

"Spring MVC – AJAX JSON"

Módulo 5 / 2

© Todos los logos y marcas utilizados en este documento, están registrados y pertenecen a sus respectivos dueños.

Objetivo

El objetivo de este laboratorio es aprenderemos a trabajar Spring en conjunto con Ajax usando la librería JavaScript **jQuery**, esta librería nos permite simplificar la manera de interactuar con AJAX, manejar eventos, tratar la respuesta en formato JSON y HTML, entre otras muchas cosas más, en otras palabras **“escribe menos haz más”**.

Hoy en día AJAX es algo bastante cotidiano, en este capítulo veremos algunos componentes que nos provee Spring que nos facilita mucho las cosas y hace que del lado del servidor el trabajo sea más simple. Cuando trabajamos con AJAX queremos, por un lado, identificar los request que fueron realizados por XMLHttpRequest, y por otro, retornar los datos en el formato que nos sea más conveniente (HTML, JSON, XML, etc).

Ajax con JSON es soportado como parte de Spring MVC. Esto incluye soporte con algunas anotaciones como **@ResponseBody**, clases y API para generar peticiones y respuestas con JSON utilizando Spring MVC **@Controller** o bien **@RestController**.



Introducción

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página, aunque existe la posibilidad de configurar las peticiones como síncronas de tal forma que la interactividad de la página se detiene hasta la espera de la respuesta por parte del servidor.

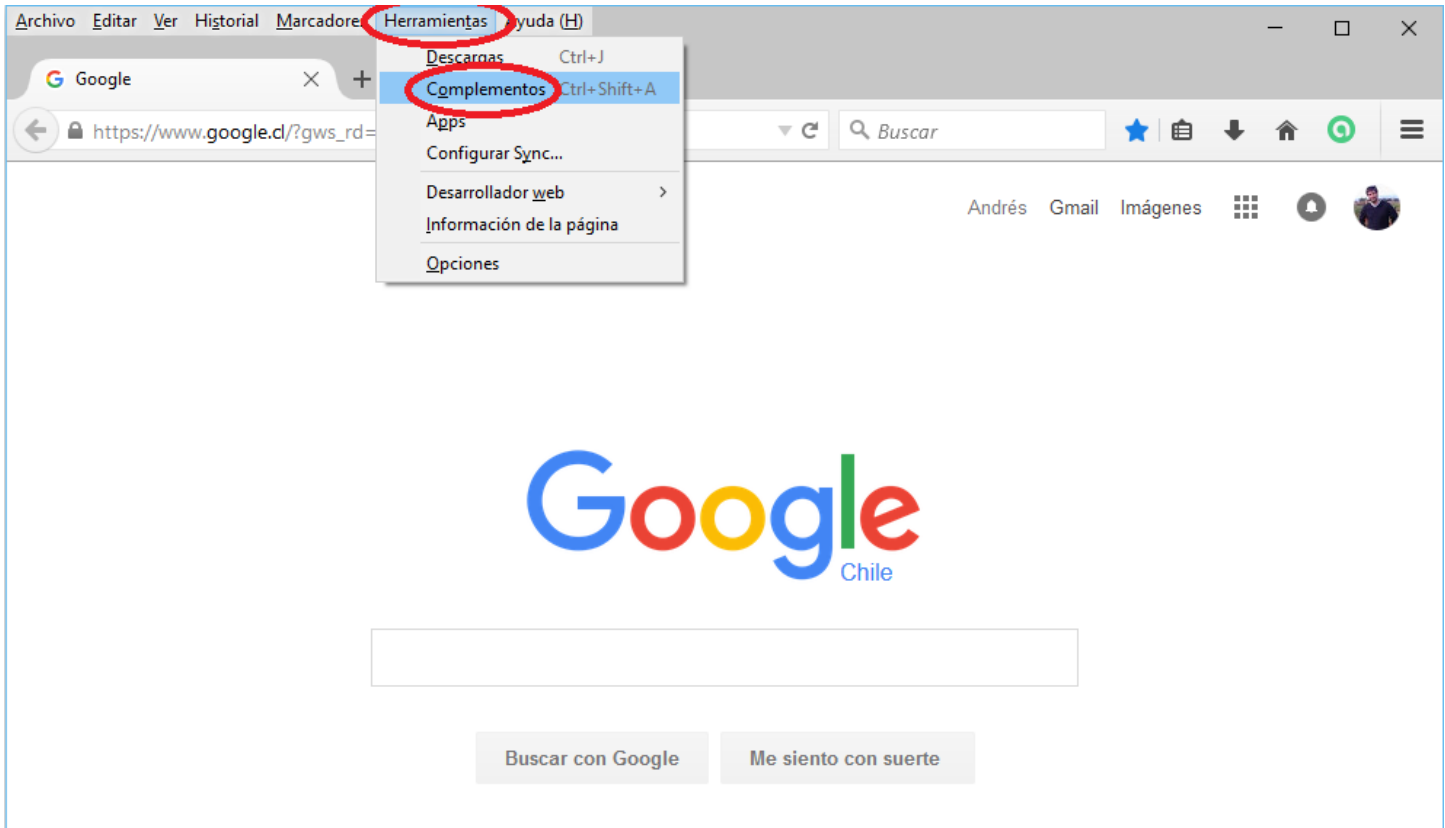
JavaScript es el lenguaje interpretado (scripting language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML.

"Quemar etapas"

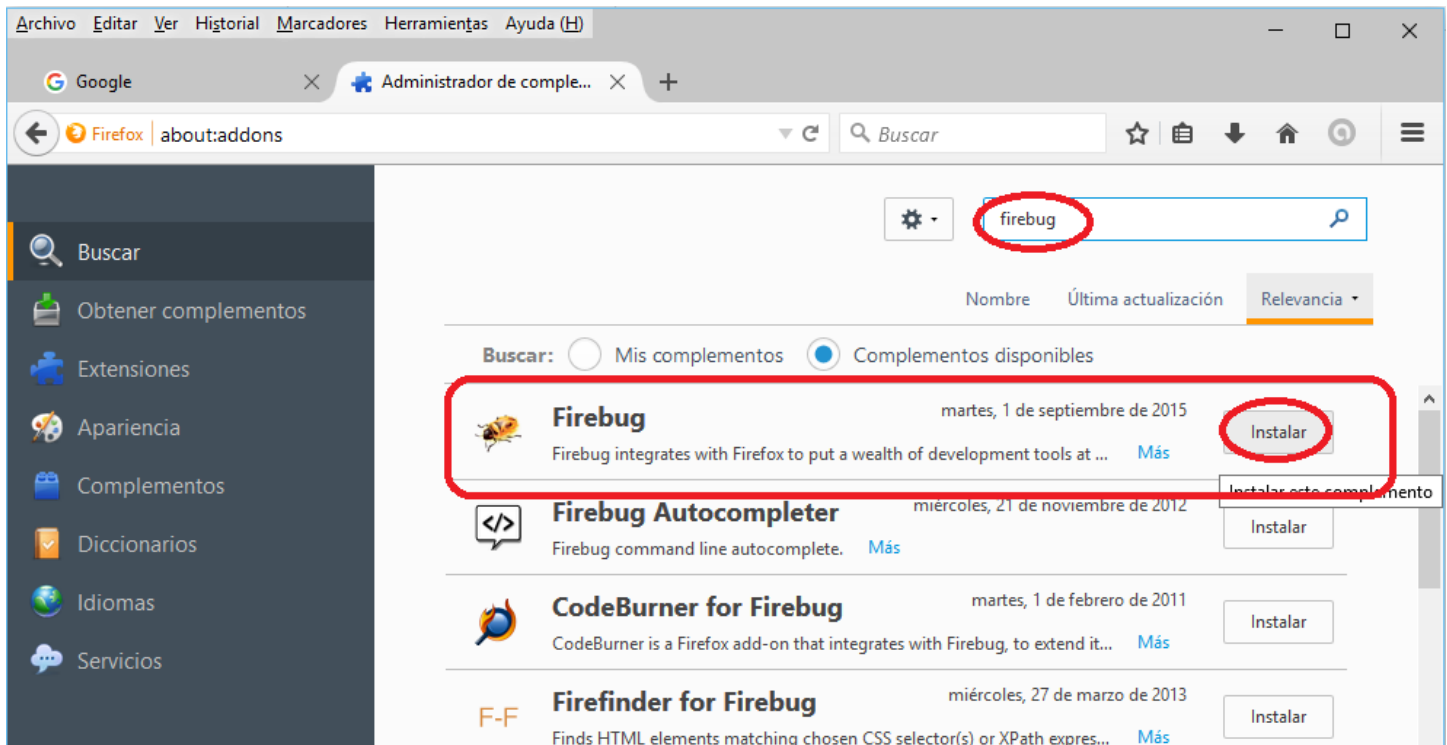
Es importante que saques provecho de cada módulo y consultes todos los temas que se van tratando, sin adelantar etapas.

Ejercicio 1: Instalar plug-in Firebug de Firefox

En este ejercicio, vamos a instalar en pocos pasos el plug-in Firebug de Firefox para hacer un seguimiento y debug de los datos.



