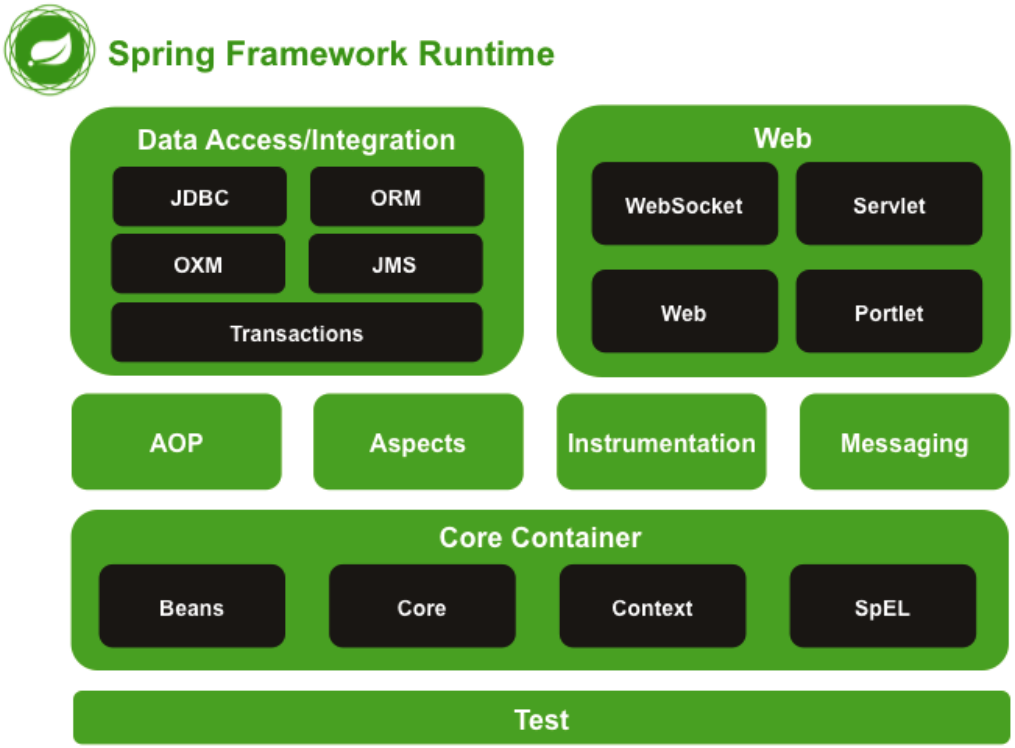
LibrosDigitales.net
Libros para Profesionales de Informática

SPRING FRAMEWORK

MODULO 02
SPRING MVC



TEMA 07
Integración con AJAX

Eric Gustavo Coronel Castillo

gcoronelc.blogspot.com

I N S T R U C T O R

CONTENIDO

Introducción a JQuery

Qué es JQuery?

JQuery es una librería específica de código JavaScript. JQuery se ha convertido en la librería más popular debido a su facilidad de uso y su gran potencia.

Muchas veces se confunden JavaScript y jQuery como dos lenguajes de programación distintos, es importante que entiendas que ambos son JavaScript. La diferencia es que jQuery ha sido optimizado para realizar muchas funciones de script frecuentes y lo hace a la vez que utiliza menos líneas de código.

Entre sus características tenemos:

- Permite interactuar con el DOM.
- Integración con AJAX.
- Manejo de eventos.

Función Principal

La función principal de esta librería se llama **`JQuery`** o simplemente **`$`**.

El uso de esta función y el **`ID`** de un elemento del DOM nos permite acceder al dicho elemento de un manera muy sencilla.

Aquí tenemos un ejemplo:

```
JQuery("#btnConsultar")
```

Es lo mismo que:

```
$("#btnconsultar")
```

Ejecutar Evento

La sintaxis para ejecutar un evento es:

```
$("#control").evento( función );
```

- **control**: Representa un elemento del DOM, por ejemplo un botón.
- **función**: Representa una función, puede ser una función in-line o el nombre de una función ya definida.

Aquí tenemos ejemplo de una función in-line:

```
$("#btnProcesar").click( function(){
   alert("Hola amigos de Gustavo");
} );
```

Aquí tenemos un ejemplo de una función previamente definida:

```
function saludar(){
   alert("Hola amigos de Gustavo");
}

$("#btnProcesar").click( saludar );
```

