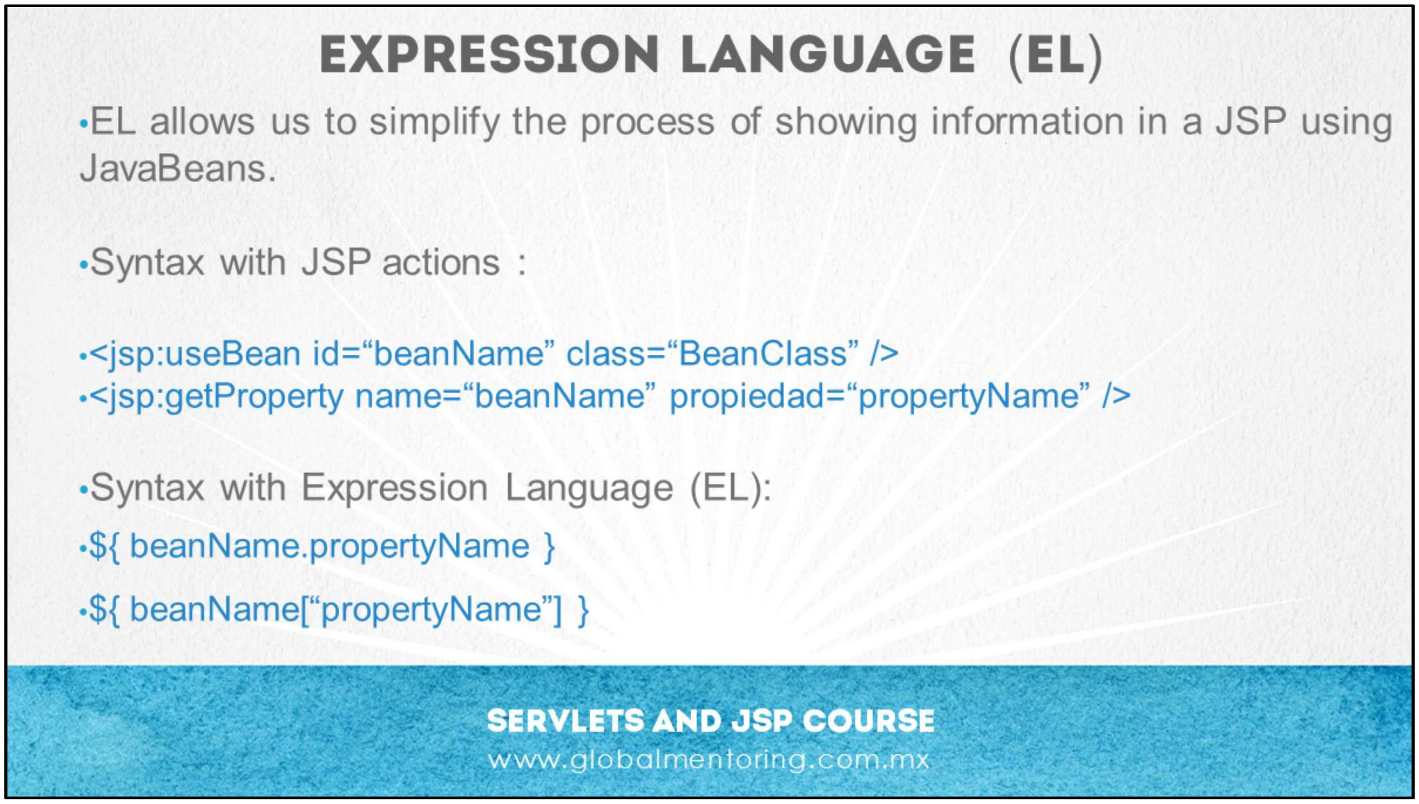
SERVLETS AND JSP COURSE
EXPRESSION LANGUAGE WITH JSP' S
By the expert: Eng. Ubaldo Acosta
Eng. Ubaldo Acosta
Global Mentoring
JAVA ONLINE COURSES UNIVERSITY
SERVLETS AND JSP COURSE
www.globalmentoring.com.mx

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

We are going to study the theme of Expression Language with JSP's.

