

## JAVA EE COURSE

# ROLE OF SERVLETS AND JPS IN JAVA EE



By the expert: Eng. Ubaldo Acosta



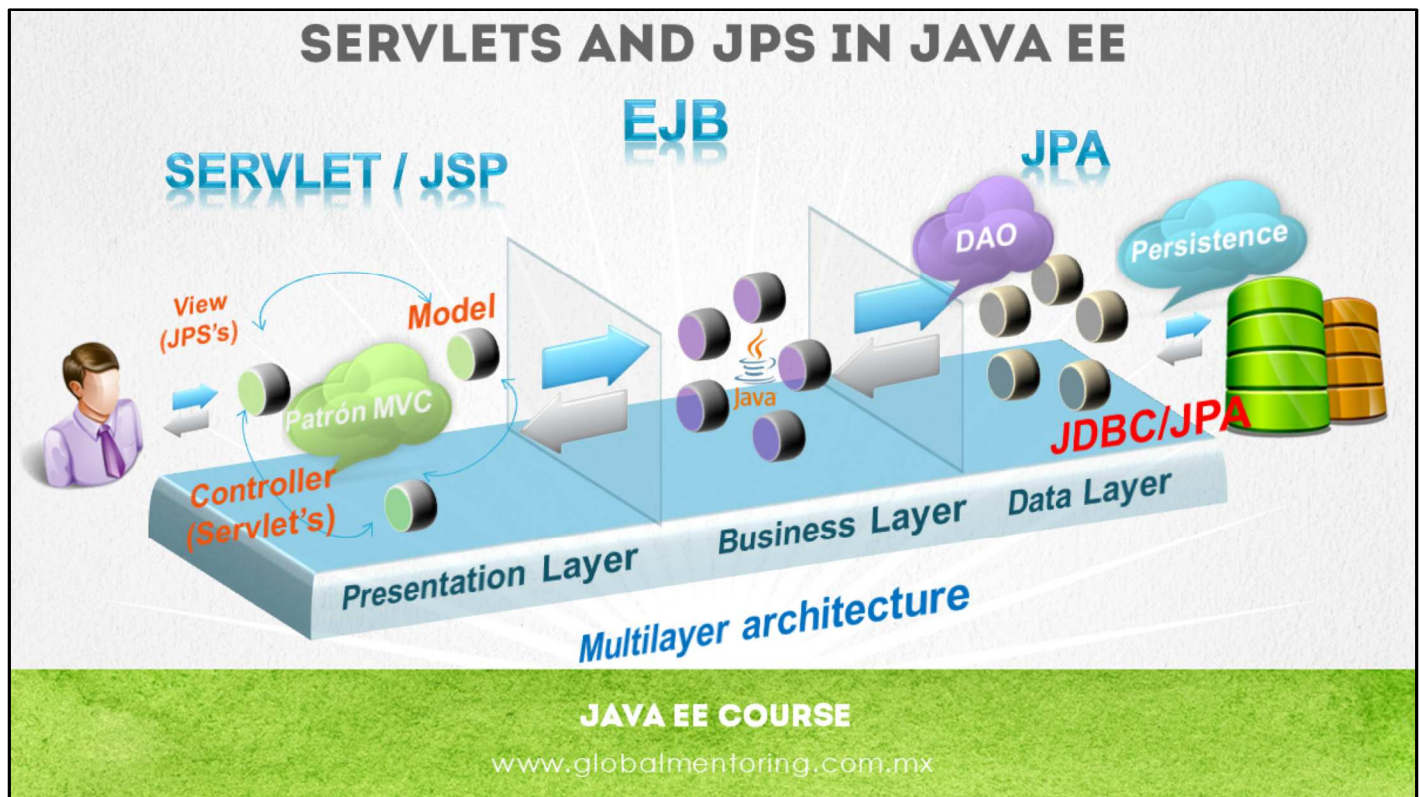
**JAVA EE COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

Hello, Ubaldo Acosta greets you. Welcome again. I hope you're ready to start with this lesson.

We are going to study the role of Servlets and JPSs in a Java Enterprise application.

Are you ready? OK let's go!