- Buscamos por **firebug**
- Clic en instalar



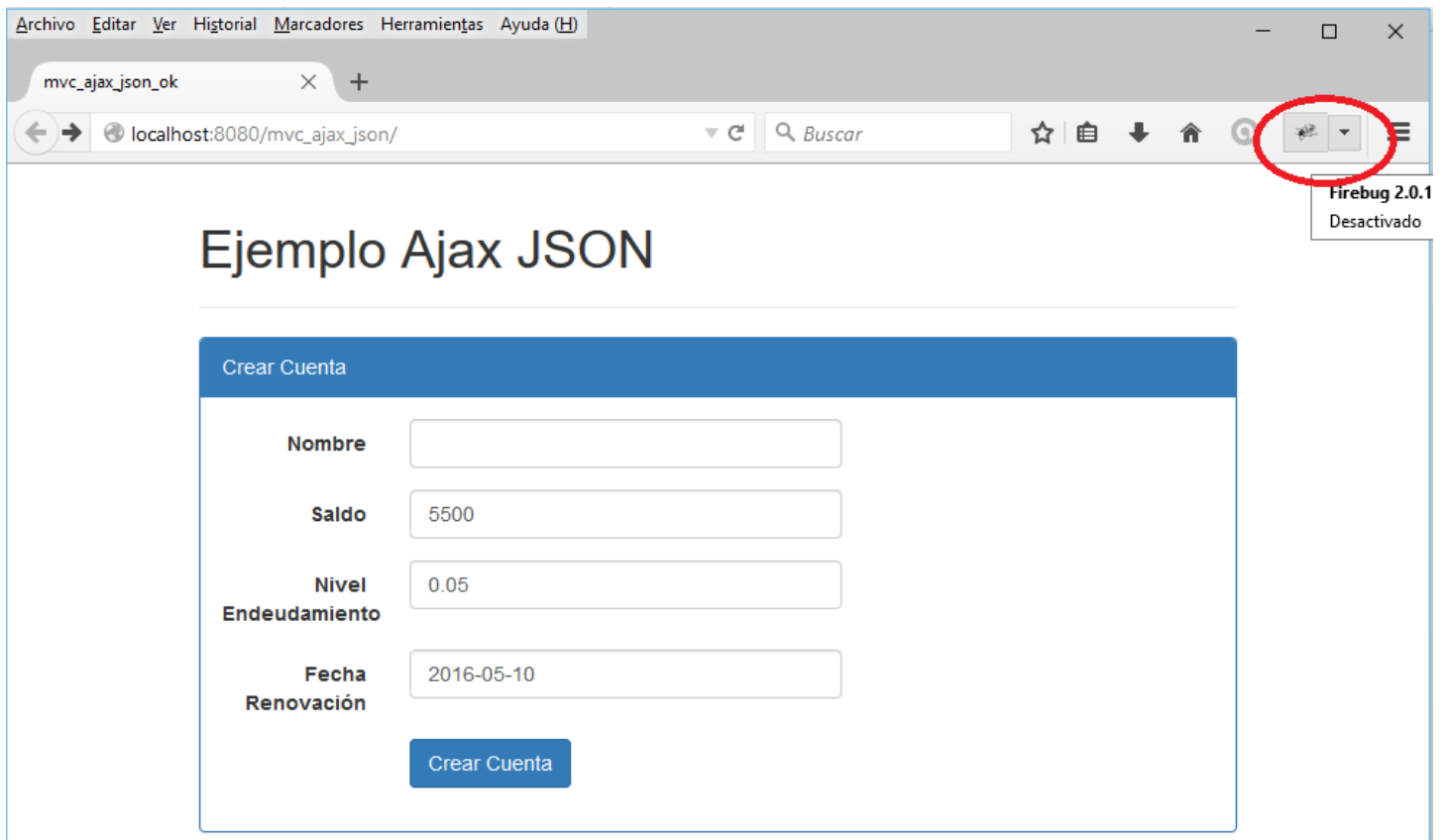
Ejercicio 2: Generar y ejecutar el ejemplo "mvc_ajax_json"

1. Clic derecho sobre el proyecto **Run As->Maven Clean** y **Run As->Maven Install**
2. Clic derecho **Maven->Update Project...**
3. Clic derecho sobre el proyecto **mvc_ajax_json-> Run As on Server**
4. Observe el resultado en el navegador.

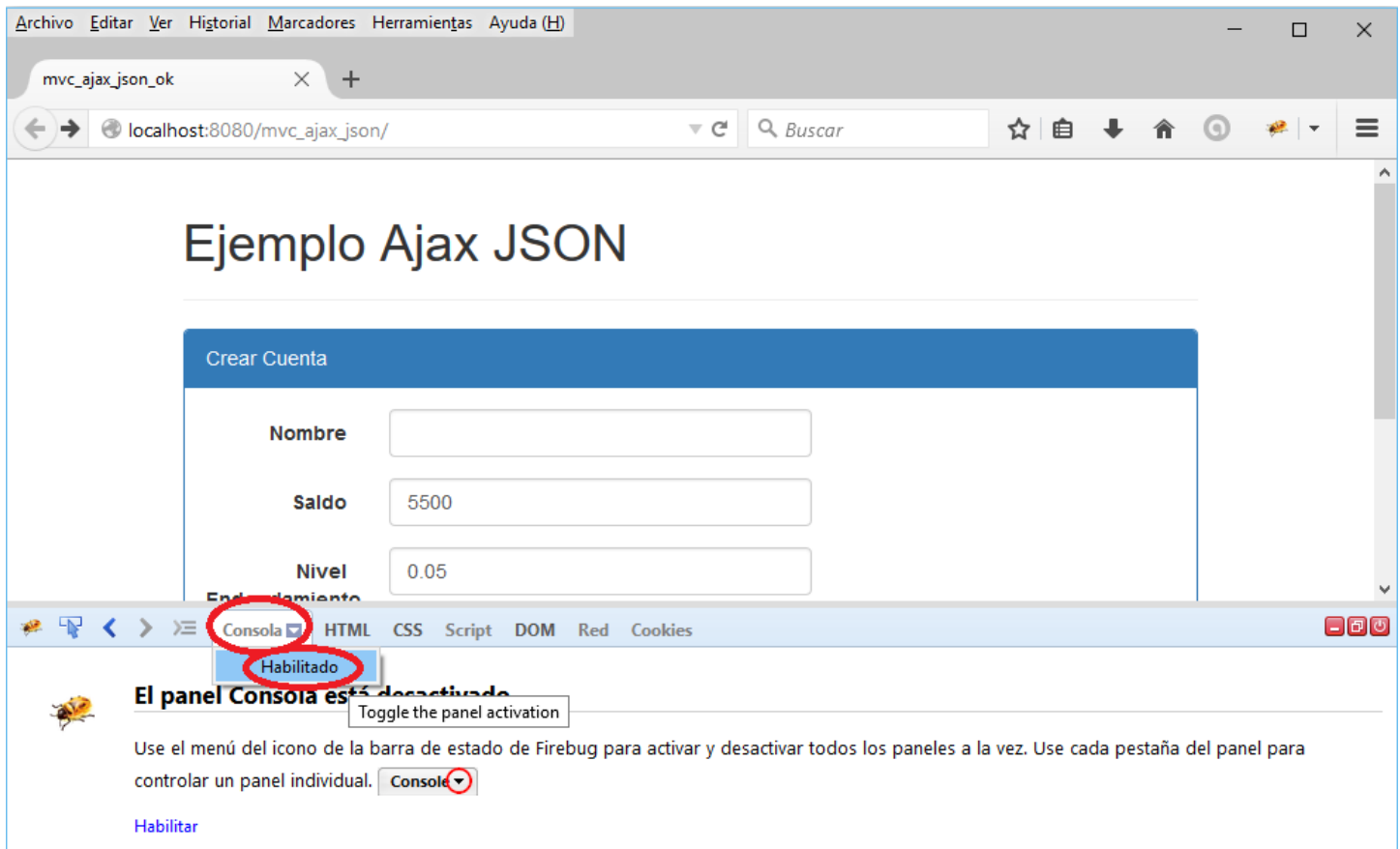
The screenshot shows a web browser window with the address bar displaying 'localhost:8080/mvc_ajax_json/'. The page content includes a header 'Ejemplo Ajax JSON' and a form titled 'Crear Cuenta'. The form contains four input fields: 'Nombre' (empty), 'Saldo' (5500), 'Nivel Endeudamiento' (0.05), and 'Fecha Renovación' (2016-05-10). A blue button labeled 'Crear Cuenta' is positioned below the fields.

5. Habilitamos Firebug y su consola:

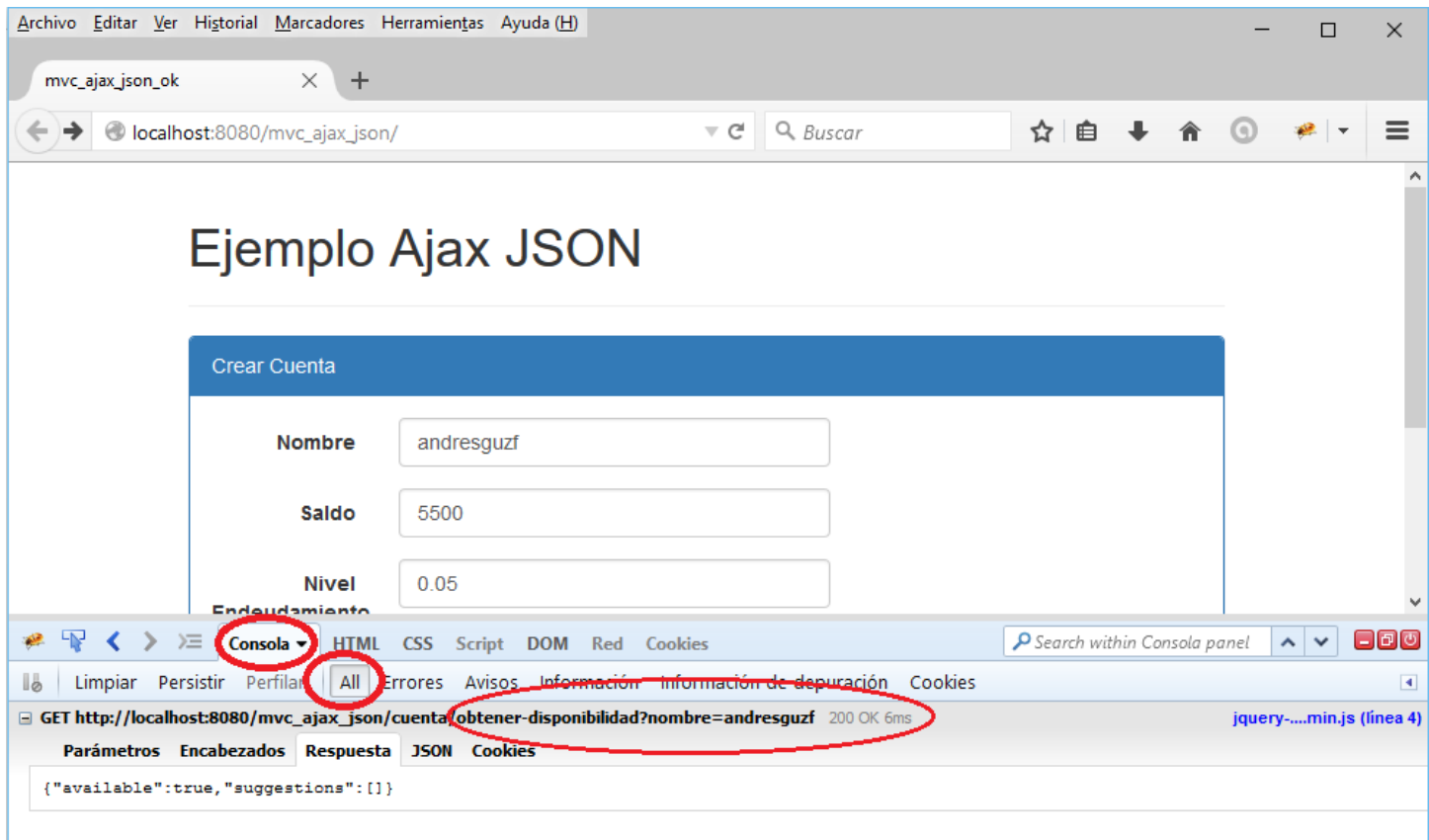
- Habilitamos primero Firebug:



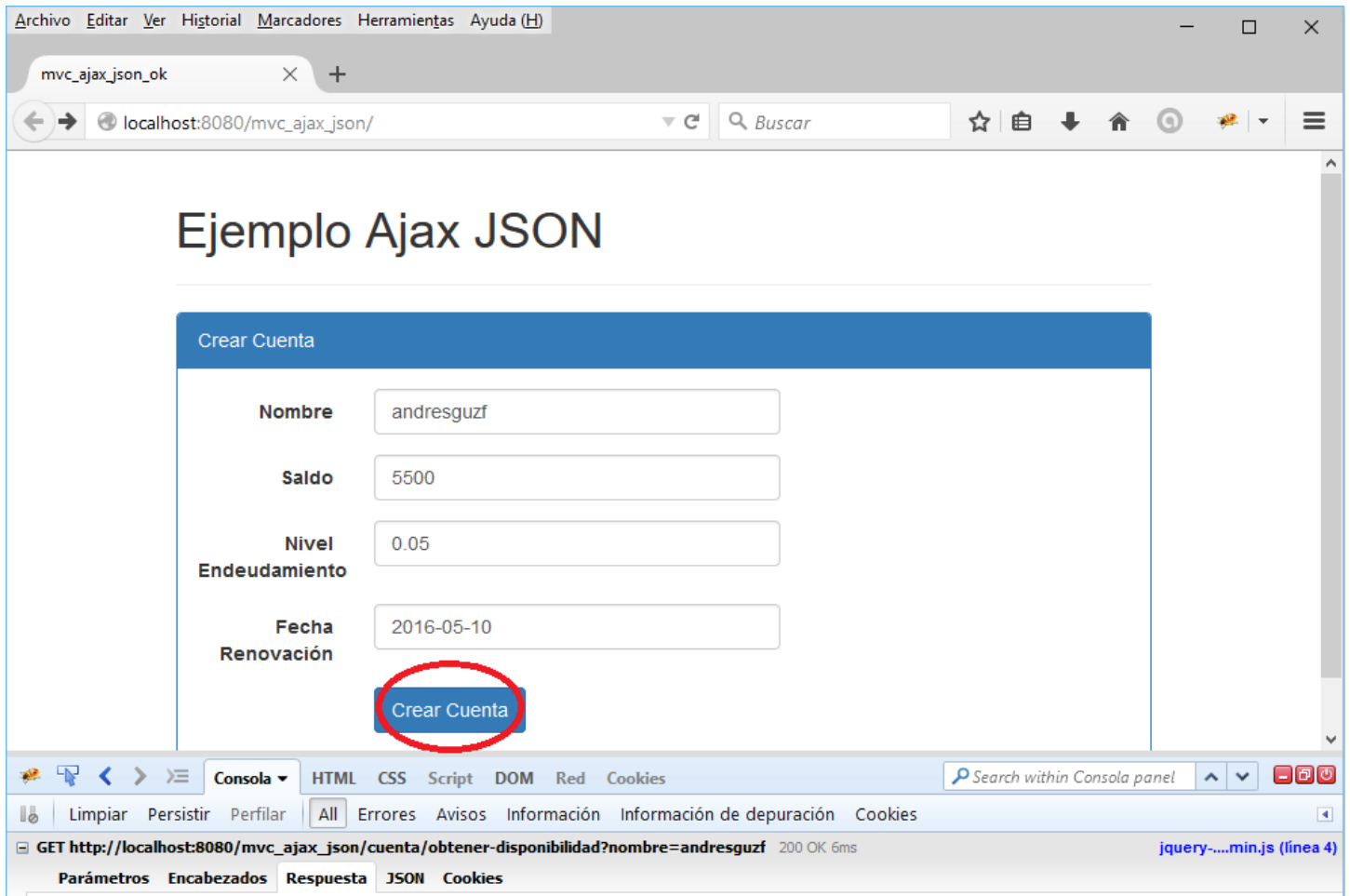
- Luego habilitamos la consola:



6. Ingrese algún valor para el campo Nombre y mueva el mouse fuera de campo de texto del nombre y clic al algún lugar fuera del input o textbox, esto genera el evento Javascript "Focus lost", observamos las peticiones y respuestas HTTP Ajax en la consola de Firebug.
 - Lo primero que hace es revisar mediante petición Ajax, si existe un nombre de usuario con el nombre ingresado, para no duplicarlo.



7. Clic en Crear para enviar la petición Ajax HTTP Post



Archivo Editar Ver Historial Marcadores Herramientas Ayuda (H)

mvc_ajax_json_ok

localhost:8080/mvc_ajax_json/ Buscar

Ejemplo Ajax JSON

Crear Cuenta

Nombre andresguzf

Saldo

Nivel Endeudamiento

Fecha Renovación

Cuenta creada con éxito

Cuenta Id	Nombre	Saldo	Nivel Endeudamiento	Fecha Renovación
1	andresguzf	5500	0.05	1462838400000

Cerrar

Crear Cuenta

Ejemplo Ajax JSON

Crear Cuenta

Nombre: andresguzf

Saldo

Nivel Endeudamiento

Fecha Renovación

Cuenta creada con éxito

Cuenta Id	Nombre	Saldo	Nivel Endeudamiento	Fecha Renovación
1	andresguzf	5500	0.05	1462838400000

Consola

HTML CSS Script DOM Red Cookies

Limpiar Persistir Perifoneo All Errores Avisos Información Información de depuración Cookies

POST http://localhost:8080/mvc_ajax_json/cuenta 200 OK 301ms

Encabezados Envío Respuesta JSON Cookies

```
{ "id": 1 }
```

GET http://localhost:8080/mvc_ajax_json/cuenta/1 200 OK 12ms

Encabezados Respuesta JSON Cookies

```
{ "id": 1, "nombre": "andresguzf", "saldo": 5500, "nivelEndeudamiento": 0.05, "fechaRenovacion": 1462838400000 }
```

8. Abrir y estudiar la clase controladora `/src/main/java/com.bolsadeideas.ejemplos.cuenta.controllers/CuentaController.java`

```
package com.bolsadeideas.ejemplos.mvc.ajax.cuenta;

import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;

import javax.servlet.http.HttpServletResponse;
import javax.validation.Valid;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.validation.FieldError;
import org.springframework.validation.ObjectError;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
@RequestMapping(value="/cuenta")
public class CuentaController {

    private Map<Long, Cuenta> cuentas = new ConcurrentHashMap<Long, Cuenta>();

    // Metodo handler formulario, para crear la cuenta
    @RequestMapping(method=RequestMethod.GET)
    public String crearCuentaForm(Model model) {
        model.addAttribute("cuenta", new Cuenta());
        return "cuenta/crearForm";
    }

    @RequestMapping(value="/obtener-disponibilidad", method=RequestMethod.GET)
    public @ResponseBody AvailabilityStatus getDisponibilidad(@RequestParam String nombre) {

        for (Cuenta a : cuentas.values()) {
            if (a.getNombre().equals(nombre)) {
                return AvailabilityStatus.notAvailable(nombre);
            }
        }

        return AvailabilityStatus.available();
    }
}
```

```
@RequestMapping(method=RequestMethod.POST)
public @ResponseBody Map<String, ? extends Object> crearCuenta(@Valid @RequestBody Cuenta
cuenta, BindingResult result, HttpServletResponse response) {
    List<ObjectError> errores =result.getAllErrors();

    if (!errores.isEmpty()) {
        response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
        return mensajesValidacion(errores);
    } else {
        cuentas.put(cuenta.asignarId(), cuenta);
        return Collections.singletonMap("id", cuenta.getId());
    }
}

@RequestMapping(value="{id}", method=RequestMethod.GET)
public @ResponseBody Cuenta verDetalle(@PathVariable Long id) {
    Cuenta cuenta = this.cuentas.get(id);
    if (cuenta == null) {
        throw new RecursoNoEncontradoException(id);
    }
    return cuenta;
}

private Map<String, String> mensajesValidacion(List<ObjectError> errores) {
    Map<String, String> mensajesErrores = new HashMap<String, String>();

    for (ObjectError fieldError : errores) {
        mensajesErrores.put( ((FieldError) fieldError).getField(),
fieldError.getDefaultMessage());
    }
    return mensajesErrores;
}
}
```

9. Abrir y estudiar la vista jsp "crearForm.jsp"

```

<%@page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>mvc_ajax_json_ok</title>
<!-- Bootstrap -->
<link rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">

<link rel="stylesheet" href="<c:url value="/recursos/css/popup.css" />" type="text/css">
<!-- jquery -->
<script src="//code.jquery.com/jquery-2.1.4.min.js"></script>
</head>
<body>
  <div class="container">
    <div class="page-header">
      <h1>Ejemplo Ajax JSON</h1>
    </div>
    <div class="panel panel-primary">
      <div class="panel-heading">Crear Cuenta</div>
      <div class="panel-body">
        <div class="container">
          <div class="row">
            <form:form modelAttribute="cuenta" action="cuenta" method="post"
              cssClass="form-horizontal" role="form">
              <div class="form-group">
                <form:label for="nombre" path="nombre"
                  cssClass="col-sm-2 control-label">Nombre</form:label>
                <div class="col-sm-10">
                  <form:input path="nombre" style="width: 300px;"
                    cssClass="form-control"
                    cssErrorClass="form-control alert-danger" />
                  <form:errors path="nombre" />
                </div>
              </div>
            </div>
            <div class="form-group">
              <form:label for="saldo" path="saldo"
                cssClass="col-sm-2 control-label">Saldo</form:label>
              <div class="col-sm-10">
                <form:input path="saldo" style="width: 300px;"
                  cssClass="form-control"
                  cssErrorClass="form-control alert-danger" />
                <form:errors path="saldo" />
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>

