Carrera: Licenciatura en Sistemas Materia: Orientación a Objetos II Año: 2018

Equipo docente:
Titular: Prof. María Alejandra Vranić alejandravranic@gmail.com
Ayudantes: Prof. Leandro Ríos leandro.rios.unla@gmail.com
Prof. Gustavo Siciliano gussiciliano@gmail.com
Prof. Romina Mansilla romina.e.mansilla@gmail.com

JavaScript

JavaScript es uno de las tres herramientas que los desarrolladores web deben aprender:

1. HTML: Para definir el contenido de las páginas web.
2. CSS: Para especificar el diseño de las páginas web.
3. JavaScript: Para programar el comportamiento de las páginas web.

Lenguaje interpretado que hace uso del Modelo de Objetos del Documento (DOM) de los navegadores web para poder manipular diferentes aspectos de una página web.

Características:

1. Estructurado.
2. Dinámico.
3. Funcional.

Una de las grandes potencias que tiene JavaScript es que permite crear páginas web dinámicas desde el lado del cliente, mediante la utilización de AJAX (Asynchronous JavaScript And XML). Es posible realizar peticiones al servidor, obtener un resultado y mostrarlo en la vista de la página sin tener la necesidad de actualizar.

JQuery:

JQuery es una biblioteca de JavaScript, la cual posee toda la funcionalidad de JS integrada y estructurada de una manera más cómoda y eficaz. Posee una amplia gama de plugins como por ejemplo manejo de tablas, imágenes, validaciones, etc. Es la librería más utilizada hoy a la hora de realizar peticiones AJAX al servidor.

Se puede descargar la librería de Jquery desde la web: https://jquery.com/download/

Problema del Sistema Francés:

Para poder aplicar sentencias de JS en el TP lo primero es agregar la librería de JQuery en la vista (página HTML, JSP, etc), esto se hace dentro de la etiqueta HEAD de la siguiente manera:

Agregamos la librería *jquery-3.3.1.js* al proyecto en la carpeta NombreDelProyecto\WebContent\js\.

```
<HEAD>
	<META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
	<TITLE>Sistema Francés - Cliente </TITLE>
	<script src="js/jquery-3.3.1.js"></script>

</HEAD>
```

Luego se pueden agregar sentencias que estén asociadas al inicio de la página y/o funciones que sean reutilizables.

Por ejemplo, el botón consultar cliente por DNI del inicio realiza un POST al servidor por intermedio de un SUBMIT, esto produce que se refresque toda la página, entonces podemos realizar el llamado POST mediante AJAX para no tener que actualizar toda la pantalla, en este ejemplo se actualizará solamente la sección con los datos del cliente.

En el código agregamos un evento del tipo document.ready (este se va a ejecutar una vez que la página termine de cargar todos los componentes), dentro de este evento asociamos el código JS a los componentes del HTML, de la siguiente manera:

```
<HEAD>
	<META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
	<TITLE>Sistema Francés - Cliente </TITLE>
	<script src="js/jquery-3.3.1.js"></script>
```

```
	<script type="text/javascript">
		$(document).ready(function () {
			$('#consultar').click(function () {
				var dni = $('#dni').val();
				$.ajax({
				method: "POST",
					url: "MostrarClienteJSP",
					data: { dni: dni },
					async: false
				}).done(function (data) {
					$("#responsecliente").html(data);
				})
			 });

		});

	</script>
</HEAD>
```

Entonces analicemos el código JQuery:

$('#consultar') => Esto significa que se tomará el componente en el HTML con id="consultar" y lo asociará al DOM de la web, de esta manera podemos aplicarle sentencias JQuery a este componente; por ejemplo:

".click(function () { //Código a ejecutar });"

Cuando se haga clic en el componente se ejecutará el código de la función definida que se pasa como parámetro.

Dentro de la función encontramos:

"var dni = $('#dni').val();" Este caso es similar al anterior, se toma el componente con id="dni", se asocia al DOM, se puede tomar el valor que tiene el componente (sólo en caso de que sea un componente que tenga valor como un <input>, <select>, etc) y se guarda en la variable dni.

Código AJAX:

```
				$.ajax({
				method: "POST",
					url: "MostrarClienteJSP",
					data: { dni: dni },
					async: false
				}).done(function (data) {
					$("#responsecliente").html(data);
				})
```

En este ejemplo sea crea una petición AJAX al servidor con el método de envío POST, que invocará al controlador MostrarClienteJSP, al que se le pasa por request el dni, se marca asincrónico como false, para que se quede esperando la respuesta y se define el evento done(). Cuando el servidor retorne la respuesta, se invocará este evento, el cual ejecutará una función JS que recibe como parámetro esa

respuesta (data). Dentro de esta función tomamos el componente con id=”responcecliente” y lo llenamos con el resultado que nos devuelve el servidor.

