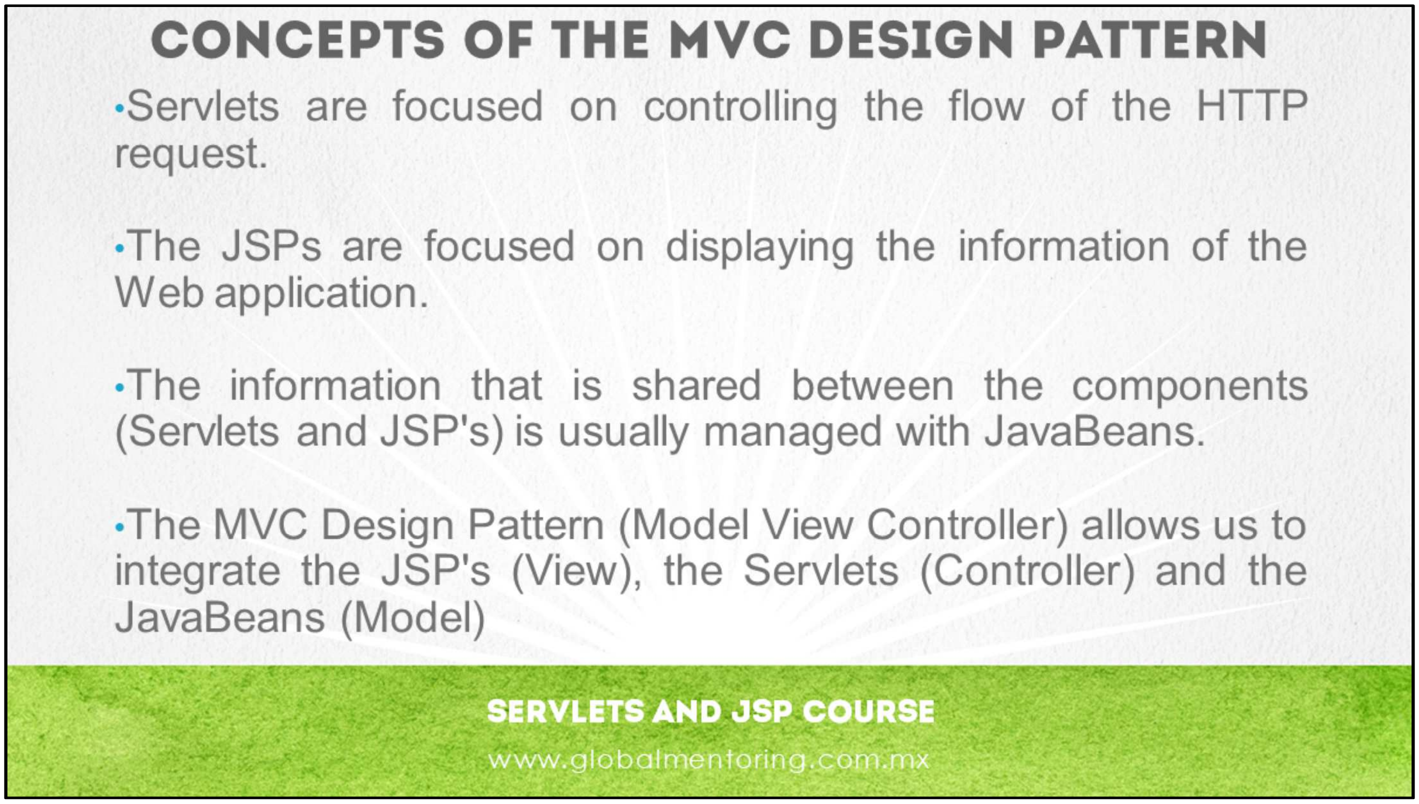
SERVLETS AND JSP COURSE
MVC DESIGN PATTERN
By the expert: Eng. Ubaldo Acosta
Eng. Ubaldo Acosta
Global Mentoring
JAVA
ONLINE COURSES
UNIVERSITY
SERVLETS AND JSP COURSE
www.globalmentoring.com.mx

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

We are going to study the Model View Controller design pattern, also known as MVC.