Are you ready? Come on!



EXPRESSION LANGUAGE (EL)

- EL allows us to simplify the process of showing information in a JSP using JavaBeans.
- Syntax with JSP actions :
- <jsp:useBean id="beanName" class="BeanClass" />
- <jsp:getProperty name="beanName" propiedad="propertyName" />
- Syntax with Expression Language (EL):
- ${ beanName.propertyName }
- ${ beanName["propertyName"] }

SERVLETS AND JSP COURSE
www.globalmentoring.com.mx

Next we are going to review the concept of Expression Language (EL).

The objective of this language is that it allows us to simplify the information that we are displaying in the JSPs. Normally this type of information deployment we will rely on JavaBeans, as we have already seen, to display information stored in a Web application.

In the sheet we can review a comparison between the syntax using the JavaBeans actions and the Expression Language syntax.

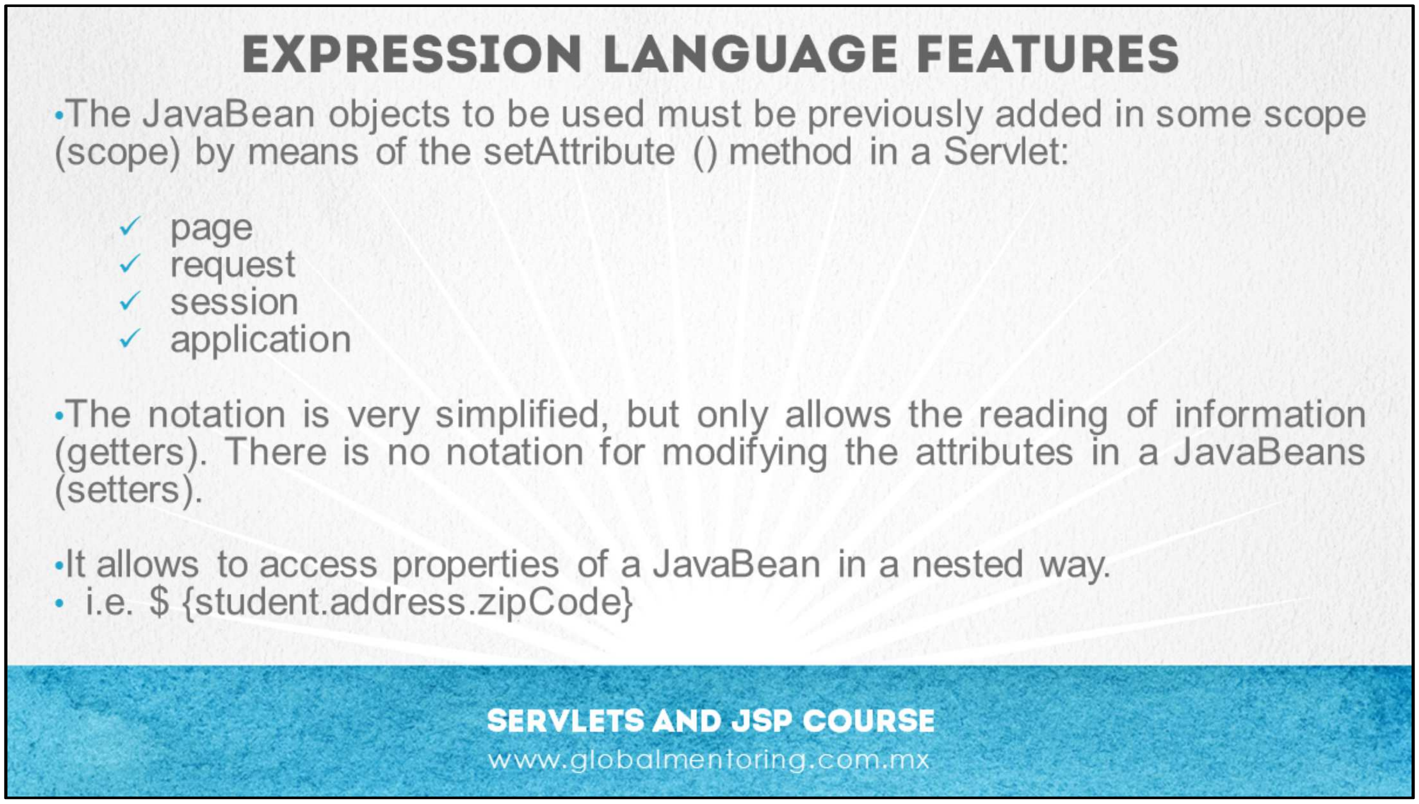
As we can see the syntax of the JavaBeans as we have seen we have to declare the name of the bean that we are going to use within the JSP and then access the property, this syntax can be very extensive if we are doing a lot of code, for example to access a property with JavaBean actions:

```
<jsp:useBean id="beanName" class="BeanClass" />
<jsp:getProperty name="beanName" propiedad="propertyName" />
```

On the other hand, with the Expression Language syntax, we only need to specify the name of the bean that we are going to use and the name of the property we want to access, the most common way to do it is: $ {beanName.propertyName}, but also we have another form $ {beanName["propertyName"]}.

Internally the expression language (EL) what it does is store its attributes or beans as if it were a map, so we can specify the name of the property or we can also specify it in brackets, since it can function as an index to access the map and subsequently recover the respective value of the property.

Any of these two annotations will allow us to access the JavaBean and JavaBean properties using Expression Language and how we can see that the syntax is greatly simplified if we compare it with the concept of JavaBeans actions.



EXPRESSION LANGUAGE FEATURES

- The JavaBean objects to be used must be previously added in some scope (scope) by means of the setAttribute () method in a Servlet:
  - ✓ page
  - ✓ request
  - ✓ session
  - ✓ application
- The notation is very simplified, but only allows the reading of information (getters). There is no notation for modifying the attributes in a JavaBeans (setters).
- It allows to access properties of a JavaBean in a nested way.
- i.e. $ {student.address.zipCode}

SERVLETS AND JSP COURSE
www.globalmentoring.com.mx

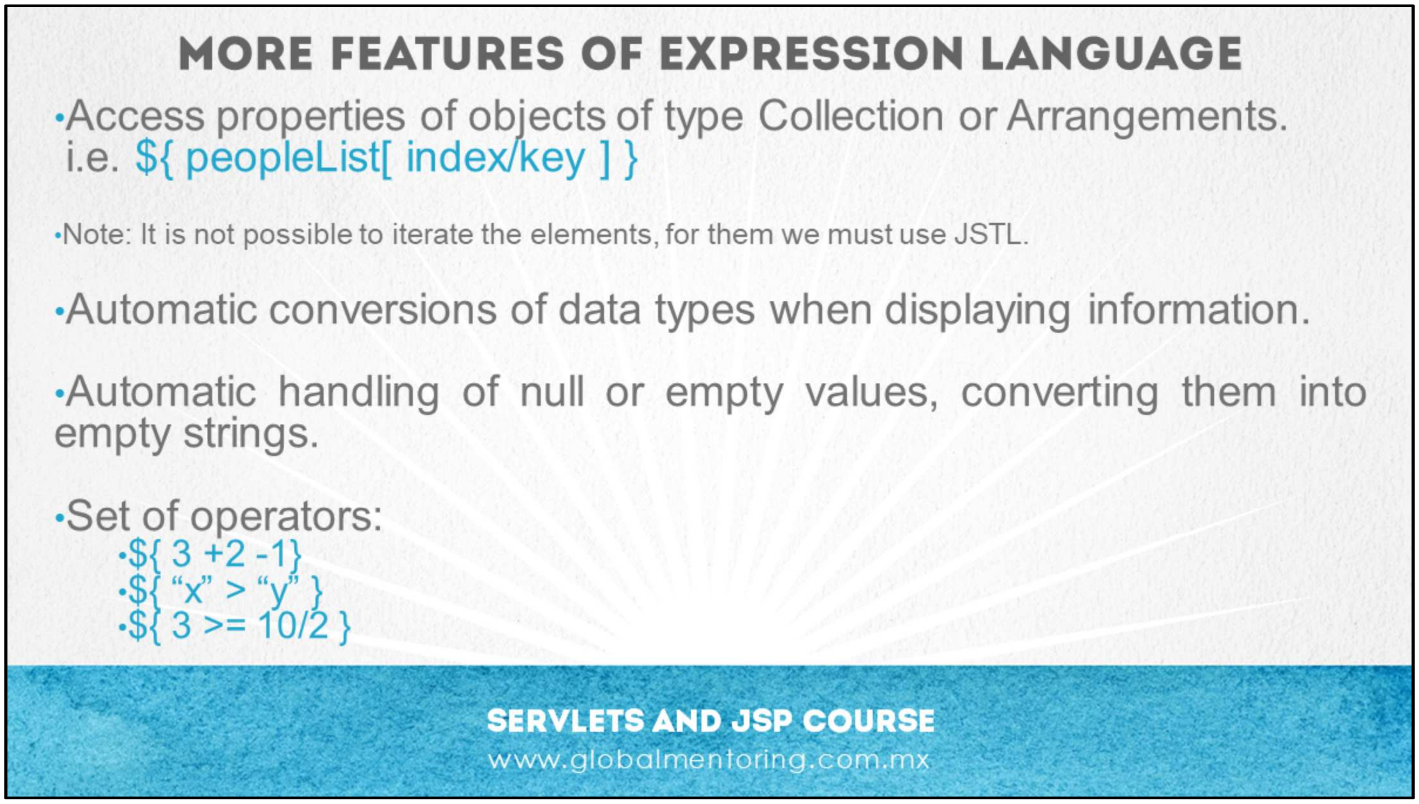
We will now review some of the characteristics of the expression language of the JSPs.

An important detail to make use of this language is that JavaBeans objects to be used must already be previously added in some scope, for example, using the setAttribute() method in a Servlet. If we remember the scopes we have in a web application are: page, request, session and application.

So, for this, we can use Expression Language to access an attribute that we have added to our web application, it must be previously stored in any of these scopes.

Another feature of this language is that the notation is very simplified as we saw in the previous sheet, it is simplified to a single line of code, but only allows us to read information but not to modify it in our Java Beans.

The Expression Language will allow us to access the properties of a JavaBean in a nested way, for example, suppose we have a “Student” class and in turn have a property that is another class called “Address” and this class Address has the attribute “zipCode". Then, we can see that we no longer have to instantiate each of these objects to retrieve the information, but in a nested way and through dot notation, we can access our JavaBean called student, later this attribute is sent to call the method getAddress(), which returns an object of type Address and finally calls the getZipCode() method. We will review some exercises later to be able to understand this nested notation in more detail.



MORE FEATURES OF EXPRESSION LANGUAGE

- Access properties of objects of type Collection or Arrangements. i.e. ${ peopleList[index/key] }
- Note: It is not possible to iterate the elements, for them we must use JSTL.
- Automatic conversions of data types when displaying information.
- Automatic handling of null or empty values, converting them into empty strings.
- Set of operators:
  - ${ 3 +2 -1}
  - ${ “x” > “y” }
  - ${ 3 >= 10/2 }

SERVLETS AND JSP COURSE
www.globalmentoring.com.mx

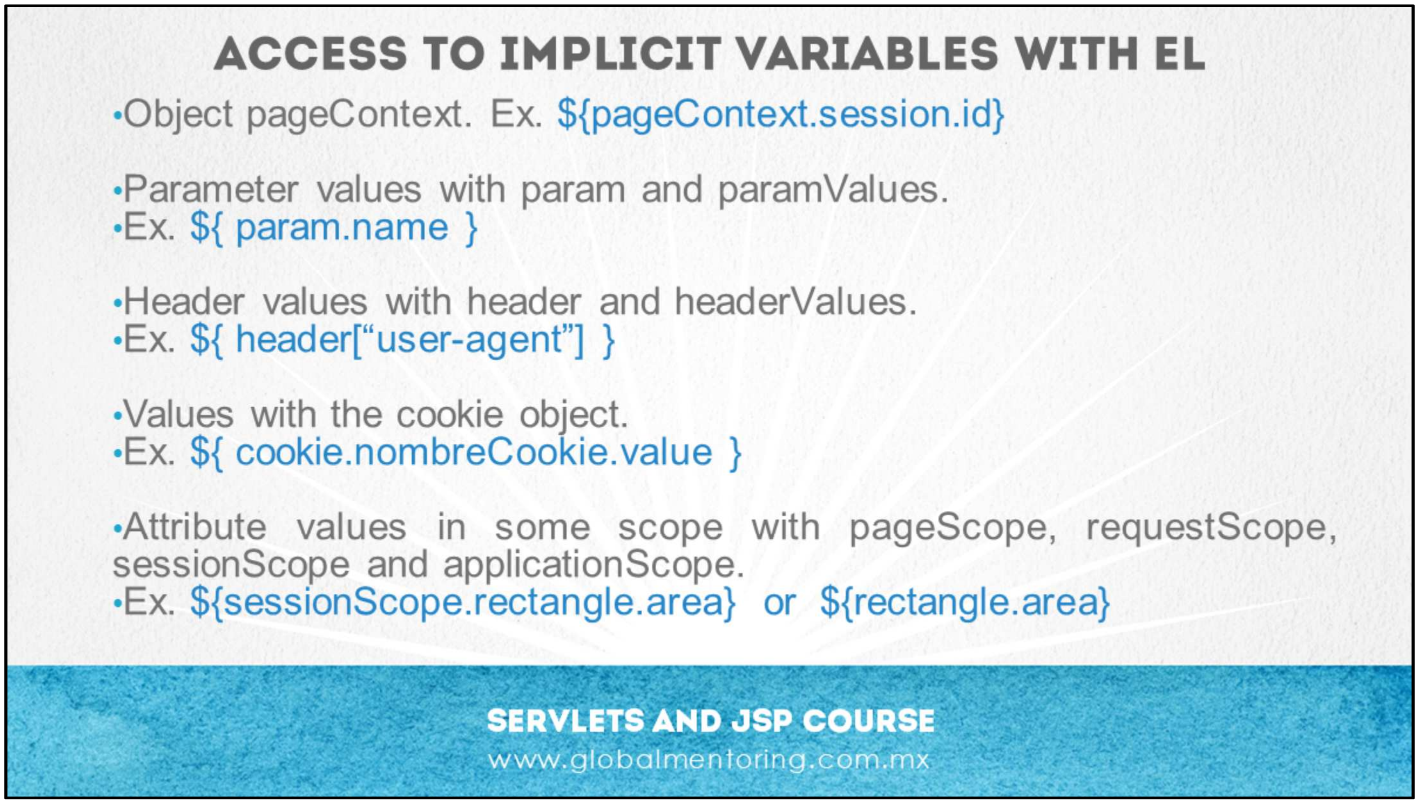
Continuing with the characteristics of this language, we can access properties that are in a collection or in an arrangement. As we have said, simply with the name of the bean we want to access $ {peopleList [index / key]} and between braces we specify the index of the collection to which we want to access or the key.

For example, if the attribute is a map, this notation will allow us to access in a very simple way the elements that we have in a collection. However, it is not possible to iterate the elements. If we want to iterate each of the elements that we have in a collection we must use JSTL (JSP Standard Tag Library) which we will study later.

Another feature we have regarding the Expression Language is that the information is automatically converted to the specified type.

