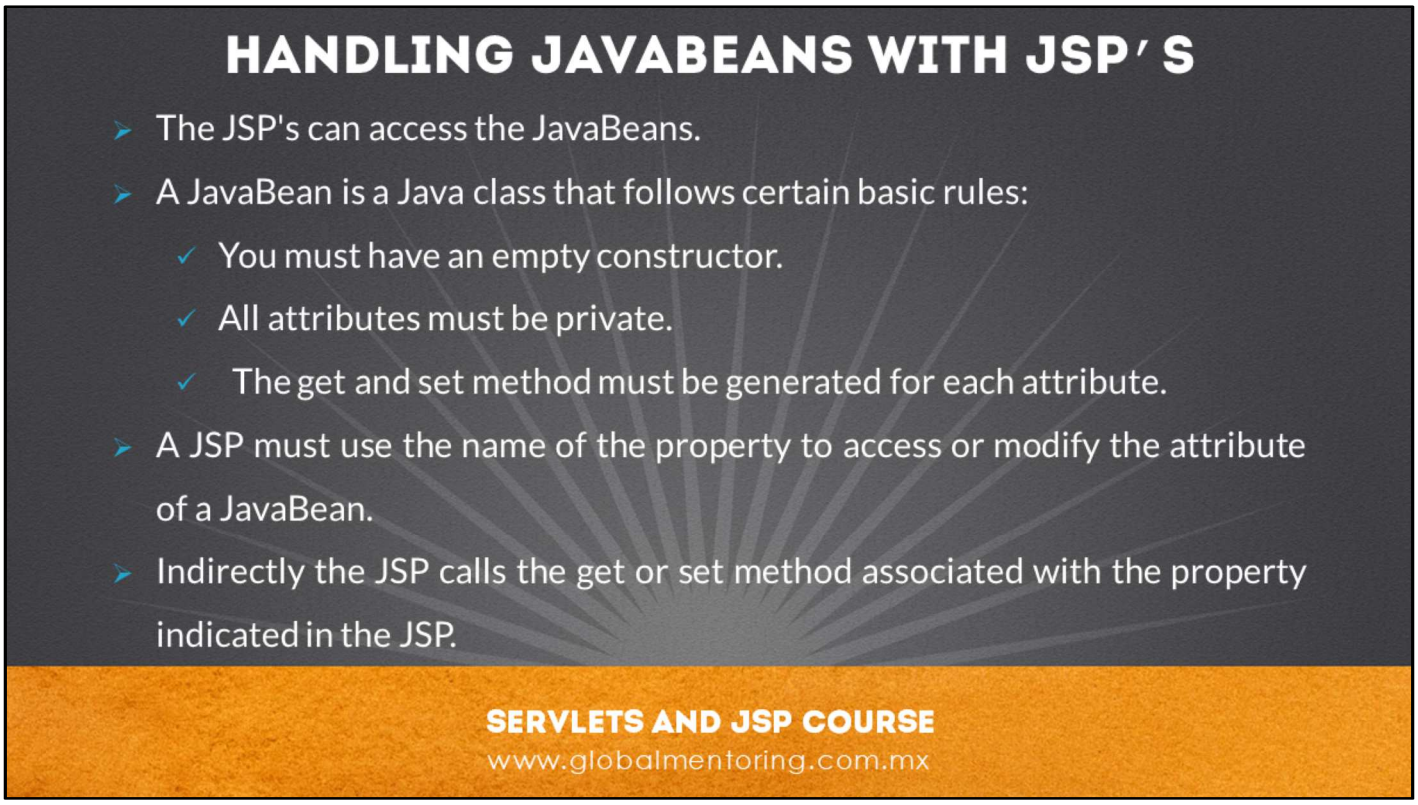
Hello, Ubaldo Acosta greets you again. I hope you're ready to start with this lesson.

We are going to study the topic of Handling JavaBeans with JSPs.

Are you ready? Come on!



We are going to study next the topic of JavaBeans management using our JSPs.

The JSP's will allow us to access the JavaBeans that we have declared in our web application. A JavaBean is simply a pure Java class that follows certain rules. One of the rules is that it must have an empty constructor and that restriction is for the following, if we have a class that does not have an empty constructor then we must have a constructor with a certain number of parameters, so when the JSP tries to access This JavaBean must know how many parameters must be specified in order to create a JavaBean object stay, and this implies greater complexity when instantiating a Java object.

Therefore, JavaBeans require an empty constructor, so there is no need to indicate which are the arguments needed to instantiate a JavaBean class.

In addition to having an empty constructor, JavaBeans must have private attributes and for each of the declared attributes we must add their get and set method for each attribute. We will see later that there are certain exceptions, for example, if we only want to access the property but we do not intend to modify the property, then we could have only its get method and vice versa, in case we only want to modify the property we add the respective set method and we can ignore its get method.