Are you ready? OK let's go!



CONCEPTS OF THE MVC DESIGN PATTERN

- Servlets are focused on controlling the flow of the HTTP request.
- The JSPs are focused on displaying the information of the Web application.
- The information that is shared between the components (Servlets and JSP's) is usually managed with JavaBeans.
- The MVC Design Pattern (Model View Controller) allows us to integrate the JSP's (View), the Servlets (Controller) and the JavaBeans (Model)

SERVLETS AND JSP COURSE

www.globalmentoring.com.mx

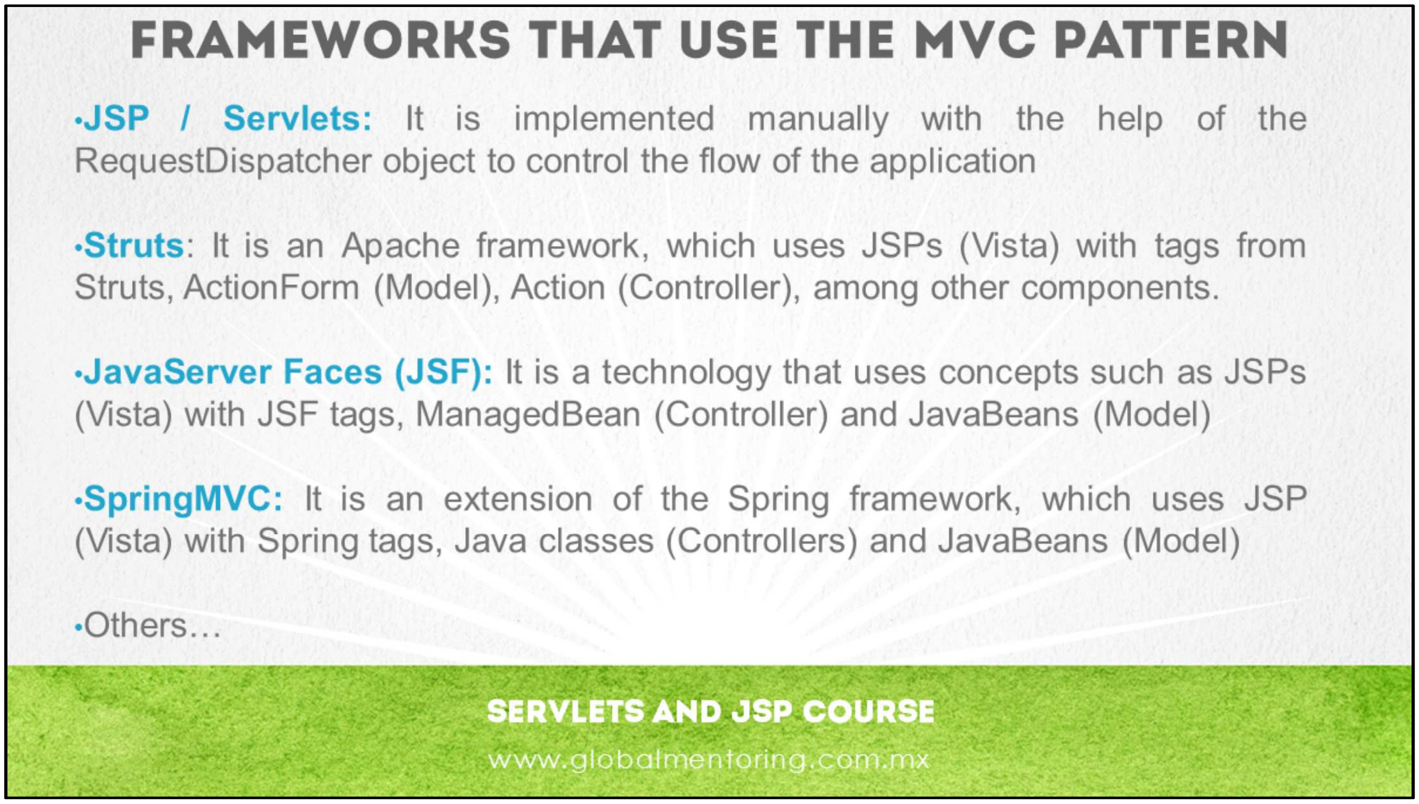
In this lesson we will review the concepts of the design pattern in Model-View-Controller, also known as MVC.