Another characteristic is that the null or empty values are automatically handled by simply converting them into an empty string. This allows us in some way to simplify the handling of exceptions using EL, the detail with this is that if there is an exception for example by a nullPointerExcepcion the JSP will not show this exception on our screen, but that message will only go to send to the standard output and it will not be displayed in the web browser. Then this feature can facilitate the display of information but could also hide important details regarding the handling of exceptions.

Another feature is that this same language has a set of basic operators to do simple operations directly. For example, the syntax $ {3 + 2-1} internally evaluates the expression and gives us a result that is finally what is displayed. This at the end of accounts is equivalent to having an out.println and will display the information contained within this evaluated expression. Also some conversions happen automatically, for example $ {"x"> "and"} the value that has X and the value you have AND are converted to integers and a comparison is made using the elements X and Y. Another example is $ {3> = 10/2} in the same way this expression is evaluated and is equivalent to having the syntax of the out.print method.



ACCESS TO IMPLICIT VARIABLES WITH EL

- Object pageContext. Ex. ${pageContext.session.id}

- Parameter values with param and paramValues.
- Ex. ${ param.name }

- Header values with header and headerValues.
- Ex. ${ header["user-agent"] }

- Values with the cookie object.
- Ex. ${ cookie.nombreCookie.value }

- Attribute values in some scope with pageScope, requestScope, sessionScope and applicationScope.
- Ex. ${sessionScope.rectangle.area} or ${rectangle.area}

SERVLETS AND JSP COURSE
www.globalmentoring.com.mx

Expression Language allows us to access some implicit objects, for example one of them is the cookie object $ {cookie.nameCookie.value} automatically we have this object available and we can directly access the object to start requesting cookies or cookies objects, simplifying the syntax a lot.

One of the most important objects in JSPs is the pageContext object. This object contains all the necessary information required by a JSP. This pageContext object contains most of the implicit objects used by a JSP, we only show as an example $ {pageContext.session.id} to access the http session object and how we can obtain the identifier of our session (JSession), but there are many more variables that can be accessed when reviewing the API of the JSPs.

We will also be able to access the parameters that we receive from a form or from a get or post request. To be able to access the parameters of an HTTP request we can use the implicit variable param or use the variable paramValues, for example the syntax $ {param.name}.

We can also access the HTTP headers by means of the implicit variables header or headerValues. As an example we can use the syntax $ {header ["user-agent"]}. And as we have commented we also have the cookie object $ {cookie.nombreCookie.value} with which we will be able to directly access the cookies that we have stored in our web browser and simply by putting the name of the cookie we can access later the value of the respective cookie.

Finally we observe that we can access the attributes that we have stored in our web application through the objects pageScope, requestScope, sessionScope or applicationScope. As we have commented previously, unlike the use of pageContext.session, it will return us to the http session object, through which we can access the attributes but of the session object and not the attributes that we have added in the session scope. To be able to access the attributes that we have added in the session scope we can use the sessionScope object, for example if we have added to the session an object called rectangle we can use two syntax one is specifying $ {sessionScope.rectangle.area} that we are looking for this object in the scope of sessionScope and later we can access the respective property of this JavaBean, but we can also use the following notation $ {rectangle.area}, in this notation we specify only the name of the JavaBean and later we access the respective property. If we do not specify the scope what it is going to do is first check if there is already a variable called rectangle in the scope of pageScope, if it is not found here it follows with the following scopes, resquestScope, sessionScope and applicationScope in that order. If it is not found in any of them, the object would be null, but as we have said in Expression Language it will allow us to handle the null concept and it will simply display an empty string.

What happens if this object is found in two scopes, for example pageScope and resquestScope? What happens is that you will take the first one according to the order mentioned. The first object in order of the mentioned ranges that you find is the object that is going to be used. Next we are going to review some exercises to put into practice the concept of Expression Language using JSPs.

ONLINE COURSE

SERVLETS AND JSP'S

By: Eng. Ubaldo Acosta

Global Mentoring
JAVA
ONLINE COURSES
UNIVERSITY

SERVLETS AND JSP COURSE

www.globalmentoring.com.mx