A JSP must use the name of the JavaBean property. Indirectly the JSP when placing the name of the property is called to call the respective get or set method as we indicate in the action that we use in our JSP. Let's review later how to make this code.

EXAMPLES OF JAVABEANS PROPERTIES

Property Name	Name of the Methods	Code in the JSP
userName (String)	getUserName setUserName	<jsp:getProperty … property="userName" /> <jsp:setProperty … property="userName" />
deleted (boolean)	isDeleted setDeleted	<jsp:getProperty … property="deleted" /> <jsp:setProperty … property="deleted" />
address	getAddress setAddress	<jsp:getProperty … property="address" /> <jsp:setProperty … property="address" />
zip_code	getZip_code setZip_code	<jsp:getProperty … property="zip_code" /> <jsp:setProperty … property="zip_code" />

SERVLETS AND JSP COURSE
www.globalmentoring.com.mx

We can observe the following examples, we have the name of the property, the name of the associated methods get and set and the code that we are using within the JSP.

For example, if we have a property named username, we can see that its get method must be written as follows get UserName. We are going to use the notation of highs and lows, or camel notation, that is, the n becomes capitalized and the letter u is also converted to upper case, because we are separating each word from the name of the method.

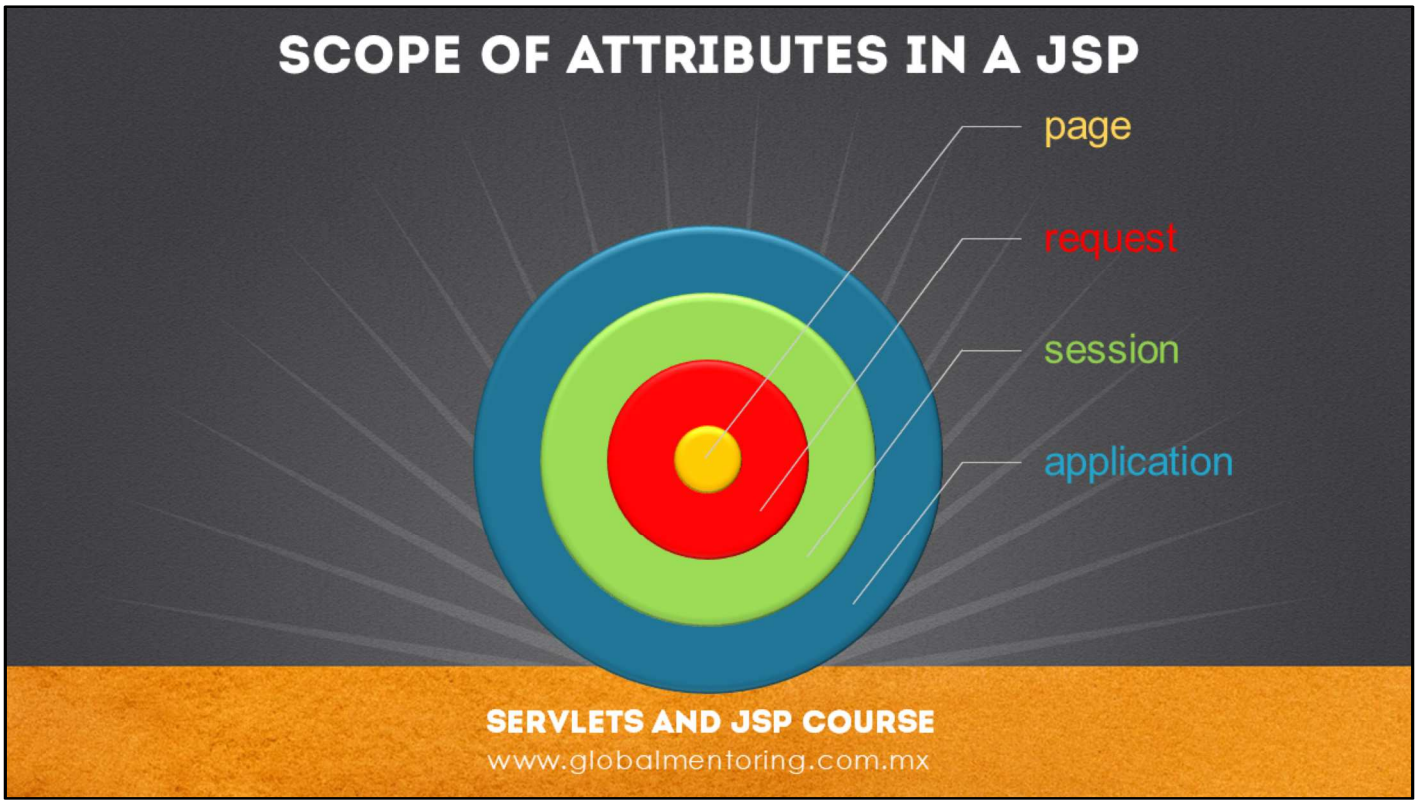
This is the Java notation that we should use when we create the respective get and set method of a property.