A design pattern allows us to solve common problems that arise when creating applications, and in particular in Web applications we are interested in separating the view of the data (model) and joining them by means of a component that acts as a controller.

As we have studied so far the Servlets are focused on controlling the flow of the application and in this case they process the HTTP requests, as well as use the JavaBean to store information and finally redirect to the respective JSP.

The JSPs are focused on displaying the information of the Web application, in this case the information is provided through the Servlets and the information that is shared between these components, that is, between Servlets and JSPs, it is usually managed with JavaBeans.

In summary, the model pattern MVC model-view-controller allows us to integrate JSPs, Servlets and JavaBean in a single model to interact and thus create web applications increasingly robust and easier to maintain.



FRAMEWORKS THAT USE THE MVC PATTERN

- **JSP / Servlets:** It is implemented manually with the help of the RequestDispatcher object to control the flow of the application
- **Struts**: It is an Apache framework, which uses JSPs (Vista) with tags from Struts, ActionForm (Model), Action (Controller), among other components.
- **JavaServer Faces (JSF):** It is a technology that uses concepts such as JSPs (Vista) with JSF tags, ManagedBean (Controller) and JavaBeans (Model)
- **SpringMVC:** It is an extension of the Spring framework, which uses JSP (Vista) with Spring tags, Java classes (Controllers) and JavaBeans (Model)
- Others…

SERVLETS AND JSP COURSE

www.globalmentoring.com.mx

Let's discuss some Frameworks that use the MVC design pattern.

The MVC design pattern can be implemented manually using JSPs and Servlets and with the help of the RequestDispatcher object we will be able to control the flow of our application.