```



```

</div>
<div class="form-group">
  <form:label for="nivelEndeudamiento" path="nivelEndeudamiento"
    cssClass="col-sm-2 control-label">Nivel Endeudamiento
  </form:label>
  <div class="col-sm-10">
    <form:input path="nivelEndeudamiento" style="width: 300px;"
      cssClass="form-control"
      cssErrorClass="form-control alert-danger" />
    <form:errors path="nivelEndeudamiento" />
  </div>
</div>
<div class="form-group">
  <form:label for="fechaRenovacion" path="fechaRenovacion"
    cssClass="col-sm-2 control-label">Fecha Renovación
  </form:label>
  <div class="col-sm-10">
    <form:input path="fechaRenovacion" style="width: 300px;"
      cssClass="form-control"
      cssErrorClass="form-control alert-danger" />
    <form:errors path="fechaRenovacion" />
  </div>
</div>
<div class="form-group">
  <div class="col-sm-offset-2 col-sm-10">
    <input id="create" type="submit" value="Crear Cuenta"
      class="btn btn-primary" role="button" />
  </div>
</div>
</form:form>
</div>
</div>
</div>
</div>
<div id="mask" style="display: none;"></div>
<div id="popup" style="display: none;">
  <div class="span-8 last">
    <div class="panel panel-primary">
      <div class="panel-heading">Cuenta creada con éxito</div>
      <div class="panel-body">

        <table class="table table-striped table-hover table-bordered">
          <thead>
            <tr>
              <th>Cuenta Id</th>
              <th>Nombre</th>
              <th>Saldo</th>
              <th>Nivel Endeudamiento</th>
              <th>Fecha Renovación</th>
            </tr>
          </thead>

```

```

        <tbody>
            <tr>
                <td><span id="idAsignado"></span></td>
                <td><span id="nombreConfirmado"></span></td>
                <td><span id="saldoConfirmado"></span></td>
                <td><span id="nivelEndeudamientoConfirmado"></span></td>
                <td><span id="fechaRenovacionConfirmado"></span></td>
            </tr>
        </tbody>
    </table>
</div>
<div>
    <a href="#" onclick="closePopup();">Cerrar</a>
</div>
</div>
<script type="text/javascript">
    $(document).ready(function() {
        // check name availability on focus lost
        $('#nombre').blur(function() {
            if ($('#nombre').val()) {
                checkAvailability();
            }
        });

        $("#cuenta").submit(function() {
            var datos = {
                nombre : $('#nombre').val(),
                saldo : $('#saldo').val(),
                nivelEndeudamiento : $('#nivelEndeudamiento').val(),
                fechaRenovacion : $('#fechaRenovacion').val()
            };

            $.ajax({
                type : "post",
                url : "cuenta",
                data : JSON.stringify(datos),
                dataType : 'json',
                contentType : "application/json; charset=utf-8",
                success : function(data) {
                    $("#idAsignado").text(data.id);
                    showPopup();
                },
                error: function(e) {
                    console.log(e);
                },
            });

            return false;
        });
    });

```

```
function checkAvailability() {
    $.ajax({
        type : "get",
        url : "cuenta/obtener-disponibilidad",
        dataType : 'json',
        data : {
            nombre : $('#nombre').val()
        },
        success : function(availability) {

            if (availability.available) {
                fieldValidated("nombre", {
                    valid : true
                });
            } else {
                fieldValidated("nombre", {
                    valid : false,
                    message : $('#nombre').val()
                        + " no está disponible, intente "
                        + availability.suggestions
                });
            }
        }
    });
}

function fieldValidated(field, result) {
    if (result.valid) {
        $("#" + field + "Label").removeClass("error");
        $("#" + field + "\\errors").remove();
        $('#create').attr("disabled", false);
    } else {

        $("#" + field + "Label").addClass("error");
        if ($("#" + field + "\\errors").length == 0) {
            $("#" + field).after(
                "<span id='" + field + ".errors'" + result.message
                + "</span>");
        } else {

            $("#" + field + "\\errors").html(
                "<span id='" + field + ".errors'" + result.message
                + "</span>");
        }
        $('#create').attr("disabled", true);
    }
}
```

```
function showPopup() {

    $.ajax({
        type : "get",
        url : "cuenta/" + $("#idAsignado").text(),
        dataType : 'json',
        success : function(cuenta) {
            $("#nombreConfirmado").text(cuenta.nombre);
            $("#saldoConfirmado").text(cuenta.saldo);
            $("#nivelEndeudamientoConfirmado").text(
                cuenta.nivelEndeudamiento);
            $("#fechaRenovacionConfirmado").text(
                cuenta.fechaRenovacion);
        }
    });

    $('body').css('overflow', 'hidden');
    $('#popup').fadeIn('fast');
    $('#mask').fadeIn('fast');
}

function closePopup() {
    $('#popup').fadeOut('fast');
    $('#mask').fadeOut('fast');
    $('body').css('overflow', 'auto');
    resetForm();
}

function resetForm() {
    $('#cuenta')[0].reset();
}
</script>
</body>
</html>
```

Otro punto importante a tener en cuenta, es tener el pom.xml con las dependencias maven de Jackson JSON Mapper:

ETC...

```
<!-- Jackson JSON Mapper -->
```

```
<dependency>
```

```
    <groupId>com.fasterxml.jackson.core</groupId>
```

```
    <artifactId>jackson-databind</artifactId>
```

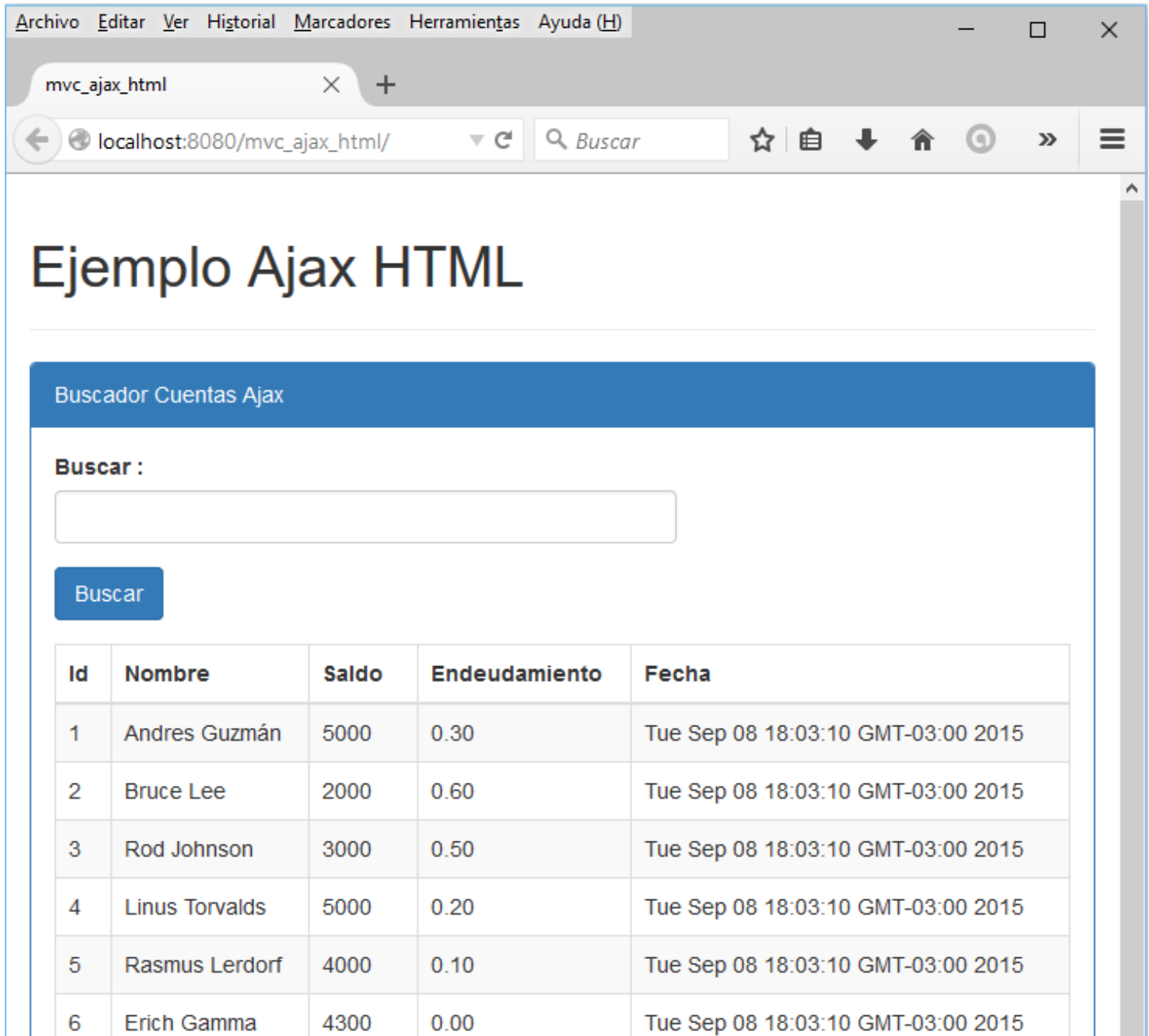
```
    <version>2.6.1</version>
```

```
</dependency>
```

ETC...

Ejercicio 3: Generar y ejecutar el ejemplo "mvc_ajax_html"

1. Clic derecho sobre el proyecto **Run As->Maven Clean** y **Run As->Maven Install**
2. Clic derecho **Maven->Update Project...**
3. Clic derecho sobre el proyecto **mvc_ajax_html-> Run As on Server**
4. Observe el resultado en el navegador.



Archivo Editar Ver Historial Marcadores Herramientas Ayuda (H)

mvc_ajax_html

localhost:8080/mvc_ajax_html/ Buscar

Ejemplo Ajax HTML

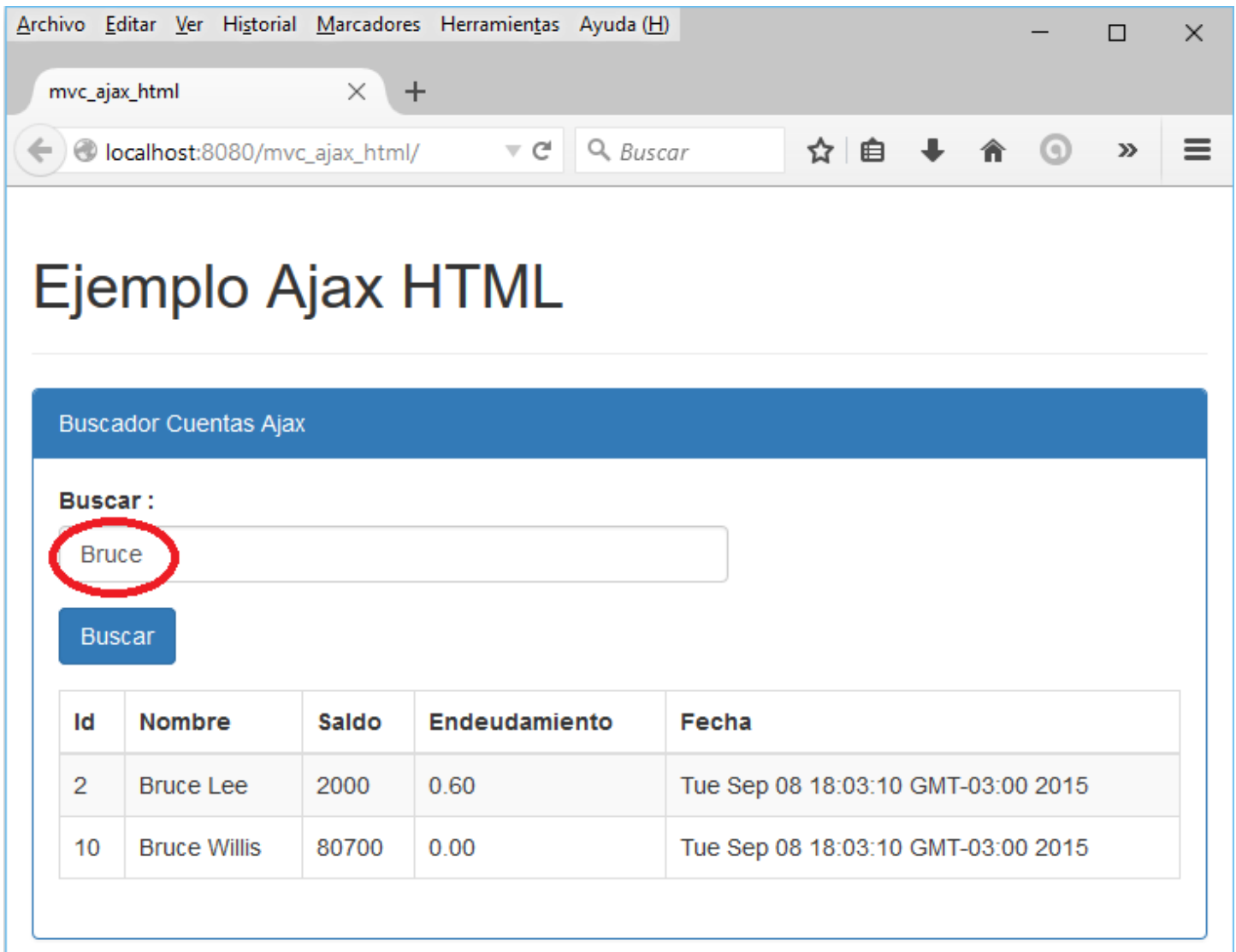
Buscador Cuentas Ajax

Buscar :

Buscar

Id	Nombre	Saldo	Endeudamiento	Fecha
1	Andres Guzmán	5000	0.30	Tue Sep 08 18:03:10 GMT-03:00 2015
2	Bruce Lee	2000	0.60	Tue Sep 08 18:03:10 GMT-03:00 2015
3	Rod Johnson	3000	0.50	Tue Sep 08 18:03:10 GMT-03:00 2015
4	Linus Torvalds	5000	0.20	Tue Sep 08 18:03:10 GMT-03:00 2015
5	Rasmus Lerdorf	4000	0.10	Tue Sep 08 18:03:10 GMT-03:00 2015
6	Erich Gamma	4300	0.00	Tue Sep 08 18:03:10 GMT-03:00 2015

5. Buscar por el nombre "Bruce"



Archivo Editar Ver Historial Marcadores Herramientas Ayuda (H)

mvc_ajax_html

localhost:8080/mvc_ajax_html/ Buscar

Ejemplo Ajax HTML

Buscador Cuentas Ajax

Buscar :

Bruce

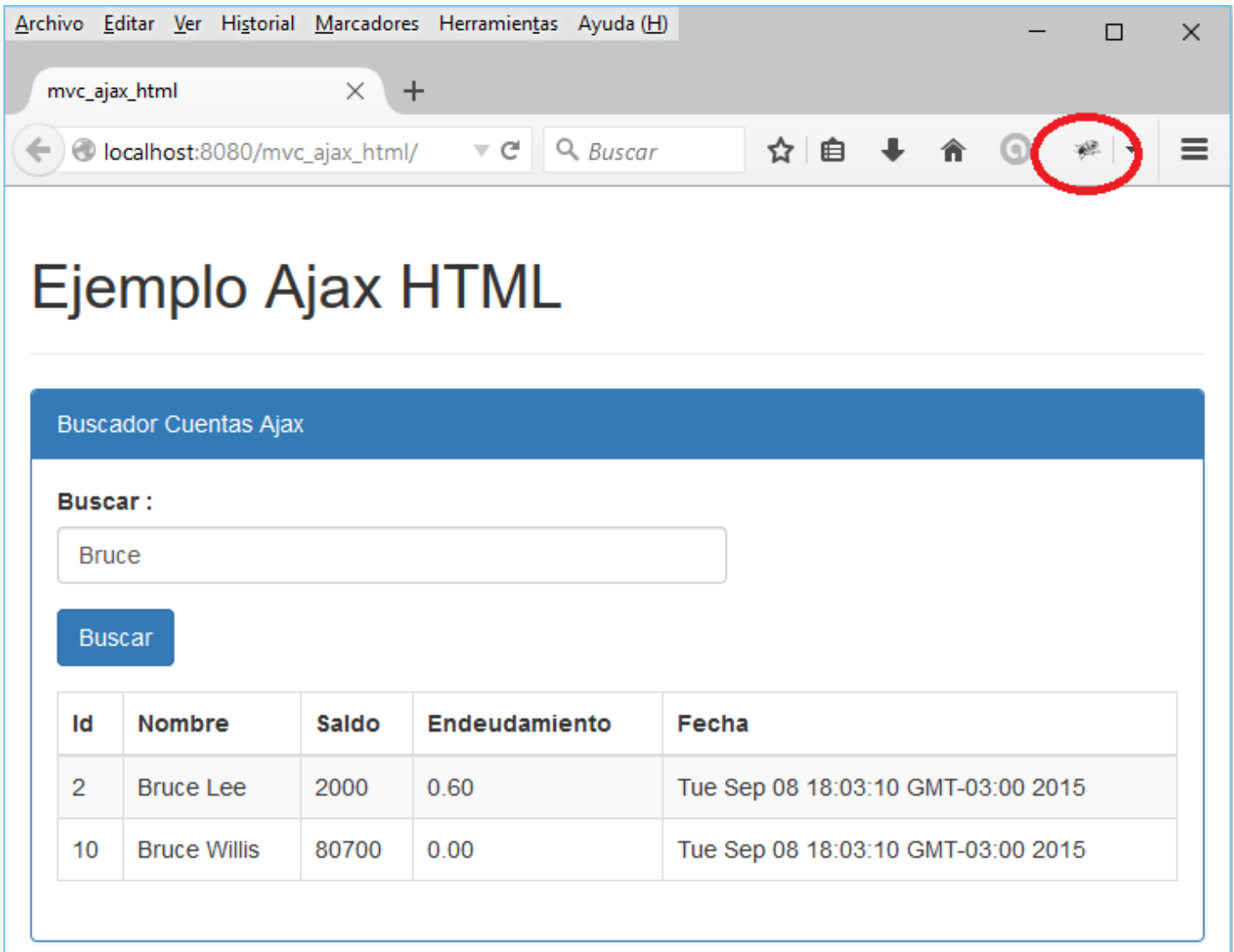
Buscar

Id	Nombre	Saldo	Endeudamiento	Fecha
2	Bruce Lee	2000	0.60	Tue Sep 08 18:03:10 GMT-03:00 2015
10	Bruce Willis	80700	0.00	Tue Sep 08 18:03:10 GMT-03:00 2015

- Observe cómo se realiza la búsqueda automáticamente ,mediante Ajax (Asynchronous JavaScript And XML) con el evento `onkeyup="buscarNombreAjax()"` (tipiar en una entrada input)
- Observe que se actualiza el contenedor `<div id="lista-cuentas">` con el resultado de la búsqueda sin actualizar (o recargar) la página mediante una petición asíncrona del tipo XMLHttpRequest y salida del tipo XHTML.

6. Habilitamos Firebug y su consola:

- Habilitamos primero Firebug:



7. Volvemos a repetir la búsqueda para ver el resultado en la consola
- Observamos las peticiones y respuestas HTTP Ajax en la consola de Firebug.

The screenshot shows a web browser window with the address bar at `localhost:8080/mvc_ajax_html/`. The page title is "Ejemplo Ajax HTML". Below the title is a blue header with the text "Buscador Cuentas Ajax". Underneath, there is a search form with the label "Buscar :", a text input containing "Andres", and a blue "Buscar" button. Below the form is a table with the following data:

Id	Nombre	Saldo	Endeudamiento	Fecha
1	Andres Guzmán	5000	0.30	Tue Sep 08 18:03:10 GMT-03:00 2015

Below the table, the Firebug console is open. The "Consola" tab is selected, and the "All" filter is chosen. The console shows a POST request to `http://localhost:8080/mvc_ajax_html/cuenta/buscar` with a status of 200 OK and a response time of 2ms. The response is a JSON object, and the "Respuesta" tab is selected, showing the following HTML structure:

```
<table style="width: 800px;" class="table table-striped table-hover table-bordered">
  <thead>
    <tr>
      <th>Id</th>
      <th>Nombre</th>
      <th>Saldo</th>
      <th>Endeudamiento</th>
      <th>Fecha</th>
    </tr>
  </thead>
  <tbody>
```


8. Abrir y estudiar la clase controladora `/src/main/java/com.bolsadeideas.ejemplos.cuenta.controllers/CuentaController.java`

```
package com.bolsadeideas.ejemplos.mvc.ajax.cuenta;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
@RequestMapping(value = "/cuenta")
public class CuentaController {

    @Autowired
    private ICuentaService cuentaService;

    @RequestMapping
    @ModelAttribute("cuentas")
    public List<Cuenta> listar() {
        return cuentaService.listar();
    }

    @RequestMapping(value = "/buscar", method = RequestMethod.POST)
    @ModelAttribute("cuentas")
    public List<Cuenta> buscar(@RequestParam String buscarTexto) {
        return cuentaService.buscarPorNombre(buscarTexto);
    }
}
```

9. Abrir y estudiar la vista jsp "cuenta.jsp"

```

<%@page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>mvc_ajax_html</title>
<!-- Bootstrap -->
<link rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
<!-- jquery -->
<script src="//code.jquery.com/jquery-2.1.4.min.js"></script>
</head>
<body>

    <div class="container">
        <div class="page-header">
            <h1>Ejemplo Ajax HTML</h1>
        </div>

        <div class="panel panel-primary">
            <div class="panel-heading">Buscador Cuentas Ajax</div>
            <div class="panel-body">
                <div class="container">
                    <div class="row">

                        <form id="cuenta" class="form-horizontal" style="width: 600px;">
                            <div class="form-group">
                                <label for="buscarTexto" class="col-sm-2 control-label">
                                    Buscar :
                                </label>
                                <div class="col-sm-10">
                                    <input id="buscarTexto" name="buscarTexto"
                                            onkeyup="javascript:buscarNombreAjax()"
                                            style="width: 400px;"
                                            class="form-control" />
                                </div>
                            </div>
                        </div>

                        <div class="form-group">
                            <div class="col-sm-offset-2 col-sm-10">
                                <input type="submit" value="Buscar"
                                        class="btn btn-primary" role="button" />
                            </div>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>

```

```

<div id="lista-cuentas">
  <table style="width: 700px;"
    class="table table-striped table-hover table-bordered">
    <thead>
      <tr>
        <th>Id</th>
        <th>Nombre</th>
        <th>Saldo</th>
        <th>Endeudamiento</th>
        <th>Fecha</th>
      </tr>
    </thead>
    <tbody>
      <c:forEach items="${cuentas}" var="cuenta">
        <tr>
          <td>${cuenta.id}</td>
          <td>${cuenta.nombre}</td>
          <td>${cuenta.saldo}</td>
          <td>${cuenta.nivelEndeudamiento}</td>
          <td>${cuenta.fechaRenovacion}</td>
        </tr>
      </c:forEach>
    </tbody>
  </table>
</div>
</div>
</div>
</div>
</div>
<script type="text/javascript">
  function buscarNombreAjax() {
    $.ajax({
      type      : "post",
      url       : "cuenta/buscar",
      data      : {buscarTexto: $('#buscarTexto').val()},
      success   : function(html) {
        $("#lista-cuentas").html(html);
      },
      error     : function(e){console.log(e);},
    });
    return false;
  }

  $(document).ready(function() {
    $("#cuenta").submit(function() {
      buscarNombreAjax();
      return false;
    });
  });
</script>
</body>
</html>

```

Hay dos cosas a destacar en el formulario de búsqueda, primero que el elemento **input** del nombre a buscar le asignamos un atributo de evento **onkeyup** y el valor la función JavaScript Ajax **buscarNombreAjax()**, de manera que a medida que escribamos el nombre (tipiando) llame la función JavaScript implementada con **jQuery** y realice la búsqueda Ajax (Asíncrona), y además asignamos el id del elemento **"buscarTexto"**:

```
Etc...  
<input id="buscarTexto" name="buscarTexto"  
onkeyup="javascript:buscarNombreAjax()" style="width: 400px;" class="form-  
control" />  
Etc...
```

El segundo punto a destacar es que **la tabla de las cuentas** se encuentra dentro de un contenedor **<div>** con **id = lista-cuentas** que es necesario para trabajar con Ajax y así poder asignar el contenido/resultado de forma asincrónico en la petición XMLHttpRequest por medio de Ajax manipulando los id de los elementos XHTML, en este caso el contenedor **div**:

```
<div id="lista-cuentas">  
  <table style="width: 700px;" class="table table-striped table-hover table-bordered">  
Etc...  
  </table>  
</div>
```

Ahora que tenemos listo el formulario, analicemos el código javascript implementado en la función con **jQuery**, entonces dentro de la vista al final observamos el siguiente código:

ETC...

```
<script type="text/javascript">
    function buscarNombreAjax() {
        $.ajax({
            type      : "post",
            url       : "cuenta/buscar",
            data      : {buscarTexto: $('#buscarTexto').val()},
            success   : function(html) {
                $("#lista-cuentas").html(html);},
            error     : function(e) {console.log(e);},
        });
        return false;
    }

    $(document).ready(function() {
        $("#cuenta").submit(function() {
            buscarNombreAjax();
            return false;
        });
    });
</script>
```

ETC...

Obtenemos el id del texto a buscar usando el framework **jQuery** Ajax/JavaScript:

\$('#buscarTexto').val(), luego con la función **\$.ajax()** realizamos la solicitud Ajax con los parámetros a enviar mediante POST al controlador **CuentaController**, específicamente al método handler **buscar(...)**.

El segundo punto a destacar es el **success**: que es el lugar donde queremos que nos muestre el resultado, aquí es donde se especifica que deberá insertar el resultado dentro del contenedor **div** con id **lista-cuentas** (que contiene la tabla de cuentas). Por lo tanto es en éste contenedor donde actualizará la tabla con las ceuntas/usuarios encontrados de la solicitud Ajax.

Ahora, estudiemos la vista **buscar.jsp** de la acción **buscar(...)** de controlador que recibe la petición Ajax:

webapp/WEB-INF/views/cuenta/buscar.jsp

```
<%@page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<table style="width: 800px;" class="table table-striped table-hover table-bordered">
  <thead>
    <tr>
      <th>Id</th>
      <th>Nombre</th>
      <th>Saldo</th>
      <th>Endeudamiento</th>
      <th>Fecha</th>
    </tr>
  </thead>
  <tbody>
    <c:forEach items="${cuentas}" var="cuenta">
      <tr>
        <td>${cuenta.id}</td>
        <td>${cuenta.nombre}</td>
        <td>${cuenta.saldo}</td>
        <td>${cuenta.nivelEndeudamiento}</td>
        <td>${cuenta.fechaRenovacion}</td>
      </tr>
    </c:forEach>
  </tbody>
</table>
```

Para terminar con el ejemplo, faltaría ver la clase de servicio encargada de crear los objetos cuentas de manera simple usando Map y realizar la búsqueda por nombre:

com.bolsadeideas.ejemplos.mvc.ajax.cuenta.CuentaService:

```
package com.bolsadeideas.ejemplos.mvc.ajax.cuenta;
ETC...
@Service
public class CuentaService implements ICuentaService{

    private Map<Long, Cuenta> cuentas = new ConcurrentHashMap<Long, Cuenta>();

    @Override
    @PostConstruct
    public void init() {
        Cuenta account = new Cuenta( new Long(0001), "Andres Guzmán",
            new BigDecimal("5000"), new BigDecimal("0.30"), new Date());
        cuentas.put(new Long(0001), account);

        account = new Cuenta( new Long(0002), "Bruce Lee", new BigDecimal("2000"),
            new BigDecimal("0.60"), new Date());
        cuentas.put(new Long(0002), account);

        ETC...
    }

    @Override
    public List<Cuenta> listar() {
        return new ArrayList<Cuenta>(cuentas.values());
    }

    @Override
    public Cuenta getPorId(Long id) {
        return this.cuentas.get(id);
    }

    @Override
    public List<Cuenta> buscarPorNombre(String searchString) {
        searchString = searchString == null? "": searchString;
        List<Cuenta> resultado = new ArrayList<Cuenta>();
        for (Cuenta cuenta : cuentas.values()) {
            if (cuenta.getNombre().toUpperCase()
                .contains(searchString.toUpperCase())) {
                resultado.add(cuenta);
            }
        }
        return resultado;
    }
}
```

Y la interface **ICuentaService** quedaría de la siguiente forma:

```
package com.bolsadeideas.ejemplos.mvc.ajax.cuenta;

import java.util.List;

public interface ICuentaService{
    public void init();
    public List<Cuenta> listar();
    public Cuenta getPorId(Long id);
    public List<Cuenta> buscarPorNombre(String searchString);
}
```

Ejercicio 4: Listas desplegables relacionadas en cascada Ajax con formato HTML



En el siguiente ejemplo veremos cómo relacionar listas desplegables combinadas en cascada usando Ajax (formato HTML). Para el ejercicio nos basaremos en países con sus ciudades, de tal forma que si seleccionamos un país específico en un select de países, nos despliegue más abajo una lista con todas sus ciudades pertenecientes.

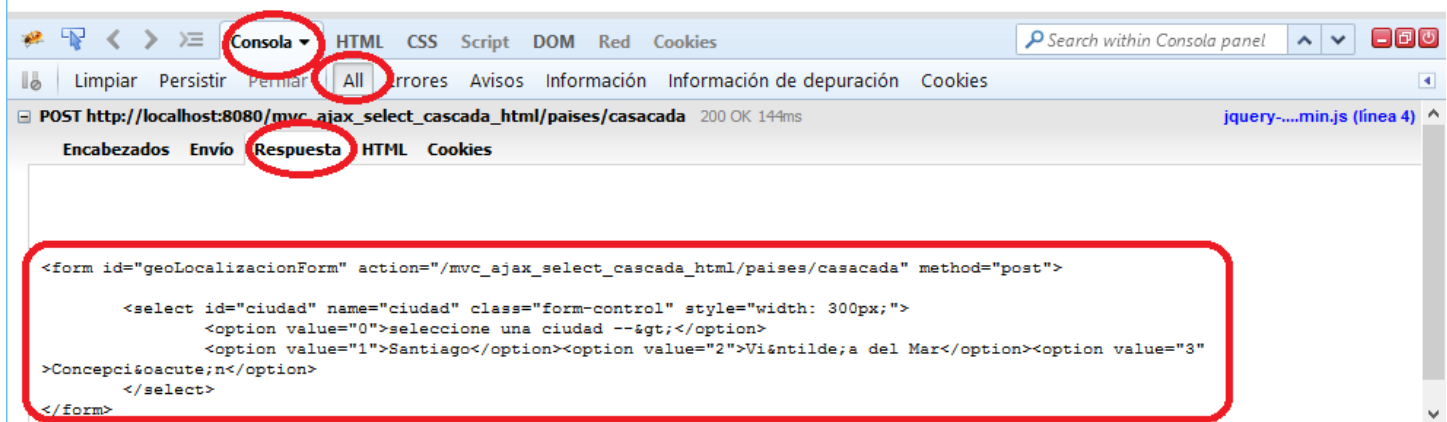
1. Clic derecho sobre el proyecto **Run As->Maven Clean** y **Run As->Maven Install**
2. Clic derecho **Maven->Update Project...**
3. Clic derecho sobre el proyecto **mvc_ajax_select_cascada_html-> Run As on Server**
4. Seleccionamos un país, ejemplo Chile.
 - Observamos que aparecen las ciudades relacionadas al país Chile:

Ejemplo Ajax HTML: Listas Desplegables en Cascada

Países/Ciudades

Países

Ciudades



Estudiemos primero la vista **países.jsp** ubicada es **webapp/WEB-INF/views/**, la cual contiene el formulario con los select de países y ciudades, además del código jQuery con las llamadas Ajax al controlador, entonces observe la vista **webapp/WEB-INF/views/paises.jsp**:

```
ETC...
<!-- jquery -->
<script src="//code.jquery.com/jquery-2.1.4.min.js"></script>
</head>
<body>
  <div class="container">
    <div class="page-header">
      <h1>Ejemplo Ajax HTML: Listas Desplegables en Cascada</h1>
    </div>
    <div class="panel panel-primary">
      <div class="panel-heading">Países/Ciudades</div>
      <div class="panel-body">
        <div class="container">
          <div class="row">
            <form:form modelAttribute="geoLocalizacionForm" method="post"
              cssClass="form-horizontal" role="form">

              <div class="form-group">
                <form:label for="pais" path="pais"
                  cssClass="col-sm-2 control-label">Países</form:label>
                <div class="col-sm-10">
                  <form:select onChange="javascript:cargarSelectPaíses()"
                    path="pais" style="width: 300px;" cssClass="form-control">
                    <form:option value="0" label="seleccione un pais -->" />
                    <form:options items="{países}" itemLabel="nombre"
                      itemValue="id" />
                  </form:select>
                </div>
              </div>
              <div class="form-group">
                <form:label for="ciudad" path="ciudad"
                  cssClass="col-sm-2 control-label">Ciudades</form:label>
                <div class="col-sm-10" id="lista-ciudades">
                  <form:select path="ciudad" style="width: 300px;"
                    cssClass="form-control">
                  </form:select>
                </div>
              </div>
            </form:form>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

```

<script type="text/javascript">
    function cargarSelectPaises() {

        $.ajax({
            type : "post",
            url : "países/cascada",
            data : {
                país : $('#país').val(),
            },
            success : function(html) {
                $('#lista-ciudades').html(html);
            },
            error : function(e) {
                console.log(e);
            },
        });
        return false;
    }
</script>
</body>
</html>

```

No vamos a entrar mucho en detalle en el form ya que supone que todos sabemos cómo trabaja y cómo se implementa, pero hay dos cosas a destacar, primero que el elemento select de los países le asignamos un atributo del evento **onchange** y el valor la función JavaScript Ajax **cargarSelectPaises()**, de manera que cuando se seleccione un país de la lista llame al evento y ejecute la función Ajax implementada con **JQuery**:

```

Etc...
<form:select onchange="javascript:cargarSelectPaises()" path="país"
            style="width: 300px;" cssClass="form-control">
    <form:option value="0" label="seleccione un país -->" />
    <form:options items="${países}" itemLabel="nombre" itemValue="id" />
</form:select>
Etc...

```

El segundo punto a destacar es que **en el elemento select de ciudad** observamos que se encuentra dentro de un contenedor **<div>** con **id = lista-ciudades** que es necesario para trabajar con Ajax y así poder asignar el contenido de forma asincrónico en la petición XMLHttpRequest por medio de Ajax manipulando los id de los elementos XHTML en este caso el contenedor **div**:

```
<div class="col-sm-10" id="lista-ciudades">
  <form:select path="ciudad" style="width: 300px;" cssClass="form-control">
  </form:select>
</div>
```

Ahora que tenemos listo el formulario, analicemos el código javascript implementado en la función con **jQuery**, entonces dentro de la vista al final observamos el siguiente código:

ETC...

```
<script type="text/javascript">
  function cargarSelectPaises() {

    $.ajax({
      type : "post",
      url : "países/cascada",
      data : {pais : $('#pais').val(),},
      success : function(html) {
        $("#lista-ciudades").html(html);
      },
      error : function(e) {
        console.log(e);
      },
    });

    return false;
  }
</script>
```

ETC...

Obtenemos el id del país seleccionado usando el framework **jQuery** Ajax/JavaScript: `$('#pais').val()`, luego con la función **\$.ajax()** realizamos la solicitud Ajax con los parámetros a enviar mediante POST al controlador **com.bolsadeideas.ejemplos.mvc.ajax.controllers.PaisesController**, específicamente al método handler **cascada(...)**.

El segundo punto a destacar es el **success**: que es el lugar donde queremos que nos muestre el resultado, aquí es donde se especifica que deberá insertar el resultado dentro del contenedor **div** con el id **lista-ciudades** configurado anteriormente en el elemento ciudad del formulario dentro de la vista (países.jsp), por lo tanto es en éste contenedor donde mostrará la lista select de las ciudades pertenecientes al país seleccionado.

Ahora veamos el controlador **PaísesController**, comencemos por el método **listar()**, que es donde se pasan los países a la vista del formulario ajax:

```
@RequestMapping
public Model listar(Model model) {
    model.addAttribute("países", geoService.listarPaíses());
    model.addAttribute("geoLocalizacionForm", new GeoLocalizacionForm());
    return model;
}
```

Observamos que poblamos el elemento **select pais** con el listado de **países** usando el modelo la clase **GeoLocalizacionService**.

Finalmente, ya que tenemos toda la base y el grueso implementado, ahora sólo falta analizar el método **cascada(...)** del controlador que recibirá y manejará la petición POST Ajax, en este caso con salida/formato HTML, entonces en el mismo controlador **PaísesController**:

```
/* ...etc... */
@Controller
@RequestMapping(value = "/países")
public class PaísesController {
    /* ...etc... */

    @RequestMapping(value = "/cascada", method = RequestMethod.POST)
    public Model cascada(Model model, GeoLocalizacionForm form) {
        model.addAttribute("ciudades", geoService.buscarCiudadesPorPaisId(form.getPais()));
        model.addAttribute("geoLocalizacionForm", new GeoLocalizacionForm());
        return model;
    }
}
```

A simple vista, se observa que obtenemos el objeto comando del formulario **GeoLocalizacionForm** que contiene el Id del país seleccionado enviado por Ajax en la función javascript **cargarSelectPaíses()**:

```
pais : $('#pais').val()
```

Luego en el método handler ajax **cascada(...)** del controlador creamos el elemento del formulario select producto y lo poblamos con las ciudades correspondientes a ese país, usando la clase modelo y su método **buscarCiudadesPorPaisId(Long paisId)** que recibe el id del país como parámetro.

Ahora, estudiemos la vista **cascada.jsp** de la acción **cascada** que recibe la petición Ajax:

webapp/WEB-INF/views/paises/cascada.jsp

```
<%@page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<form:form modelAttribute="geoLocalizacionForm">
    <form:select path="ciudad" style="width: 300px;"
        cssClass="form-control">
        <form:option value="0" label="seleccione una ciudad -->" />
        <form:options items="${ciudades}" itemLabel="nombre" itemValue="id" />
    </form:select>
</form:form>
```

Simplemente se encarga de imprimir el elemento select de ciudad con las ciudades del país seleccionado.

Para terminar con el ejemplo, faltaría ver la clase de servicio encargada de crear los objetos relacionados a países y ciudades y sus relaciones, de manera simple y sencilla usando Map:

com.bolsadeideas.ejemplos.mvc.ajax.models.services.GeoLocalizacionService:

```
package com.bolsadeideas.ejemplos.mvc.ajax.models.services;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;
import javax.annotation.PostConstruct;
import org.springframework.stereotype.Service;
import com.bolsadeideas.ejemplos.mvc.ajax.models.dominio.Ciudad;
import com.bolsadeideas.ejemplos.mvc.ajax.models.dominio.Pais;

@Service
public class GeoLocalizacionService implements IGeoLocalizacionService {

    private Map<Long, Pais> paises = new ConcurrentHashMap<Long, Pais>();
    private Map<Long, Ciudad[]> ciudades = new ConcurrentHashMap<Long, Ciudad[]>();

