Expresiones de Lenguaje

EXPRESSION LANGUAGE (EL)

- EL nos permite simplificar el despliegue de información en un JSP utilizando JavaBeans.

- Sintaxis con acciones JSP:

- <jsp:useBean id="nombreBean" class="ClaseBean" />

- <jsp:getProperty name="nombreBean" propiedad="nombre Propiedad" />

- Sintaxis con Expression Language (EL):

- ${ nombreBean.nombrePropiedad }
- ${ nombreBean["nombrePropiedad"] }

El objetivo de este lenguaje es que nos permita simplificar la información que estamos desplegando en los JSPs. Normalmente este tipo de despliegue de información nos apoyaremos de los JavaBeans, según ya hemos visto, para desplegar información almacenada en una aplicación Web.

En la lamina podemos revisar una comparativa entre la sintaxis utilizando las acciones de los JavaBeans y la sintaxis de Expression Language.

Como podemos observar la sintaxis de los JavaBeans según hemos visto tenemos que declarar el nombre del bean que vamos a utilizar dentro del JSP y posteriormente acceder a la propiedad, esta sintaxis puede ser muy extensa si es que estamos haciendo mucho código para despliegue de propiedades ejemplo:

```
<jsp:useBean id="nombreBean" class="ClaseBean"/>
<jsp:getProperty name="nombreBean" propiedad="nombre Propiedad"/>
```

En cambio, con la sintaxis de Expression Language únicamente debemos de especificar el nombre del bean que vamos a utilizar y el nombre de la propiedad a la cual queremos acceder, la forma más común de hacerlo es: ${nombreBean.nombrePropiedad}, pero también tenemos otra forma ${nombreBean["nombrePropiedad"]}.

Internamente el lenguaje de expresión (EL) lo que hace es almacenar sus atributos o beans como si fuera un mapa, entonces por ello podemos especificar el nombre de la propiedad o también podemos especificarlo entre corchetes, ya que puede funcionar como un índice para acceder al mapa y posteriormente recuperar el valor respectivo de la propiedad.

Cualquiera de estas dos anotaciones nos va a permitir acceder a los JavaBean y a las propiedades de los JavaBean utilizando Expression Language y cómo podemos observar se simplifica bastante la sintaxis si es que la comparamos con el concepto de las acciones de los JavaBeans.

CARACTERÍSTICAS DE EXPRESSION LANGAGUE

- Los objetos JavaBean a utilizar se deben agregar previamente en algún alcance (scope) por medio del método setAttribute() en un Servlet:
  - page
  - request
  - session
  - application
- La notación es muy simplificada, pero solo permite la lectura de información (getters). No existe notación para la modificación de los atributos en un JavaBeans (setters).
- Permite acceder a propiedades de un JavaBean de manera anidada.
  Ej. ${alumno.direccion.calle}

Un detalle importante para hacer uso de este lenguaje es que los objetos JavaBeans a utilizar ya deben de estar agregados previamente en algún alcance, por ejemplo, utilizando el método setAttribute() en un Servlet. Si recordamos los alcances que tenemos en una aplicación web tenemos los alcances de page, resquest, session y application.

Entonces, para que este lenguaje de Expression Language pueda acceder a un atributo que hayamos agregado a nuestra aplicación web debe de estar previamente ya almacenado en cualquiera de estos alcances.

Otra característica de este lenguaje es que la notación es muy simplificada según vimos en la lámina anterior, se simplifica a una sola línea de código, perso solo nos permite leer información más no modificarla en nuestros Java Beans.

El Expression Language nos va a permitir acceder a las propiedades de un JavaBean de manera anidada, por ejemplo, supongamos que tenemos una clase Alumno y a su vez tiene una propiedad que es otra clase llamada Dirección y esta clase Dirección tiene como atributo “calle”. Entonces, podemos observar que ya no tenemos que instanciar cada uno de estos objetos para ir recuperando la información, si no que de manera anidada y por medio de la notación de punto, podemos acceder a nuestro JavaBean llamado alumno, posteriormente con este atributo se manda a llamar el método getDireccion(), el cual regresa un objeto de tipo Dirección y finalmente se manda llamar el método getCalle(). Vamos a revisar más adelante algunos ejercicios para poder entender a más detalle esta notación anidada.

MÁS CARACTERÍSTICAS DE EXPRESSION LANGAGUE

- Acceso a propiedades de un objetos de tipo Collection o Arreglos.
  Ej. ${ listaPersonas[índice/llave] }

- Nota: No es posible iterar los elementos, para ellos debemos usar JSTL.

- Conversiones automáticas de tipos de datos al desplegar la información.

- Manejo automático de valores null o vacíos, convirtiéndolos en cadenas vacías.

- Conjunto de operadores :
  - ${ 3 +2 -1}
  - ${ "x" > "y" }
  - ${ 3 >= 10/2 }

Continuando con las características de este lenguaje, podemos acceder a propiedades que están en una colección o en un arreglo. Según hemos comentado, simplemente con el nombre del bean que queremos acceder **${listaPersonas[índice/llave]}** y entre llaves especificamos el índice de la colección a la cual queremos acceder o la llave.

Por ejemplo, si es que el atributo es un mapa, esta notación nos va a permitir acceder de manera muy simple a los elementos que tengamos en una colección. Sin embargo no es posible iterar los elementos. Si queremos iterar cada uno de los elementos que tengamos en una colección debemos que utilizar JSTL (JSP Standard Tag Library) el cual estudiaremos más adelante.