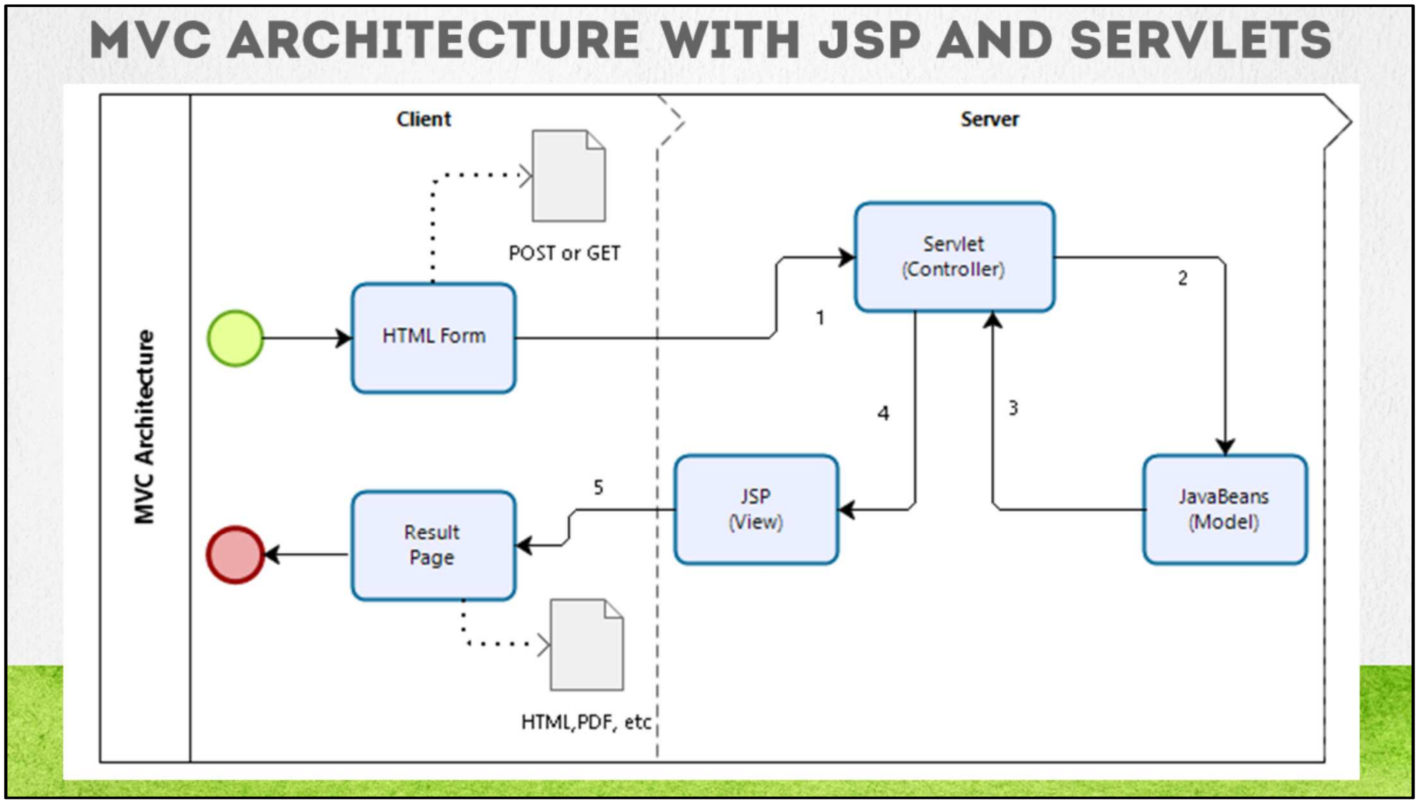
There are several Frameworks that already implement this pattern, a design pattern is simply a guide, therefore each of these Frameworks, Struts, JavaServer Faces, Spring MVC, among others, only follow this guide or recommendations but it is not really a specification that tells us step by step how to implement this pattern, but simply is a series of generic steps with which each of these Frameworks implements according to the best practices from the point of view of each of these Frameworks.

For example, in the case of the Struts Framework, it is an Apache framework, which uses JSPs as the View, also using Struts tags, and then uses the concept called ActionForm that in some way replaces the JavaBeans, being the model of our application and finally we have the concept of Action, which covers the role of the controller. These are simply some of the components that are managed within the Struts Frameworks.

JavaServer Faces is a technology in which concepts such as JSPs are used but using JSF tags, although they also have their own presentation technology known as Facelets. The concept of ManagedBean is also used, which covers the role of the controller and finally we have the concept of JavaBeans to manage the concept of the model although it is also worth noting that ManagedBean can play the role of both controller and model, then everything could be mixed in a single bean and the use of JavaBeans could be omitted.

Finally we will mention the Spring MVC framework. This framework is an extension of Spring, in which JSPs are used as part of the view and Spring tags can be used to strengthen these JSPs. Subsequently, Java classes, also known as POJOs (Plain Old Java Objects) or pure Java classes, are used as controllers and JavaBeans are also used as part of the model.

There are many more frameworks that implement the MVC pattern using the Java language, these are just some of the most representative and as we have only commented the design pattern is a general guide and each framework defines the specification according to the best practices from the point of view. view of each framework.



Let's now comment on the MVC architecture using JSPs and Servlets.

As we can see, the flow starts with an HTML form, this information is stored in our client and on the right side we will have the server. Once our client sends the form request to the web server, who is going to process this request is a Servlet (controller) and although in previous years we have placed a JSP to directly handle forms, as we have said, this is not Good practice. Therefore, the requests must be processed by a controller Servlet, we can even observe the numbers with which we are going to follow our flow (1 to 5).

Once the controller Servlet has received the request, one of its tasks may be to process the parameters of the HTML form if it applies and once we have processed the parameters of the form, normally what we do is to support JavaBeans to store or process the business logic information or presentation logic of our Web application.