    @Override
    @PostConstruct
    public void init() {
        Pais pais = new Pais(new Long(1L), "Chile");
        paises.put(new Long(1L), pais);
        ciudades.put(new Long(1L), new Ciudad[] { new Ciudad(1L, "Santiago"),
                                                    new Ciudad(2L, "Viña del Mar"),
                                                    new Ciudad(3L, "Concepción") });

        pais = new Pais(new Long(2L), "España");
        paises.put(new Long(2L), pais);
        ciudades.put(new Long(2L), new Ciudad[] { new Ciudad(1L, "Madrid"),
                                                    new Ciudad(2L, "Barcelona"),
                                                    new Ciudad(3L, "Sevilla") });

        pais = new Pais(new Long(3L), "México");
        paises.put(new Long(3L), pais);
        ciudades.put(new Long(3L), new Ciudad[] {
            new Ciudad(1L, "Ciudad de México"), new Ciudad(2L, "Puebla"),
            new Ciudad(3L, "Guadalajara") });

        pais = new Pais(new Long(4L), "Argentina");
        paises.put(new Long(4L), pais);
        ciudades.put(new Long(4L), new Ciudad[] { new Ciudad(1L, "Buenos Aires"),
                                                    new Ciudad(2L, "Córdoba"),
                                                    new Ciudad(3L, "Mendoza") });
    }
}
```

```

    pais = new Pais(new Long(5L), "Perú");
    paises.put(new Long(5L), pais);
    ciudades.put(new Long(5L), new Ciudad[] { new Ciudad(1L, "Lima"),
                                              new Ciudad(2L, "Arequipa"),
                                              new Ciudad(3L, "Cuzco") });

    pais = new Pais(new Long(6L), "Colombia");
    paises.put(new Long(6L), pais);
    ciudades.put(new Long(6L), new Ciudad[] { new Ciudad(1L, "Bogotá"),
                                              new Ciudad(2L, "Medellín"),
                                              new Ciudad(3L, "Cali") });

    pais = new Pais(new Long(7L), "Uruguay");
    paises.put(new Long(7L), pais);
    ciudades.put(new Long(7L), new Ciudad[] { new Ciudad(1L, "Montevideo"),
                                              new Ciudad(2L, "Colonia"),
                                              new Ciudad(3L, "San José") });
}

@Override
public List<Pais> listarPaises() {
    return new ArrayList<Pais>(paises.values());
}

@Override
public Pais getPaisPorId(Long id) {
    return this.paises.get(id);
}

@Override
public List<Ciudad> buscarCiudadesPorPaisId(Long paisId) {
    return Arrays.asList(this.ciudades.get(paisId));
}
}

```

Y la interface IGeoLocalizacionService quedaría de la siguiente forma:

```

package com.bolsadeideas.ejemplos.mvc.ajax.models.services;

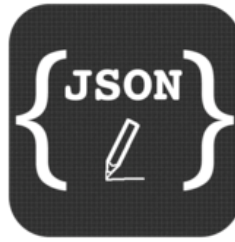
import java.util.List;
import com.bolsadeideas.ejemplos.mvc.ajax.models.dominio.Ciudad;
import com.bolsadeideas.ejemplos.mvc.ajax.models.dominio.Pais;

public interface IGeoLocalizacionService {
    public void init();
    public List<Pais> listarPaises();
    public Pais getPaisPorId(Long id);
    public List<Ciudad> buscarCiudadesPorPaisId(Long paisId);
}

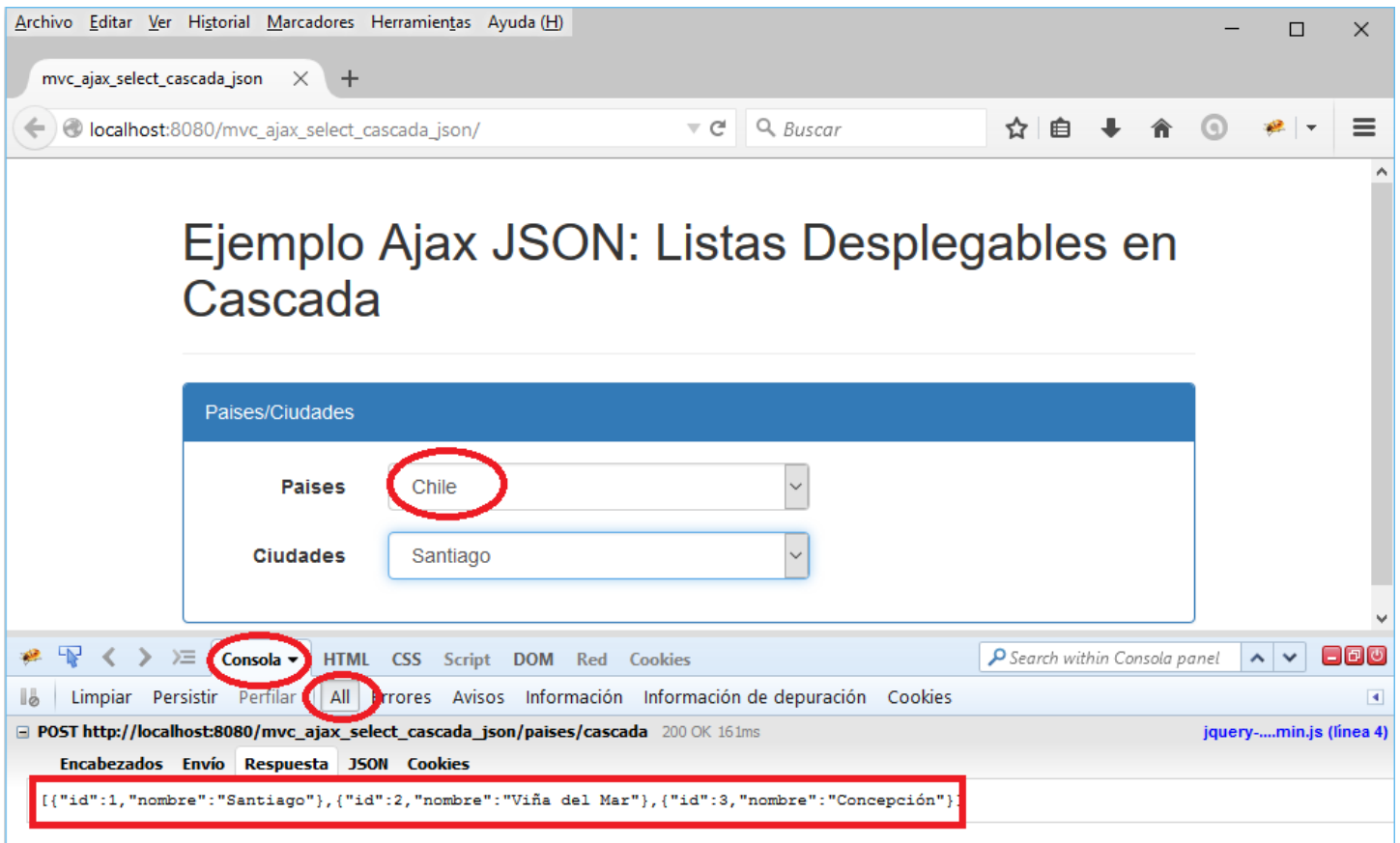
```

Ejercicio 5: Listas desplegables en cascada Ajax con formato JSON

Ahora vamos a hacer algunos ajustes al ejemplo anterior para que la implementación sea a través de json en vez de html/jsp.



1. Clic derecho sobre el proyecto **Run As->Maven Clean** y **Run As->Maven Install**
2. Clic derecho **Maven->Update Project...**
3. Clic derecho sobre el proyecto **mvc_ajax_select_cascada_json-> Run As on Server**



Comenzamos con analizar y observar el código javascripts correspondiente e implementado con **jQuery** y **JSON**. Veamos cómo es nuestro nuevo javascripts, entonces dentro del directorio "**webapp/WEB-INF/views/**" observemos la vista: "**países.jsp**":

ETC...

ETC...

```
<script type="text/javascript">
  function cargarSelectPaíses() {
    $.ajax({type : "post",
      url : "países/cascada",
      data : JSON.stringify({pais : $('#pais').val(),}),
      dataType : 'json',
      contentType : "application/json; charset=utf-8",
      success : function(dataJson) {
        $("select[name=ciudad]").empty();
        $("select[name=ciudad]")
          .append("<option value="">Seleccione un producto --</option>");
        $.each(dataJson, function(index, item) {
          $("select[name=ciudad]").append(
            "<option value="" + item.id + ""> + item.nombre
            + "</option>");});});
        error : function(e) {
          console.log(e);
        },
      });
    return false;
  }
</script>
```

Similar al ejemplo anterior usando HTML (como salida), Obtenemos el id del país seleccionado usando el framework Ajax/JavaScript **jQuery**, luego con la función `$.ajax()`, hacemos la solicitud o petición Ajax con los parámetros que vamos a enviar mediante POST al controlador **PaísesController**, método handler **cascada(...)** y el Ajax Context con formato **JSON**.

El segundo punto a destacar es la función **success callback**: que es el lugar donde recuperamos el objeto **JSON** mediante petición Ajax, y poblamos la lista desplegable select con los datos obtenidos del resultado **JSON**, por lo tanto es aquí donde se poblará la lista select con las ciudades pertenecientes al país seleccionado usando **DOM** o **Document Object Model** (Modelo de Objetos del Documento).

Luego, observemos el método **cascada(...)** del controlador que recibe y maneja la petición Ajax **JSON**:

```
/* ...etc... */
@Controller
@RequestMapping(value = "/países")
public class PaisController {
    /* ...etc... */

    @RequestMapping(value = "/cascada", method = RequestMethod.POST)
    public @ResponseBody List<Ciudad> cascada(@RequestBody GeoLocalizacionForm form) {
        return geoService.buscarCiudadesPorPaisId(form.getPais());
    }
}
```

Simplemente, si el formato es **JSON**, entonces usamos la anotación **@ResponseBody** para generar la salida de los datos con el formato JSON.

Otro punto importante a tener en cuenta, es tener el pom.xml con las dependencias maven de Jackson JSON Mapper:

```
ETC...
<!-- Jackson JSON Mapper -->
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.6.1</version>
</dependency>
ETC...
```

Resumen

Nos adentramos en el uso de AJAX con jQuery, vimos un ejemplo real y típico de un diseño de un sistema de listas desplegables relacionadas.

A continuación presentaremos una tarea desafío que busque aplicar todos los conceptos vistos en este material

Fin.

Envía tus consultas a los foros!

Aquí es cuando debes sacarte todas las dudas haciendo consultas en los foros correspondientes