Otra característica que tenemos respecto al Expression Languaje es que el despliegue de la información se hace una conversión al tipo especificado de manera automática. 55 4573 8370

Otra característica es que se maneja de manera automática los valores nulos o vacíos convirtiéndolos simplemente en cadena vacía. Esto nos permite de alguna manera simplificar el manejo de excepciones utilizando EL, el detalle con esto es que si existe alguna excepción por ejemplo por un nullPointerExcepcion el JSP no nos va a mostrar esta excepción en nuestra pantalla, si no que ese mensaje únicamente se va a mandar a la salida estándar y no se desplegará en el navegador web. Entonces esta característica puede facilitar el despliegue de información pero también podría ocultar detalles importantes respecto al manejo de excepciones.

Otra característica es que este mismo lenguaje tiene un conjunto de operadores básicos para hacer operaciones simples directamente. Por ejemplo la sintaxis **${3+2-1}** internamente se realiza la evaluación de la expresión y nos da un resultado que es finalmente lo que se despliega. Esto a final de cuentas es equivalente a tener un **out.println** y va a desplegar la información contenida dentro de esta expresión evaluada. También algunas conversiones ocurren de manera automática, por ejemplo **${ "x">"y" }** el valor que tenga X y el valor que tengas Y se convierten a tipos enteros y se hace una comparación utilizando los elementos X y Y. Otro ejemplo es **${ 3>= 10/2 }** de igual manera esta expresión se evalúa y es equivalente a tener la sintaxis del método out.print.

ACCESO A VARIABLES IMPÍCITAS CON EL

- Objeto **pageContext**. Ej. ${pageContext.session.id}

- Valores de parámetros con **param** y **paramValues**.
- Ej. ${ param.nombre }

- Valores de cabeceros con **header** y **headerValues**.
- Ej. ${ header["user-agent"] }

- Valores con el objeto **cookie**.
- Ej. ${ cookie.nombreCookie.value }

- Valores de atributos en algún alcance con **pageScope**, **requestScope**, **sessionScope** y **applicationScope.**
- Ej. ${sessionScope.rectangulo.area} ó ${rectangulo.area}

El lenguaje de Expression Language nos permite acceder a algunos objetos implícitos, por ejemplo uno de ellos es el objeto cookie **${ cookie.nombreCookie.value }** de manera automática ya tenemos disponible este objeto y podemos acceder directamente al objeto para empezar a solicitar cookies u objetos cookies, simplificando bastante la sintaxis.

Uno de los objetos más importantes en JSPs es el objeto de pageContext. Este objeto contiene toda la información necesaria que requiere un JSP. Este objeto pageContext contiene la mayoría de los objetos implícitos que utiliza un JSP, en solo mostramos como ejemplo **${pageContext.session.id}** para acceder al objeto http session y cómo podemos obtener el identificador de nuestra sesión (JSession), pero existen muchas más variables que se pueden acceder al revisar el API de los JSPs.

También vamos a poder acceder a los parámetros que recibamos desde un formulario o desde una petición get o post. Para poder acceder a los parámetros de una petición HTTP podemos utilizar la variable implícita **param** o utilizar la variable **paramValues**, por ejemplo la sintaxis **${param.nombre}**.

También podemos acceder a los cabeceros HTTP por medio de las variables implícitas **header** o **headerValues**. Como ejemplo podemos usar la sintaxis **${header["user-agent"]}**. Y según hemos comentado también tenemos el objeto cookie **${cookie.nombreCookie.value}** con el cual vamos a poder acceder directamente a las cookies que tengamos almacenadas en nuestro navegador web y simplemente poniendo el nombre de la cookie podemos acceder posteriormente al valor de la cookie respectiva.

Por último observamos que podemos acceder a los atributos que tengamos almacenados en nuestra aplicación web por medio de los objetos **pageScope, requestScope, sessionScope o applicationScope**. Según hemos comentado anteriormente a diferencia del uso de **pageContext.session** nos va a regresar es el objeto http session, por medio del cual podemos acceder a los atributos pero del objeto session y no a los atributos que hemos agregado en el alcance de session. Para poder acceder a los atributos que hemos agregado en el alcance de sesión podemos utilizar el objeto **sessionScope,** por ejemplo si hemos agregado a la sesión un objeto llamado rectángulo podemos utilizar dos sintaxis una es especificando **${sessionScope.rectangulo.area}** que estamos buscando este objeto en el alcance de sessionScope y posteriormente podemos acceder a la propiedad respectiva de este JavaBean, pero también podemos utilizar la siguiente notación **${rectangulo.area},** en esta notación especificamos únicamente el nombre del JavaBean y posteriormente accedemos a la propiedad respectiva. Si no especificamos el alcance lo que va hacer es primero revisar si es que ya existe una variable llamada rectángulo en el alcance de pageScope, si no la encuentra aquí se sigue con los siguientes alcances, resquestScope, sessionScope y applicationScope en ese orden. Si no se encuentra en ninguno de ellos el objeto sería nulo, pero según hemos comentado en Expression Language nos va a permitir manejar el concepto de nulos y nos va a desplegar simplemente una cadena vacía.

¿Qué sucede si este objeto se encuentra en dos alcances, por ejemplo pageScope y en resquestScope? Lo que sucede es que va a tomar el primero de ellos según el orden mencionado. El primer objeto en orden de los alcances mencionados que encuentre es el objeto que se va a utilizar. A continuación vamos a revisar algunos ejercicios para poner en práctica el concepto de Expression Language utilizando JSPs.