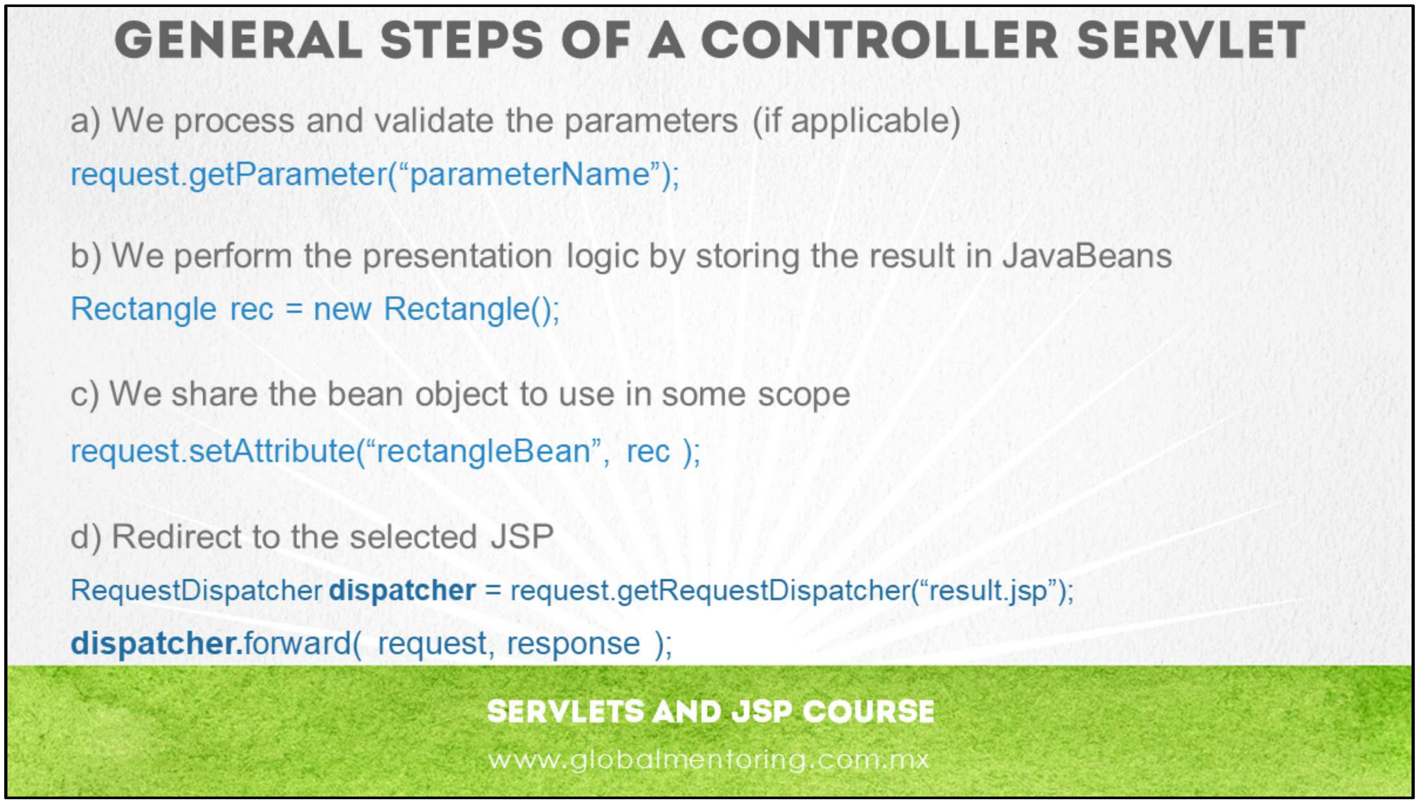
Since we have created and stored the information in our JavaBeans, we return the control to the Servlet and this Servlet controller can place these JavaBean in some scope to share information to a JSP. Scopes can be request, session or application. Servlets do not know the scope of page, since that belongs only to the JSPs, therefore we can only handle the 3 scopes mentioned.

Once the Servlet Controller has already placed the JavaBeans in some scope, it does a redirect through the RequestDispatcher object that we will see later. At this point it is also worth mentioning that the Controller Servlet makes the decision of which JSP will be used, for example we could have a JSP1 a JSP2 a JSP3 etc. everything will depend on the flow of the Web application, but in this case the Controller Servlet is the one who will decide which one is Vista or JSP is the one that is going to be used.

Finally, once we are inside the selected JSP, what the JSP will do is play the role of the Vista, this implies that it will only show the information that the Servlet shared. The JSPs should not theoretically create new Java objects, for this all the information that the JSP will use must have already been provided by the controller Servlet.

Once the JSP generates the HTML using the information of the JavaBeans that the Servlets provided, what it does is to return the content to the client and at this moment it is when the Render of our application is generated according to the Content Type that we have used. For example, it can be an output in HTML, PDF, Video, an Excel file, etc. as we have seen previously.

The point is that the JSP will only display the information it received from the Servlet and will send this information to the client. With this the flow ends and if the client needs to make a new request the process is repeated again.



GENERAL STEPS OF A CONTROLLER SERVLET

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

```
request.getParameter("parameterName");
```

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

```
Rectangle rec = new Rectangle();
```

c) We share the bean object to use in some scope

```
request.setAttribute("rectangleBean", rec );
```

d) Redirect to the selected JSP

```
RequestDispatcher dispatcher = request.getRequestDispatcher("result.jsp");
dispatcher.forward( request, response );
```

SERVLETS AND JSP COURSE

www.globalmentoring.com.mx

Finally, we are going to comment on the broadly needed code that a Servlet controller uses.

As we review in the Servlet Theory, to process a parameter we can use the following notation: request.getParameter ("parameterName"); we can use the request object and the getParameter method indicating the name of the parameter we want to process.

Afterwards, we can validate the parameters to know if the information we are receiving is correct. Once we have processed the parameters we can perform the respective presentation logic or business logic using JavaBeans. The result of processing this information will be stored in JavaBean objects, in this case we are simply creating a new object of type Rectangle Rectangle rec = new Rectangle (). We are only creating a new JavaBean to try to exemplify that these JavaBeans are going to have the information of our Web application.

Afterwards, before redirecting and selecting the JSP that we are going to use, we must share the object that we are creating in some scope, in this case we are selecting the scope of the request, but we can use the scopes of the session or the servlet context. In this case we will use the setAttribute method of the request object and indicate the name of the bean that will be recovered by the JSP.

Finally we will select the JSP that will display the information to the client, by means of the code:

RequestDispatcher dispatcher = request.getRequestDispatcher ("result.jsp"); what we do is select the JSP by relying on the request object and the getRequestDispatcher method, here what we are going to do is provide the JSP that we are selecting as the view that will display the information we are sharing through the request object and once we have selected the JSP that we are going to use, we assign it to an object called RequestDispatcher. This is the object that we have been discussing and that will allow us to redirect towards the respective JSP.

Finally we use the method dispatcher.forward (request, response); what we do is use the dispatcher object to execute the forward method. The important thing here is that we can see that we have 2 arguments, the request argument and the response argument. If we remember, once we are processing a method of a Controller Servlet, for example if we have the method doGet or the method doPost, we are receiving as arguments the object object and the object response, in a few words what we are doing here is only to provide the same objects that we have already received in this Servlet, with the intention that the information we have shared in our request, session or application objects continue to be shared with the other elements that we will continue using. In this case, by means of the forward method we are providing and sending all the necessary information to the JSP so that it does not have any problem and can access the information that we have previously shared through the Servlet. Next we will do an exercise to put into practice the concept of the MVC design pattern.

ONLINE COURSE
SERVLETS & JSP´S
By: Eng. Ubaldo Acosta
Global Mentoring
SERVLETS AND JSP COURSE
www.globalmentoring.com.mx