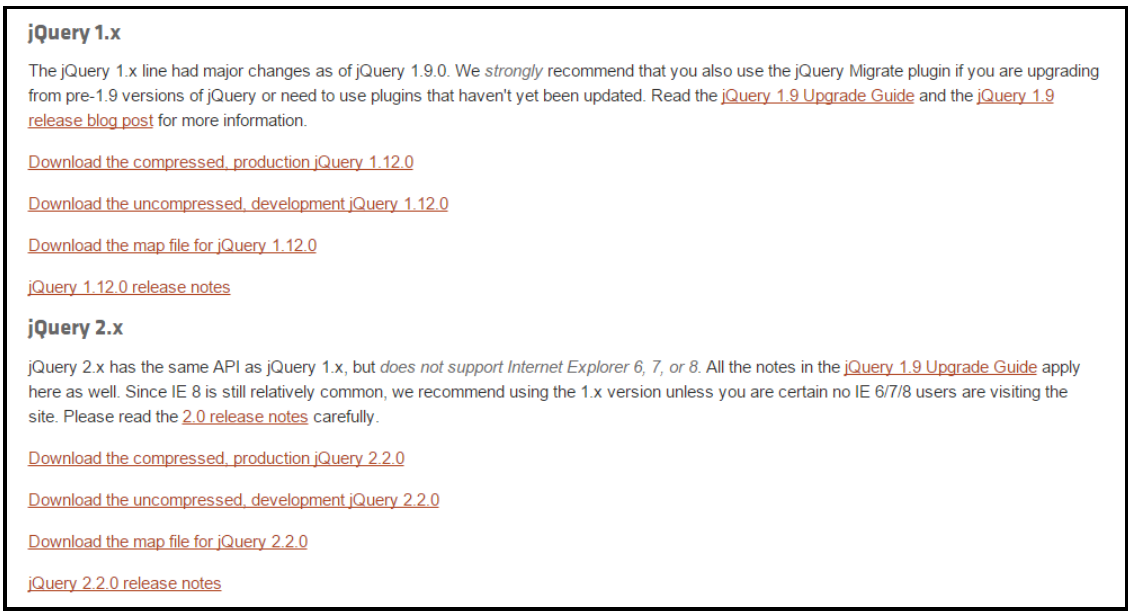
Iniciando con JQuery

Obtener la Librería

Para obtener la librería debes utilizar el siguiente enlace:

```
https://jquery.com/
```

Tienes la opción de descargar la versión 1.x o 2.x, la diferencia está en que la versión 2.x no da soporte a Internet Explorer 6, 7 y 8, tal como se puede apreciar en la siguiente imagen.



jQuery 1.x

The jQuery 1.x line had major changes as of jQuery 1.9.0. We *strongly* recommend that you also use the jQuery Migrate plugin if you are upgrading from pre-1.9 versions of jQuery or need to use plugins that haven't yet been updated. Read the jQuery 1.9 Upgrade Guide and the jQuery 1.9 release blog post for more information.

Download the compressed, production jQuery 1.12.0

Download the uncompressed, development jQuery 1.12.0

Download the map file for jQuery 1.12.0

jQuery 1.12.0 release notes

jQuery 2.x

jQuery 2.x has the same API as jQuery 1.x, but *does not support Internet Explorer 6, 7, or 8.* All the notes in the jQuery 1.9 Upgrade Guide apply here as well. Since IE 8 is still relatively common, we recommend using the 1.x version unless you are certain no IE 6/7/8 users are visiting the site. Please read the 2.0 release notes carefully.

Download the compressed, production jQuery 2.2.0

Download the uncompressed, development jQuery 2.2.0

Download the map file for jQuery 2.2.0

jQuery 2.2.0 release notes

Inicio de JQuery

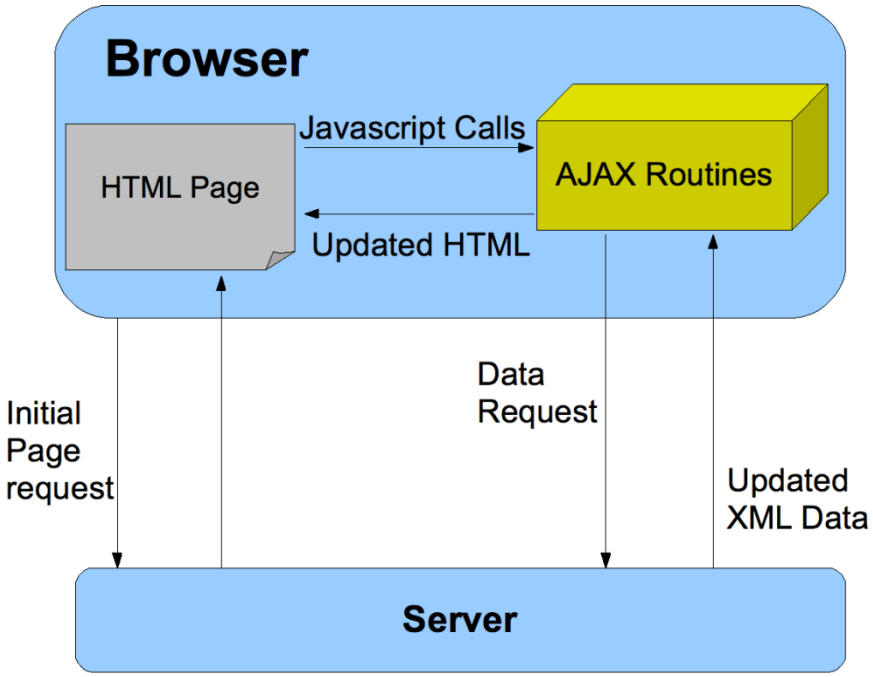
Para poder iniciar a programar con JQuery debes tener la siguiente configuración en la sección head de tu página JSP:

```
<head>
   <script src="jquery.js"></script>
   <script>
      $(function(){

         // Aquí debe empezar a programar con JQuery
         // Por ejemplo, el evento click de un botón

      });
   </script>
</head>
```

Soporte a AJAX



JQuery te proporciona varios métodos para hacer llamadas AJAX de una manera muy fácil.

