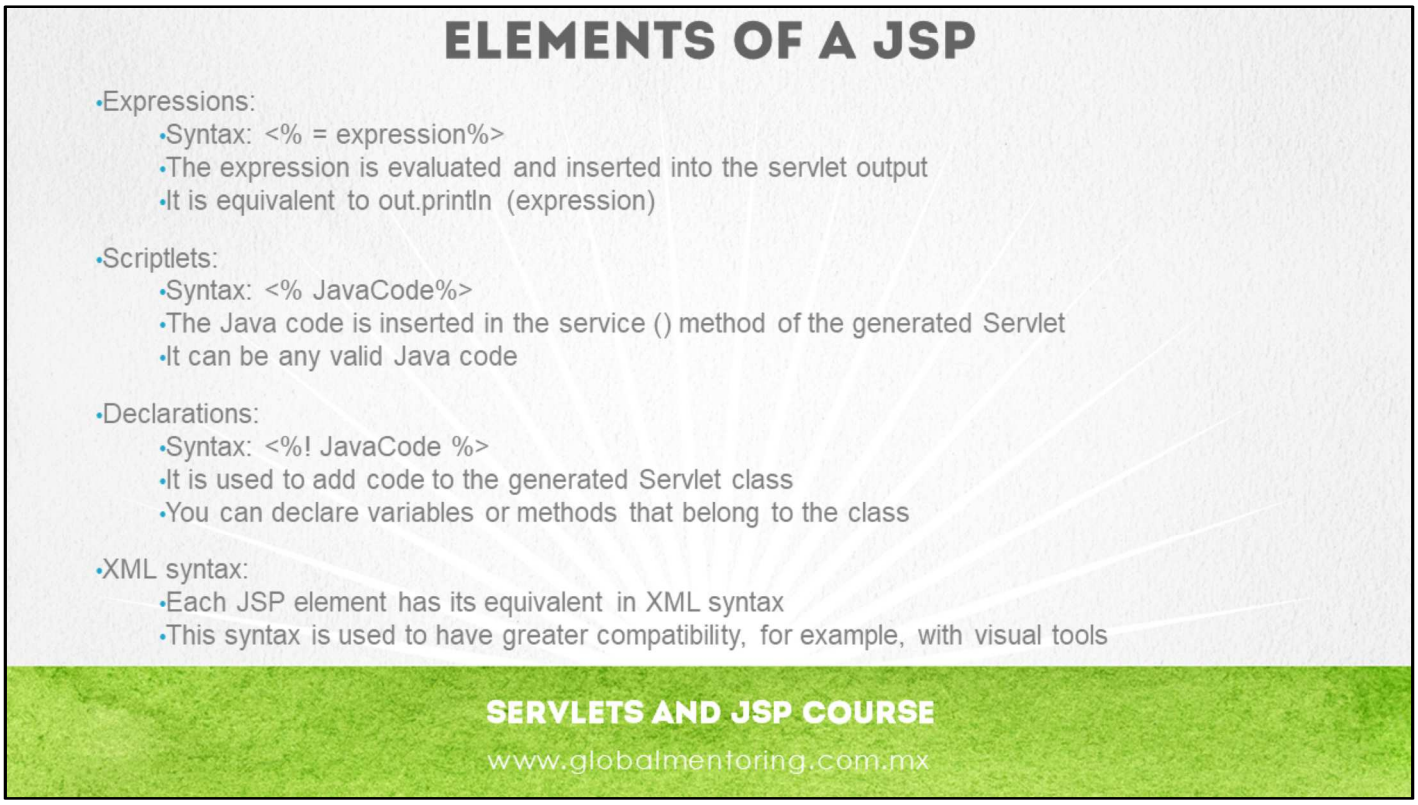

Hello, Ubaldo Acosta greets you. Welcome again. I hope you're ready to start with this lesson.

We are going to study the elements of a JSP.

Are you ready? OK let's go!




We are going to review the basic elements of a JSP. A JSP is composed of Expressions, Scriptlets, Declarations and there is also a feature in which we will handle an equivalent syntax in XML.

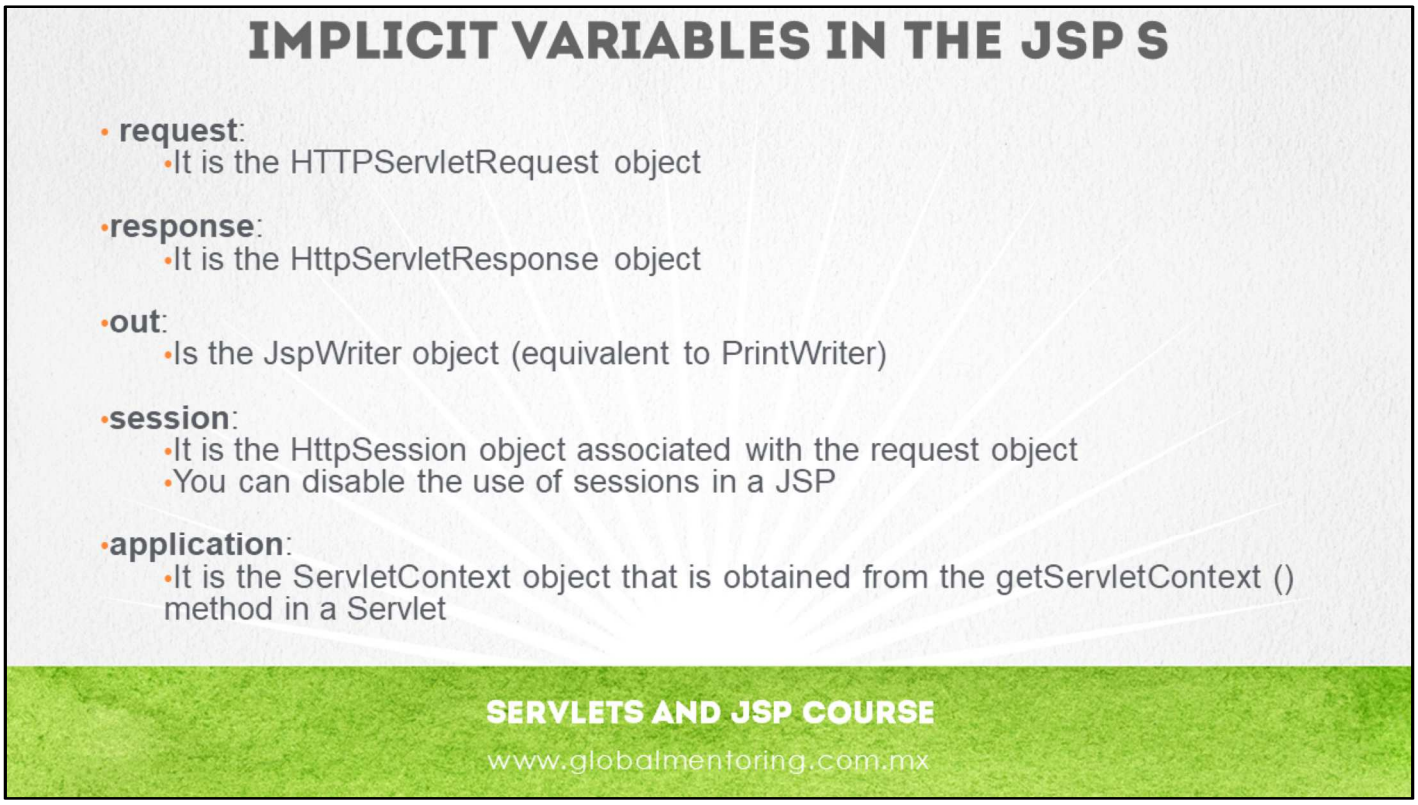
An Expression has the following syntax: Syntax <% = expression%> What we are going to do is use the syntax <% = to open our expression, later we add an expression that can be any valid expression that at the end of accounts is going to be equivalent to put the syntax out.println and what we have in our expression will send it to the client. Finally we close the expression with%> This type of syntax <% = expression%> is known as an expression tag or tag. The expression can be a value such as a string, a mathematical expression, a sum, a subtraction etc. or it can also be the result of a call to a function, but if we send a function to call the result of this function can not be void, it must return some value since as we can see the result of evaluating this expression goes to send our client then for that reason this expression must return some result.