Para que esto funcione el HTML debe estar de la siguiente manera:

```
<BODY>
  <%@ include file="/cabecera.jsp"%>
  <div class="jumbotron">
      <div class="container">
      <h1>Búsqueda de clientes</h1>
      <form class="navbar-form navbar-right">
        <div class="form-group">
            <label for="dni">DNI:</label>
      <INPUT id="dni" name="dni">
        </div>
             <INPUT    id="consultar"    type="button"    class="btn    btn-success"
value="Consultar" />
      </form>
    </div>
    <div class="container">
      <div id="responsecliente"></div>
        <div id="cuotasavencer">
            <TABLE border="0">
            <TR>
             <TD>
            <INPUT  id="btncuotasavencer"  type="button"  class="btn  btn-success"
value="Mostrar cuotas a vencer">
            </TD>
            </TR>
         </TABLE>
         <div id="divcuotasavencer"></div>
      </div>
      <TABLE border="0">
        <TR>
            <TD>
            <INPUT  id="listarPrestamos"  class="btn  btn-success"  type="button"
value="Mostrar prestamos">
      </TD>
      </TR>
    </TABLE>
   <div id="divlistarPrestamos"></div>
  </div>
 </div>
</BODY>
```

La sintaxis del controlador es la siguiente::

```
private void procesarPeticion(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            response.setContentType("text/html;charset=UTF-8");
            try {
                  ClienteABM clienteabm = new ClienteABM ();
```

```
                    int dni = Integer.parseInt(request.getParameter("dni"));
                    Cliente cliente=clienteabm.traerCliente(dni);
                    PrestamoABM prestamoAbm = new PrestamoABM();
                    List<Prestamo> lstPrestamos =
prestamoAbm.traerListaPrestamos(cliente);
                    request.setAttribute("cliente", cliente);
                    request.setAttribute("lstPrestamo", lstPrestamos);

request.getRequestDispatcher("/ajaxvistacliente.jsp").forward(request , response);
              }
              catch (Exception e) {
                    response.sendError(500, "El DNI Ingresado no existe en la base
de datos.");
              }
       }
```

La vista solo tiene la porción a mostrar:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
       pageEncoding="ISO-8859-1"%>
<%@page import="datos.Cliente"%>
<%@page import="datos.Prestamo"%>
<%@page import="java.util.List"%>
<% Cliente cliente=(Cliente)request.getAttribute("cliente"); %>
<BR>
       Apellido:
       <%= cliente.getApellido() %><BR>
       Nombre :
       <%= cliente.getNombre() %><BR>
       DNI :
       <%= cliente.getDocumento() %><BR>
       ID :
       <%= cliente.getIdCliente() %><BR>
<BR>
<input type="hidden" id="idcliente" name="idcliente" value="<%=
cliente.getIdCliente() %>" />
```

Por otra parte, también se puede llamar a sentencias JQuery dentro de una etiqueta, porque el componente puede estar definido en la porción de HTML a refrescar y por ello el document.ready pierde el efecto. Para evitarlo asociamos un evento, en este caso onchange, a la etiqueta <select>:

```
<select id="idprestamoSelected" onchange="mostrarprestamo();">
<option value="0">Seleccione un prestamo</option>
<% List<Prestamo> prestamos=(List)request.getAttribute("prestamos");
        for(Prestamo prestamo:prestamos){ %>
                    <option value="<%= prestamo.getIdPrestamo() %>"><%=
prestamo.getMonto() %></option>
       <% } %>
</select>
</div>
<div id="prestamosporcliente">
</div>
```

La función mostrarprestamo() se agrega luego del document.ready::

```
function mostrarprestamo()
{
	var idprestamo = $('#idprestamoSelected').val();
	if(idprestamo != "0")
	{
		$.ajax({
			method: "POST",
			url: "MostrarPrestamo",
			data: { idprestamo: idprestamo },
			async: false
		  }).done(function (data) {
			$("#prestamosporcliente").html(data);
		  })
	}
	else
	{
		$("#prestamosporcliente").html("");
	}
}
```

Preguntas:

1. ¿Cual es la diferencia principal entre submit y AJAX?

Referencias utilizadas:

http://getbootstrap.com/getting-started/

http://www.w3schools.com/jquery/default.asp

https://jquery.com/

https://api.jquery.com/