In the figure we can see the role of Servlets and JSP technologies in a Java Enterprise architecture. These technologies apply directly to the presentation layer, which is responsible for tasks such as:

- ✓ Generation of Presentation Code, with technologies such as HTML, CSS and JavaScript.
- ✓ Processing of HTTP requests from the client through HTML forms.
- ✓ Validation of parameters received in the HTTP request.
- ✓ Recovery of model information, relying on the Service Layer (EJB's) for this.
- ✓ Processing of the response and selection of the view to be shown to the client. Among several other tasks.

The reasons why we create multilayer business architectures are many, here are some:

- ✓ **Abstraction and Encapsulation:** The layers abstract the details of the work done by each layer. Thus, each will be responsible for certain processes, without having to involve the other layers to perform their fundamental tasks.
- ✓ **Clearly defined functionality:** Each layer defines its tasks, allowing to delegate properly the work among the group of programmers
- ✓ **High Cohesion:** This means that a layer performs only the task for which it was created, and the other functions are delegated or supported by the other layers.
- ✓ **Low Coupling:** Because each layer has the concept of abstraction applied, there are only the minimum necessary communication points between the different layers, allowing coupling and uncoupling without problems.
- ✓ **Reusable:** As a result of low coupling and other features, it is possible to reuse complete layers for different projects.

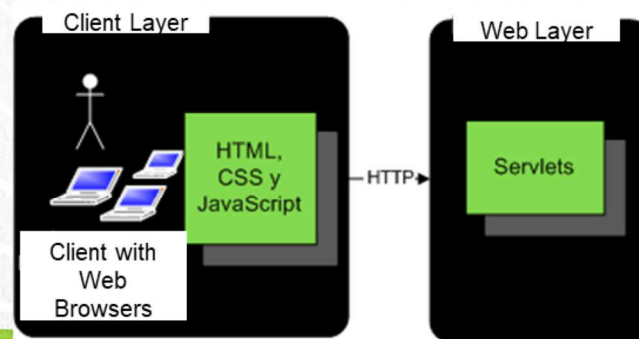
In addition, we can have the following benefits when using layers in a business architecture:

- ✓ **Isolation:** It allows to reduce the impact to change the technology, since we can go gradually migrating a layer to a new technology, without impacting too much in the total system.
- ✓ **Performance:** If there is a need to distribute the layers between different servers, it is possible to do it, with the aim of adding concepts such as scalability, reliability, fault tolerance and in conclusion to improve the complete performance of the system
- ✓ **Improvements in tests:** The clearer the responsibilities of each layer, we can perform tests of and discard software failures by isolating the problem very precisely.

In this lesson we will review how to integrate Servlet and JPS technologies into a Java EE architecture.

## CHARACTERISTICS OF SERVLETS

- ✓ A Servlet is a Java class that allows you to process Web requests.
- ✓ It allows reading web client information (request parameters).
- ✓ It allows generating a response to show the client (HTML and binary files such as PDF, Audio, Video, etc.)



**JAVA EE COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

A Servlet is the most basic component of Java technology for the development of Web applications. Even a JPS, when compiled, becomes a Servlet.

This Java class allows processing a request based on the HTTP protocol (Hypertext Transfer Protocol). This is the protocol that revolutionized the use of the Internet at the time, since through it it is possible to request Web pages from Servers of different technologies such as Java, PHP, .Net, among many other technologies.

The Servlets, are Java classes that have a well defined life cycle, and are administrator by a container of Java Web applications, therefore, it is not enough with a JDK to be executed, but we must have at least one Tomcat server or , if it requires the integration of more technologies, a complete server of Java applications such as Glassfish, WebSphere, Weblogic or JBoss.

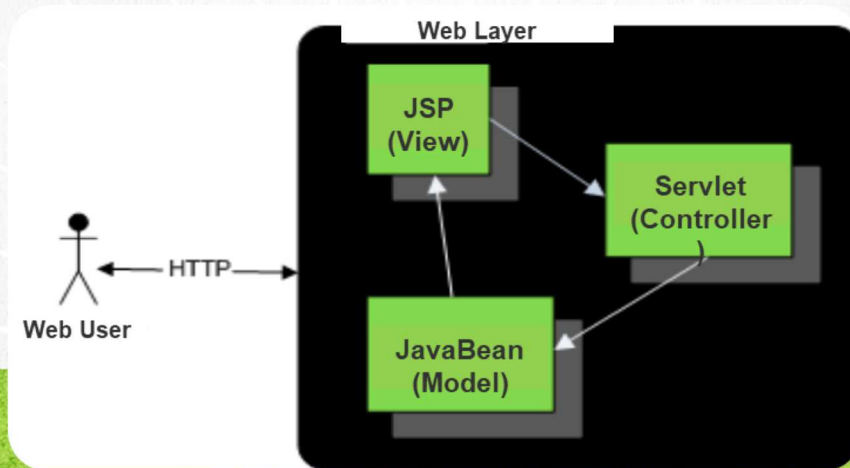
As we can see in the figure, a Servlet class allows to process the client's request and can generate a response dynamically. The response content can be HTML code or binary files, such as PDF, Audio, Video, etc.)

The Servlets are oriented to process the client's request, and therefore the code they contain is Java code, being able to embed HTML code to generate the response to the client, however, precisely to avoid the latter is that we are going to use the technology of the JSPs, which are oriented to have mainly HTML code and embed Java code.



## FUNCTIONS OF A SERVLETS

- ✓ A Servlet contains Java code, and you can add HTML code.
- ✓ A Servlet is used as a class that controls the flow of a Web application (Controller) in an MVC (Model - View - Controller) architecture.



In the figure we can observe the responsibilities of a Servlet and the role it plays in the MVC design pattern. The Servlet is the controller of the application and therefore processes the client's request, retrieves the model information and sends back the response to the client, being able to support the JSP's for this last task, or the Servlet can do it directly (although it is not recommended).

General steps that a Servlet Controller executes:

a) We process and validate the parameters (if applicable)

```
int idPerson = request.getParameter ("idPerson");
```

b) We perform the presentation logic by storing the result in JavaBeans

```
Person person = personService.findPersonById (idPerson);
```

c) We share the bean object to be used in some scope

```
request.setAttribute ("person", person);
```

d) Redirect to the selected JSP

```
RequestDispatcher dispatcher = request.getRequestDispatcher ("list.jsp");
```

```
dispatcher.forward (request, response);
```

## CHARACTERISTICS OF JPS

- ✔ The JavaServer Pages (JSP's) are server-side components, specialized in handling HTML code, and embed Java code by means of tags (tags).
- ✔ The JSP's are used as presentation components, that is, to show the information obtained or processed by the Servlets.
- ✔ A JSP when compiled creates a Servlet automatically and on the fly. Therefore, the life cycle of a JSP is very similar to that of a Servlet.

### JAVA EE COURSE

www.globalmentoring.com.mx

The JSPs are Java components on the server side, which are responsible for displaying the response information to the client, dynamically generating the necessary HTML content. These are some of the benefits of using the JSP's:

- ✔ We can focus on writing HTML code, making it easier to maintain the presentation layer
- ✔ We can use design tools to visually create HTML pages and embed the dynamic labels of the JSP's
- ✔ Separate the Java code display code
- ✔ Each member of the development team can focus on different tasks, by separating responsibilities.

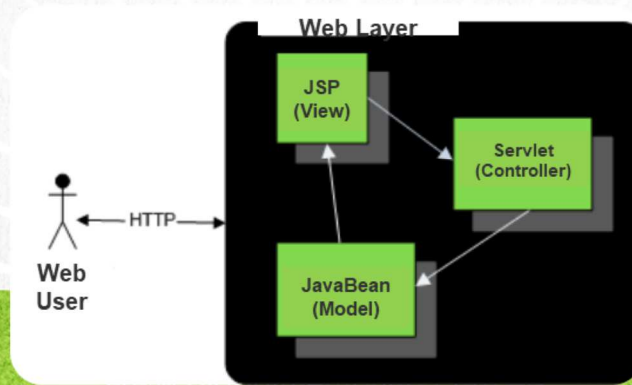
JSP's technology relies in turn on other technologies to simplify the deployment of information.

- ✔ Expression Language: The expression language (EL) simplifies the access to the information shared by the Servlets, according to the MVC model, and with it the JPS focuses on its primary task, which is to process the information to send the response to the client .
- ✔ JSTL (JSP Standard Tag Library): The JSTL library is a set of tags that we can use in JSPs, and they allow us to simplify tasks in which we would normally have to add Java code, allowing to have a cleaner and maintainable code.

We will use these technologies in the business technology integration exercise that we will carry out later.

## FUNCTIONS OF A JSP

- ✓ A JSP contains HTML code and can add Java code through tags (tags)
- ✓ A JSP is used should be used as a presentation component (Vista) in an MVC architecture (Model - View - Controller)

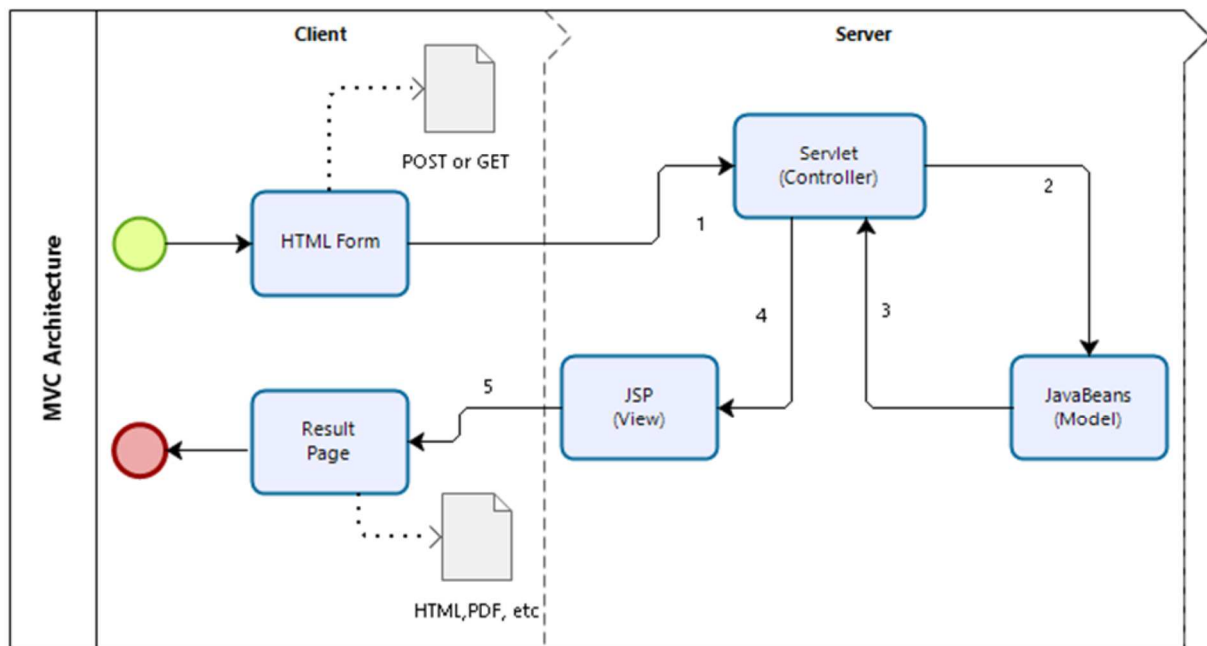


Sample Code of a JSP list.jsp that uses a list of Person objects as Model, which was shared by a Servlet:

```

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<html>
  <head>
    <title>List of People</title>
  </head>
  <body>
    <h1>List of People</h1>
    <table border="1">
      <tr>
        <th>Name</th>
        <th>Email</th>
      </tr>
      <c:forEach var="person" items="${people}">
        <tr>
          <td>${person.name}</td>
          <td>${person.email}</td>
        </tr>
      </c:forEach>
    </table>
  </body>
</html>
  
```

## MVC ARCHITECTURE WITH SERVLETS AND JSP S



Example of Code of a Controller Servlet applying the MVC pattern:

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    //1. We process the parameters
    String idPersonString = request.getParameter("idPerson");
    int idPerson = 0;
    if (idPersonString != null) {

        //2. We create / retrieve the Model
        idPerson = Integer.parseInt(idPersonString);
        Person person = new Person(idPerson);
        person = this.personService.findPersonById(idPerson);