Another element of the JSPs are the Scriptlets. A Scriptlet contains Java code as opposed to an expression that we only use to print information to our client, a Scriptlets can contain Java code that our client will not necessarily visualize because if this Java code does not send anything to the client, then our client is not going to visualize the information that is processing. This Java code can be quite robust and therefore we must be careful about the code we handle with the Scriptlets, because although they allow us to add a lot of code and can be very powerful to add Java code that can also be counterproductive especially for maintenance of a JSPs, so too much code in a scriptlet is considered a bad practice.

The syntax of a Scriptlet is <% java_code%> unlike an expression does not have an equal sign. It starts the same way with <% and the closing for the tag of a Scriptlets is the same, then the syntax is: <% codeJava%> It is important to mention that the code generated within a Scriptlets is inserted inside the service () method of the servlet generated from the JSP that has been compiled, later we will see some examples to observe and understand this paragraph. Scriptlets can contain any valid Java code, that is, they can have variable declarations, function calls and in general can have any logic as long as we respect that this code is inside another method, in this case the service () method.

In the JSPs we will also handle the declaration tags and the syntax is: <%! JavaCode %>. In the statements is also Java code, but this code we will use to declare either variables or methods that belong to the class of the servlet generated, and unlike the Java code that is in a Scriptlets in which all the variables that we declare are local to the service () method, in the case of declarations the code that we add, if we declare a variable, it becomes an instance variable because it is being added as a variable of the generated servlet class and not only as a local variable to a certain method. Also the methods that we add become methods that are part of the servlet class generated although this concept of declarations is not very used since the most common is that from a Scriptlets we send to call functions or use variables that are defined in other classes and not within our JSP, then in the case of the declarations we will use it as long as we need to add variables or methods that belong to our servlet generated from the JSP.

Finally, each of our components, such as expressions, scriptlets and declarations have their equivalent in XML syntax. This we will see that it has several applications in the sense that we will have greater compatibility if our JSPs are generated with XML syntax and not with the standard syntax. When we review the topic of XML syntax we will see the equivalences for each of these elements.




We will now review the issue of implicit variables in a JSP within the use of Scriptlets.

Within a JSP we can use objects that are already instantiated so that we can use them immediately. For example we have the objects request, response, out, the object session, application among others. The request object is the same object that we worked for example in Servlets when we overwrote the doGet or doPost method, we receive both the request object and the response object as parameters. Then we have these objects available in a JSP and we do not have to instantiate them, we just put the name of our variable and through the operator point (.) We can access the attributes and methods of the object of type HTTPServletRequest. The same happens with the response method, simply with this variable name we can access the attributes and methods of this HTTPServletResponse object.

The variable out is equivalent to the PrintWriter object that we have instantiated inside the Servlets, in this case there is a small variant since this out variable is of type JspWriter.

The variable session is a variable of type HttpSession and with this variable we can obtain information that we have already added to our session, for example by means of a Servlet. We will see later by means of a concept called directives that we can disable the access to the session in case we do not want the JSPs to directly manipulate the attributes of a session or simply that we do not want a JSP to access the attributes or modify the elements of a session.

We also have the variable called application, this is equivalent to the ServletContext object, which can be obtained in a servlet from the getServletContext () method. This concept we will see later in more detail.

In summary, there are variables already instantiated that we can access directly in a JSP. Let's review some exercises below to review these basic concepts of the JSP's.