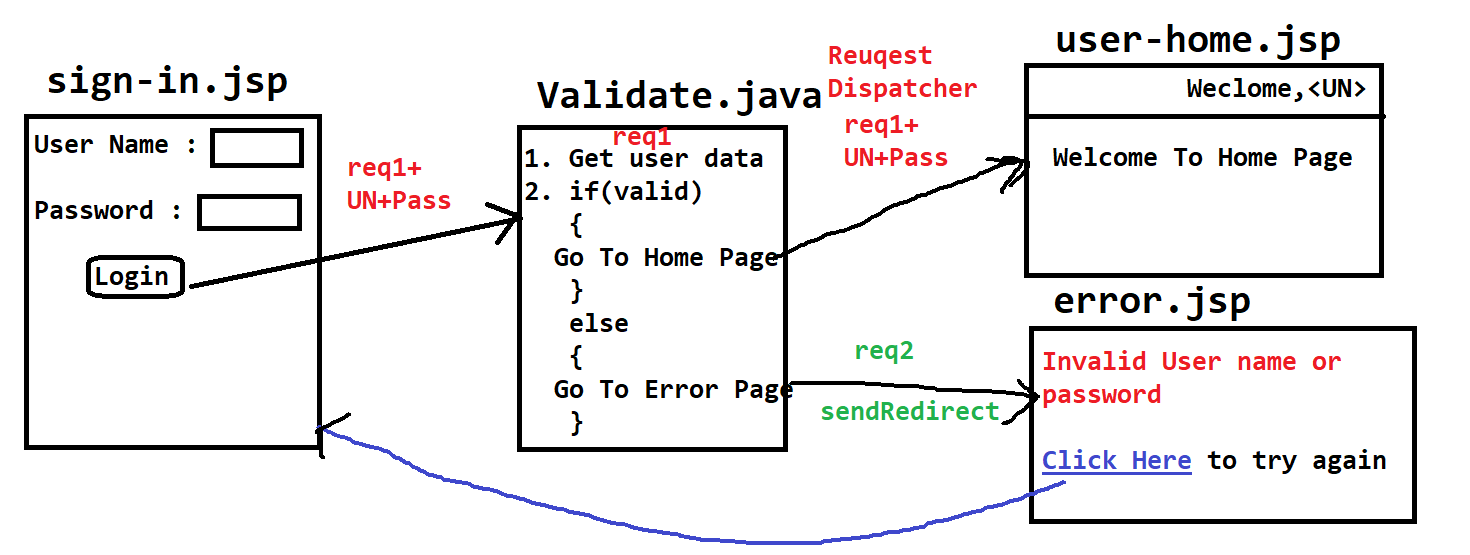
**RequestDispatcher dis = request.getRequestDispatcher("<URL>");**

**dis.forward(request, response);**

**dis.include(request, response);**

* 1. **Send Redirect**
     1. Is a method.
     2. It is use to go from one page to another without any user action.
     3. SendRedirect is generate new request to go from one page to another.
     4. The current request gets deleted and the new request will be generated in this case and hence the current request data will no longer available inside new page.
     5. Syntax:

**response.sendRedirect("<URL>");**

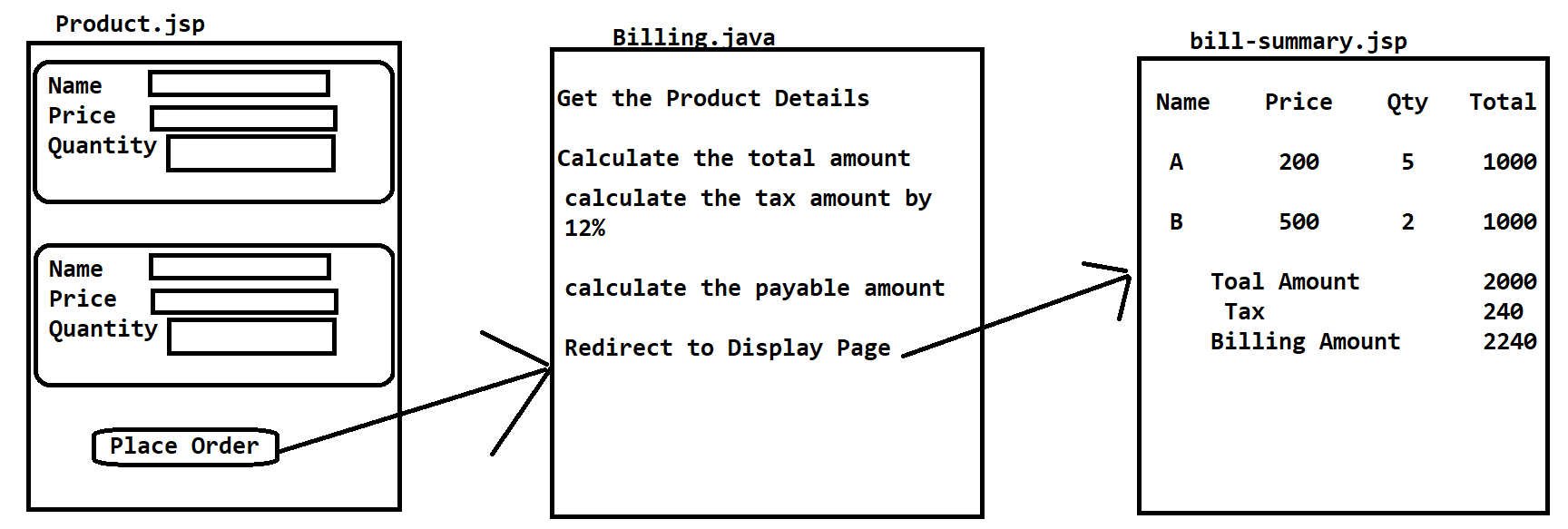


**Attribute**

1. Attribute is the use details/info which is in the form of Object.
2. Attributes use to pass user info from one page to another.
3. Every Attribute has a key and value pair.
4. You set and get attributes in the request object using following methods

request.setAttribute(“Key”, value);

request.getAttribute(“Key”) : value

Task-1  
  


**Session Tracking Techniques**

1. You can manage the data from old request to new request using these techniques.
2. There are different ways to achieve this
   1. Hidden Form Field
      1. This technique is used when new request created due to form tag and submit button.
      2. You can retain the old request data into newly generated request.
   2. Url Rewriting
      1. This is used when the new request generated using anchor tag or sendredirect.
   3. HttpSession
      1. It is use to maintain the user information at server side.
      2. The data which is placed inside a session will be available in all the pages of the application.

**Steps to follow for Project development**

1. get the project details

2. Create a modules from the project topic

3. Select a module and get the details for that modules

a. Requirement for module

b. Data dictionary

first name, middle name, last name, gender, email

contact, city, pincode, state, addressline, id, edutile,

passing year, percentage......

c. normalization on the Data dictionary to get the Table (ERD)

4. Create views (Wire-frame)

5. Coding

**MVC (Model View Controller) Pattern**

1. MVC is a pattern which is use to develop the application
2. Here the application will be divided into 3 component(parts)
3. **Model**: Model is use to execute business logic / DB Operation. These are the java files in which you used a java code and JDBC for Db connection.
4. **View**: View is use to create UI/UX, You can use a JSP pages in this component. Here HTML, CSS, JS Java code will be used. User always interact with view.
5. **Controller**: It is use for a redirection. It is also use to connect view with model (intermediate between View and Model). Servlet technology is used as a controller.

**Task Management (ToDo Management)**

1. Create a new task (Scheduled a task)
2. Update the task or task status.
3. Delete the task.
4. Get all the task.
5. Search Task by scheduled date and status.

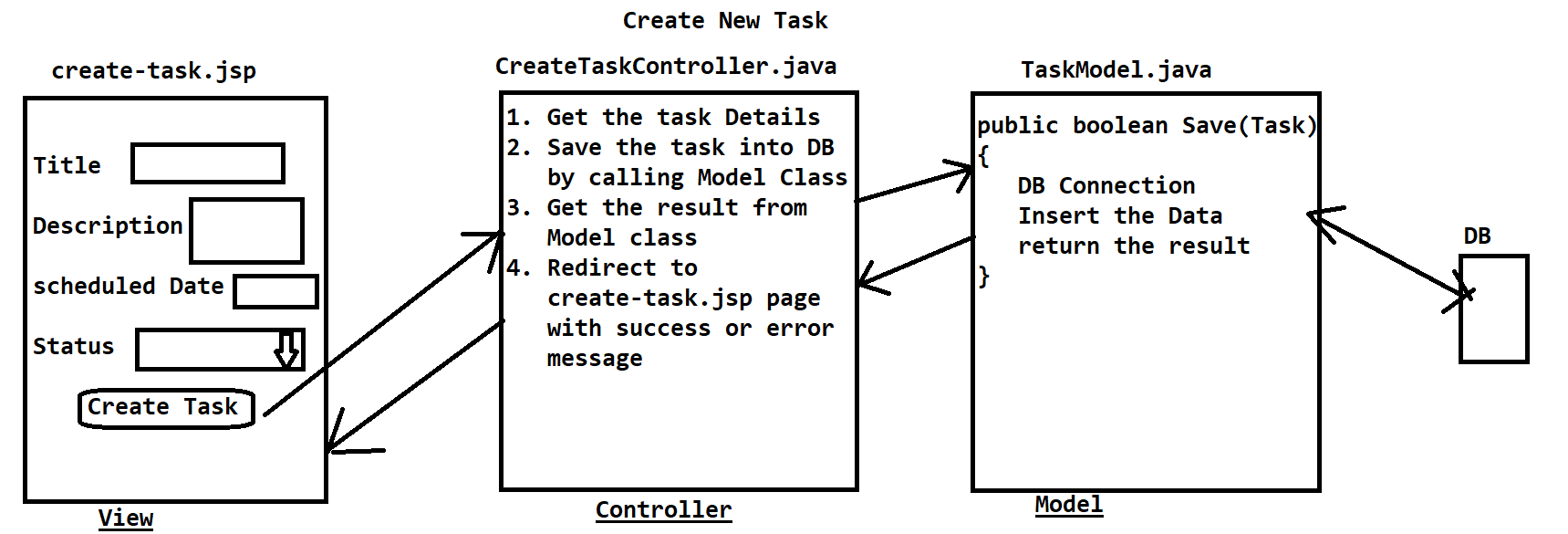
Data (Table)

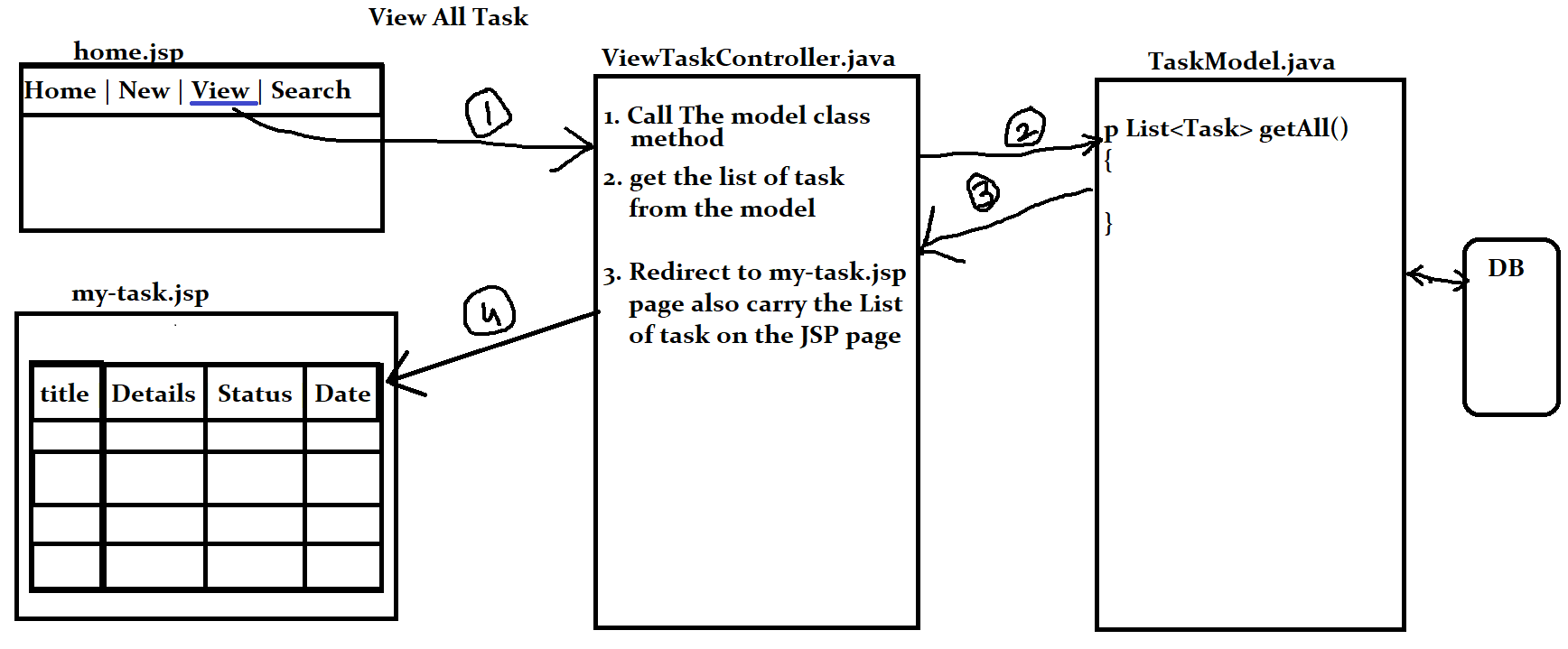
Id, title, detail, scheduled data, updated date, status (Open, InProgress, Close)

**Table : task**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Description** |
| id | Int | Primary Key and auto increment for every task |
| title | Varchar(30) | Title of the task |
| details | Varchar(1000) | This is the description of the task |
| schDt | Varchar(10) | Scheduled date of the task |
| updated | Varchar(10) | The date on which the task is updated |
| status | Varchar(10) | The status will be “Open, Inprogress, Close” |

**Create Task Flow**





Update the Task