        //3. We share the Model with the View
        request.setAttribute("person", person);

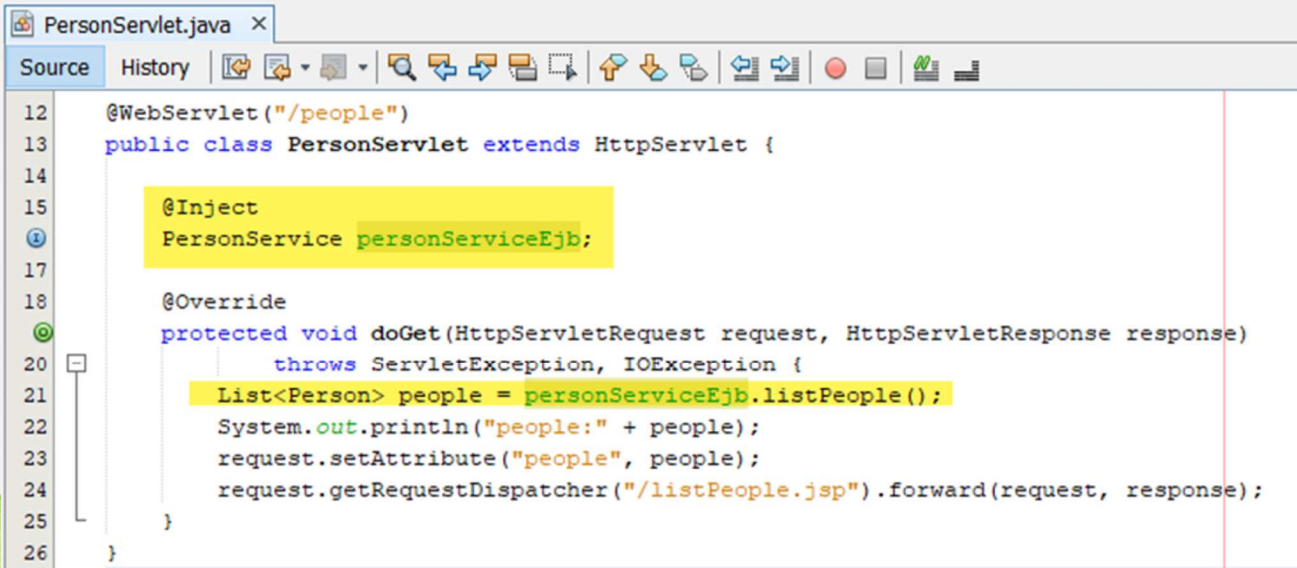
        //4. Redirect to the View that will show the Model
        request.getRequestDispatcher("editPerson.jsp").forward(request, response);

    }
}
```



# INTEGRATION BETWEEN SERVLETS AND EJB S

- Example of a Servlet that accesses the EJB Service:



```

12  @WebServlet("/people")
13  public class PersonServlet extends HttpServlet {
14
15      @Inject
16      PersonService personServiceEjb;
17
18      @Override
19      protected void doGet(HttpServletRequest request, HttpServletResponse response)
20          throws ServletException, IOException {
21          List<Person> people = personServiceEjb.listPeople();
22          System.out.println("people:" + people);
23          request.setAttribute("people", people);
24          request.getRequestDispatcher("/listPeople.jsp").forward(request, response);
25      }
26  }
  
```

www.globalmentoring.com.mx

As of today, integration between different business technologies is simpler each time.

As we can see in the following code, integrating an EJB to be used in a Servlet is as simple as using the @EJB annotation and specifying the type of EJB to be used.

This will cause the application server to automatically search for an instance of the specified EJB type, and once it is localized, this dependency is automatically injected by the Java server.

```

@WebServlet("/people")
public class PersonServlet extends HttpServlet {

    @Inject
    PersonService personServiceEjb;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        List<Person> people = personServiceEjb.listPeople();
        System.out.println("people:" + people);
        request.setAttribute("people", people);
        request.getRequestDispatcher("/listPeople.jsp").forward(request, response);
    }
}
  
```



# ONLINE COURSE

# JAVA EE JAKARTA EE

By: Eng. Ubaldo Acosta



**JAVA EE COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)