Once we have the get and set methods inside the JavaBean class, inside the JSP what we have to do is use the following action of the JSPs, we will write jsp: and depending on the action we want to do it will be getProperty or setProperty. Then we indicate the name of the property to be used, (eg username) and indirectly the name of the method associated with the respective action will be executed, eg. get UserName () and if we are using the setProperty action, the setName UserName () method is called. That is why we say that in the JSP we are going to use the name of the property and indirectly it is going to call the associated method depending on the action that we are specifying.

Let's review some other examples. The deleted property is of boolean type and for boolean type properties instead of having a get method we are going to convert it by an is method, (isEliminated). The set method does not change because to modify it the same annotation is still used (setEliminated) and in the code of the JSP it does not change, we will use its get method that indirectly will call the is method which is the of reading, or the action of setProperty and we indicate the property and the setEliminated method is indirectly called. Similarly, it should be noted that the property is in lowercase and the method that is sent to call is setEliminated and the E starts with a capital letter.

Now we are going to see an interesting case in which many times we can use it to make method names that are not necessarily going to map to a JavaBean property, since by respecting the nomenclature we can send to call methods as we can observe. The property noTelefono, we can create methods called getTelefono and setTelefono and the actions of the JSP must correspond not to the name of the property, the may not even exist, but to the name of the methods eliminating the word get or set and converting the first letter in lowercase , with this you can access methods that do not even have a property mapped, but possibly methods that perform calculations within the JavaBean.

And finally, we see another annotation of a property of a JavaBean, in this case we are using the notation that is not very common even is not recommended, but sometimes there is already code created in this way getCodigo_postal. What we have to do then is to put the name of the property and indirectly we will send it to call the getCodigo_postal method in this order if we have a low script (_). Then we do not have to do anything to respect, if not simply the only one that must be converted into uppercase is the first letter, therefore the method that is sent to execute is getCode_postal and the p is still lowercase.



We are going to review next the topic of scope of the variables in the JSPs.

Basically the scope of a variable is the duration or time of use of a variable in a web application. We can see that we have four scopes or in English scope. As we have reviewed previously, we have used the request object to add certain information and we have also used the session object to share information that can last longer than a request, that is, a session can manage several requests (request) and therefore the information that we store in a session lasts longer than the time we store it at the level of the request object.

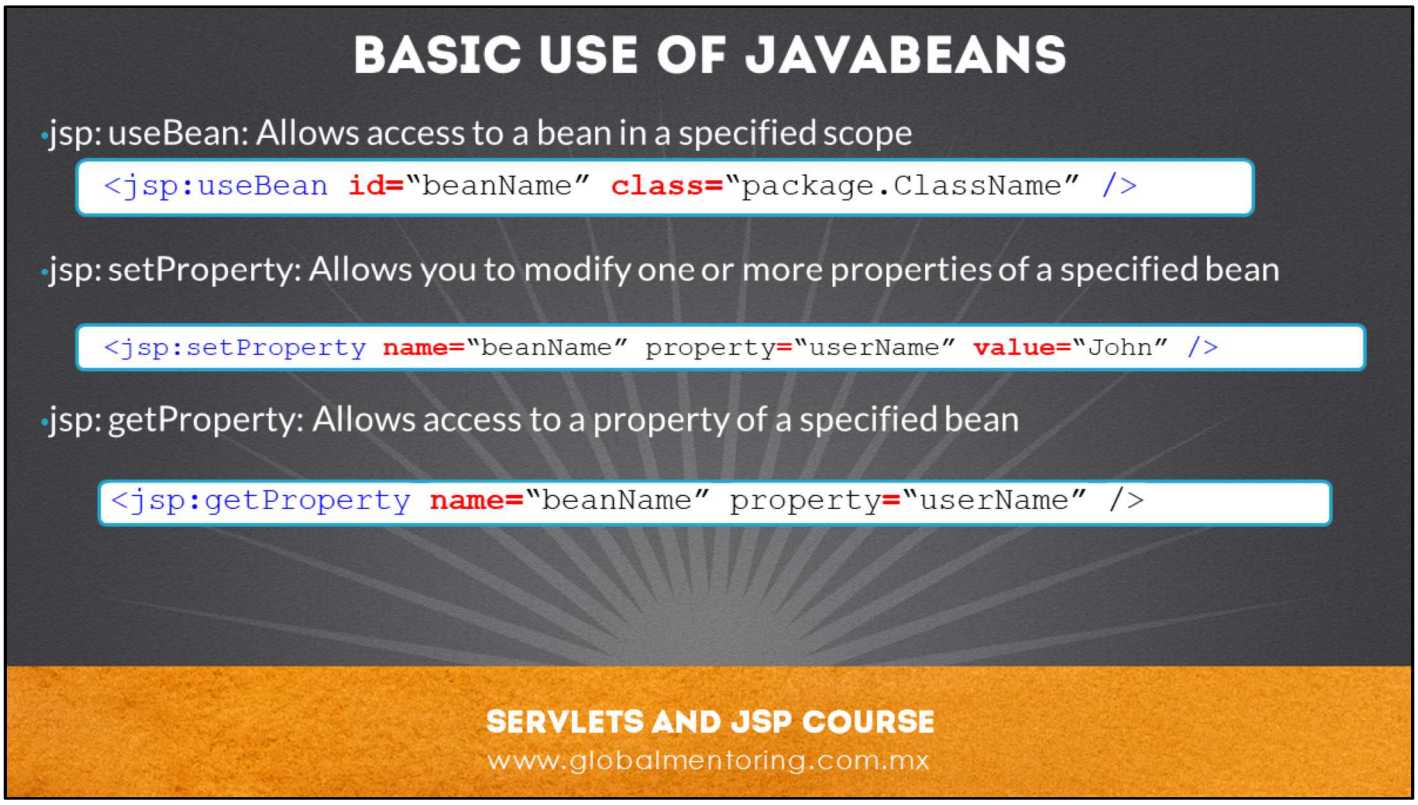
We observe in the figure that we have two more scopes and these are all the scopes that we have to share information between the different components whether they are Servlets or a JSPs.

As we can see the page scope is the lower one. This scope will only last during the translation time of the JSPs but it will not even last during the time of the request of our request, therefore the information that is stored in this page scope will only be able to be accessed in the JSP that is doing the respective translation at that time.

The next scope is request. As we can see, the request scope has a longer duration than the page scope. With the scope of request the information that we store will exist during the entire time of our web request, this means since the client initiates the request until the server returns the response to our client will be available in the web application. Therefore, the request scope is longer than the page scope.

The next scope we have is the session, we can see that the session will last longer than request and page. The information that we store in a session will allow us to save data between different requests and thus share information between different web components such as different JSPs and different Servlets. The request scope will also allow us to share information between different JSPs and Servlets components but only during the time a request lasts, instead the session allows us to do something similar but the time that the information stored in a session lasts is longer and therefore more components will be able to make use of the information that is stored in a session.

Finally, we have the scope of application, this is equivalent to handle the context of ServletContext, this concept is handled in the topic of Servlets, then what is for a servlet the scope of ServletContext in a JSP is called application. Each of these variables are implicit variables that we have in a JSP, the scope of the application is the largest scope we can have and the information that we store in that scope, for example with the setAtribute method, will last for the entire time in that this our application is up. For example, if we are using an application server and add information to this scope will last as long as the server is up and until we stop the server or download our application is when it will destroy the information we have stored in the scope application.



Finally, we will review the basic use of Java Beans from the JSPs. We can use the following actions in the JSPs:

- We have the useBean action, this will allow us to access a bean in a specified range. We can or can not specify the scope, if we do not specify it within our syntax of jsp: useBean, the scope by default is of type page. On the other hand if we want to indicate some other scope we have to put scope = request, session or application. The syntax we are going to use is similar to this <jsp: useBean id = "name" class = "package.NameClass" />. With this syntax in a few words it is as if we create a new instance and the name Bean is the name of our variable and the package. NameClass is the type of the class that we are indicating to generate this new object nameBean.

- Later we have the action of setProperty. This action allows us to modify one or several properties of a specified bean. For this action we specify the name of the bean previously defined with the action jsp: useBean, later we specify the property to be modified, followed by the value of it. Eg property = "username", this will indirectly call the setNameName method of the respective JavaBean.

- Finally we have the action of getProperty. This action allows us to access a property of the specified bean. We use the action jsp: getProperty, later we indicate the name of the bean nameBean that we are going to use, that is to say, the name of the instance previously declared with the action jsp: useBean. Later we indicate a property to access, eg. property = "userName", indirectly the getNameName () method will be called.

These are the basic actions that we are going to use with JavaBeans, there are several combinations that we can use when creating and using JavaBeans, so we are going to see some examples below to go detailing the basic use of JavaBeans in JSPs.