- `$.ajax( . . . )`

  Es la función más completa que te ofrece JQuery para hacer llamadas AJAX. Las otras funciones son formas simplificadas de esta función-

- `$('#div').load( url, param )`

  Con la función **`load`** puedes hacer una llamada AJAX de tipo `GET` y cargar el resultado directamente en un control HTML como un div.

- `$.get( url, param, function( result, status, jqxhr ) { . . . } )`

  Con la función `get` puedes hacer una llamada AJAX utilizando el método `GET`, el resultado lo obtienes en la variable `result`.

- `$.post( url, param, function( result, status, jqxhr ) { . . . } )`

  Con la función `post` puedes hacer una llamada AJAX utilizando el método `POST`, el resultado lo obtienes en la variable `result`.

- `$.getScript( url, function( result, status, jqxhr ) { . . . } )`

  Con la función `getScript` puedes cargar dinámicamente librerías JavaScript, la llamada que realiza es de tipo `GET`.

Soporte JSON

Con Spring MVC tienes varias formas de generar JSON.

Utilizando Gson y HttpServletResponse

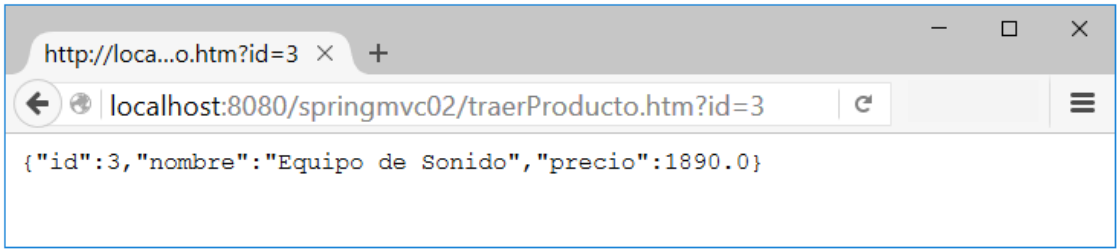
En primero lugar se debe agregar la dependencia de Gson:

```
<dependency>
   <groupId>com.google.code.gson</groupId>
   <artifactId>gson</artifactId>
   <version>2.5</version>
</dependency>
```

Aquí tienes un ejemplo:

```
  @RequestMapping(value="traerProducto.htm", method=RequestMethod.GET)
  public void traerProducto(
      @RequestParam("id") int id,
      HttpServletResponse response) throws IOException{
    // Se obtiene el producto basado en el id
    ProductoDto dto = productoService.getById(id);
    // Convertir el objeto dto en Json
    Gson gson = new Gson();
    String jsonResult = gson.toJson(dto);
    // Se envía la respuesta al browser
    response.setContentType("application/json; charset=UTF-8");
    response.getWriter().print(jsonResult);
  }
```

A continuación tienes el resultado de su ejecución:



Utilizando Gson y @ResponseBody

En este caso el método del controlador retorna una cadena contiene datos en formato json y van directamente al browser.

Aquí tienes un ejemplo:

```
@RequestMapping(value="traerProducto2.htm",
    method=RequestMethod.GET, produces="application/json")
public @ResponseBody String traerProducto2(@RequestParam("id") int id) {
  // Se obtiene el producto basado en el id
  ProductoDto dto = productoService.getById(id);
  // Convertir el objeto dto en Json
  Gson gson = new Gson();
  String jsonResult = gson.toJson(dto);
  // Se envía la respuesta al browser
  return jsonResult;
}
```

Y aquí tienes el resultado:

Utilizando Jackson

En primer lugar se debe agregar las dependencias de Jackson:

```
<dependency>
   <groupId>org.codehaus.jackson</groupId>
   <artifactId>jackson-core-asl</artifactId>
   <version>1.9.13</version>
</dependency>
<dependency>
   <groupId>org.codehaus.jackson</groupId>
   <artifactId>jackson-mapper-asl</artifactId>
   <version>1.9.13</version>
</dependency>
```

Estas librerías que generan el Json de manera transparente.

Aquí tienes un ejemplo:

```
@RequestMapping(value = "traerProducto3.htm",
      method = RequestMethod.GET, produces = "application/json")
  public @ResponseBody ProductoDto traerProducto3(@RequestParam("id") int id) {
  // Se obtiene el producto basado en el id
  ProductoDto dto = productoService.getById(id);
  // Se envía la respuesta al browser
  return dto;
}
```

Y este es el resultado: