

AJAX

JavaScript Asíncrono y XML

La palabra **Ajax** viene del inglés, **Asynchronous JavaScript And XML**. Básicamente implica ejecutar un requerimiento HTTP de manera asíncrona usando JavaScript y luego, actualizar el DOM HTML con los datos recibidos como respuesta.

Las aplicaciones AJAX se ejecutan en el cliente y mantienen comunicación asíncrona con el servidor, permitiendo realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad y la velocidad en las aplicaciones.

Normalmente las invocaciones a funciones de AJAX se hacen en JavaScript y el acceso a los datos se realiza mediante el objeto **XMLHttpRequest**, disponible en todos los navegadores. No es necesario que el contenido asíncrono esté formateado en XML, hoy en día es más común usar JSON.

El uso de este objeto **XMLHttpRequest** para que funcione en todos los navegadores era tedioso y por ello aparecieron múltiples librerías JavaScript como **jQuery**, **Prototype**, **Mootools** que simplifican esta comunicación asíncrona mediante funciones individuales que cubren las peculiaridades de cada navegador.

En síntesis usaremos **JavaScript** casi en su totalidad con **JSON** y ocasionalmente algunas funciones de **jQuery** para simplificar.

JSON

JavaScript Object Notation

¿Qué es JSON?

JSON (JavaScript Object Notation) es un formato estándar para intercambio de datos entre aplicaciones. Los programas escritos en Java/EE, Ruby, JavaScript, C#/.Net, PHP, etc. pueden fácilmente consumir y producir datos JSON porque es independiente del lenguaje y de la plataforma.

JSON fue creado por Douglas Crockford in 2001, y está especificado en la RFC 4627 (<http://tools.ietf.org/html/rfc4627>).

Existen varias APIs relacionadas con JSON que facilitan su uso desde cualquier lenguaje e IDE.

Algunas librerías JAVA populares para JSON son:

- **JSON.Simple**, <https://code.google.com/p/json-simple/>
- **Jackson**, <http://jackson.codehaus.org> (mejor performance)
- **Google GSON**, <http://code.google.com/p/google-json/>
- **org.json** (de Douglas Crockford), <http://www.json.org/java>

JSON es un formato de datos simple que tiene tres estructuras de datos básicas:

- (1) Pares de datos de la forma **nombre/valor** (o clave/valor).
- (2) **Objetos**
- (3) **Arreglos**

JSON

JavaScript Object Notation

En cuanto a los valores, **JSON** define los siguientes tipos:

- number (entero o punto flotante)
- string (entre comillas, “” o ‘’)
- boolean (true o false)
- array (entre corchetes, [])
- object (entre llaves, {})
- Null

Estructuras Básicas

(1) Pares nombre/valor

Un par de la forma nombre/valor se define entre llaves con una propiedad string y un valor. El valor puede ser uno de los tipos de datos detallados arriba.

```
{ "nombre": "Juana" }
```

```
{ "temperatura": -1.5 }
```

JSON

JavaScript Object Notation

(2) Objetos

Un objeto es una colección desordenada de pares nombre/valor.

```
{ "widget": {  
  "debug": "on",  
  "window": {  
    "title": "Sample Widget",  
    "name": "main_window",  
    "width": 500,  
    "height": 500},  
  "image": {  
    "src": "Images/Sun.png",  
    "name": "sun1",  
    "alignment": "center"}  
  }  
}
```

Arreglos

Un arreglo es una colección ordenada de valores

```
{ "alumnos" : [  
  { "nombre": "Juana", "apellido": "Díaz", "edad": 35 },  
  { "nombre": "Fausto", "apellido": "Lizaga", "edad": 32 } ]  
}
```

Archivos JSON

Librería Gson

Gson es una librería Java que puede ser usada para convertir objetos Java en su representación JSON. También puede usarse para convertir un String JSON a su equivalente objeto Java.

Gson es un proyecto open-source project hosteado en <http://code.google.com/p/google-gson>.

Provee mecanismos parecidos al toString() y constructores (factory) para convertir Java a JSON y viceversa.

Soporta objetos complejos

Generate salidas JSON con formatos bien entendibles.

Para usarla se debe agregar la dependencia en el archivo **POM.XML**

```
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.7</version>
</dependency>
```

Archivos JSON

Librería Gson

El código muestra el uso de la librería **Gson** de Google para convertir un objeto java a un String con formato **JSON** y éste a un objeto **JsonObject** a partir del cual se pueden recuperar sus propiedades.

```
package modelo;
import com.google.gson.Gson;
import com.google.gson.JsonObject;

public class ParseandoJSON {
    public static void main(String[] args) {
        List<String> m = new LinkedList<String>();
        m.add("jrrios@gmail.com");
        m.add("juanarios@hotmail.com");
        Alumno alu = new Alumno("Rios", "Juana", new Date(), m);
        Gson gson = new Gson();
        // toJson serializa el objeto alu a su representación JSON equivalente
        String strJson = gson.toJson(alu);
        System.out.println("String JSON: " + strJson);

        // fromJson convierte JSON a JsonObject
        JsonObject dataJson = gson.fromJson(strJson, JsonObject.class);
        System.out.println("Propiedades del JSON: ");
        System.out.println(dataJson.get("nombre"));
        System.out.println(dataJson.get("apellido"));
        System.out.println(dataJson.get("nacimientto"));
        System.out.println(dataJson.get("mails"));

        // fromJson convierte JSON a un objeto Java
        Alumno alumno = gson.fromJson(strJson, Alumno.class);
        System.out.println("Objeto java Alumno: " + alumno);
    }
}
```

```
public class Alumno {
    private String apellido;
    private String nombre;
    private Date nacimiento;
    private List<String> mails;

    public Alumno(String apellido, String nombre, Date nacimiento, List<String> mails) {
        this.apellido = apellido;
        this.nombre = nombre;
        this.nacimiento = nacimiento;
        this.mails = mails;
    }

    public String getApellido() {
        return apellido;
    }

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }

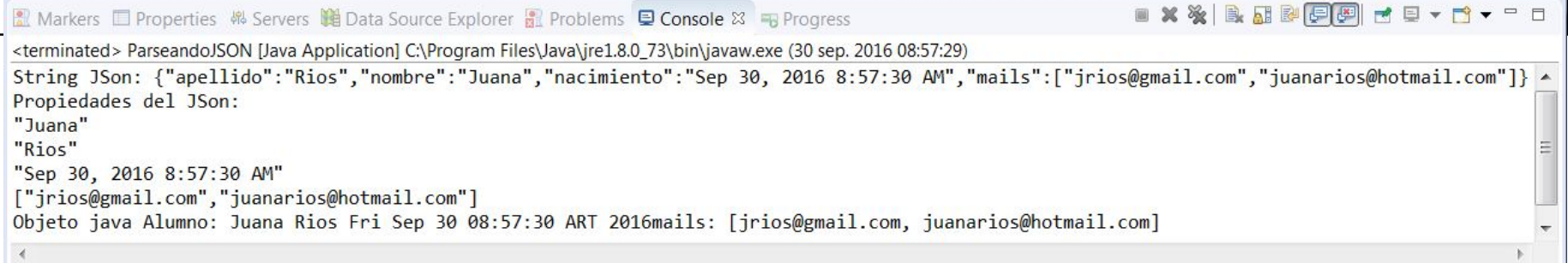
    // ...
}
```

Archivos JSON

Librería Gson

```
package modelo;
//imports
public class TestJSON {
    public static void main(String[] args) {
        List<String> m = new LinkedList<String>();
        m.add("jrios@gmail.com");
        m.add("juanarios@hotmail.com");
        Alumno alu = new Alumno("Rios", "Juana", new Date(), m);
        Gson gson = new Gson();
        // serializa el objeto alu a su representación JSON equivalente
        String strJson = gson.toJson(alu);
        System.out.println("String JSon: " + strJson);
        // fromJson convierte JSON a JsonObject
        JsonObject dataJson = gson.fromJson(strJson, JsonObject.class);
        System.out.println("Propiedades del JSon: ");
        System.out.println(dataJson.get("nombre"));
        System.out.println(dataJson.get("apellido"));
        System.out.println(dataJson.get("nacimiento"));
        System.out.println(dataJson.get("mails"));
        // fromJson convierte JSON a un objeto Java
        Alumno alumno = gson.fromJson(strJson, Alumno.class);
        System.out.println("Objeto java Alumno: " + alumno);
    }
}
```

Salida



```
<terminated> ParseandoJSON [Java Application] C:\Program Files\Java\jre1.8.0_73\bin\javaw.exe (30 sep. 2016 08:57:29)
String JSon: {"apellido":"Rios","nombre":"Juana","nacimiento":"Sep 30, 2016 8:57:30 AM","mails":["jrios@gmail.com","juanarios@hotmail.com"]}
Propiedades del JSon:
"Juana"
"Rios"
"Sep 30, 2016 8:57:30 AM"
["jrios@gmail.com","juanarios@hotmail.com"]
Objeto java Alumno: Juana Rios Fri Sep 30 08:57:30 ART 2016mails: [jrios@gmail.com, juanarios@hotmail.com]
```

Archivos JSON

Librería JSon Simple

JSon simple es otra librería java simple para trabajar con JSON. Se puede usar JSON.simple para crear objetos JSON y parsearlos.

JSon Simple is an open-source project hosted at <https://github.com/fangyidong/json-simple>

Provee las clases `JSONParser` y `JSONObject` que facilitan la lectura e iteración de los elementos del JSON. Esta librería provee mas funcionalidades para parsear texto JSON y se mantiene simple.

Para usarla se debe agregar la dependencia en el archivo **POM.XML**

```
<dependency>
    <groupId>com.googlecode.json-simple</groupId>
    <artifactId>json-simple</artifactId>
    <version>1.1.1</version>
</dependency>
```


Archivos JSON

Librería JSon Simple

```
package modelo;
//imports
public class TestJSON {
    public static void main(String[] args) {
        List<String> m = new LinkedList<String>(); String clave=null;
        m.add("jrios@gmail.com");m.add("juanarios@hotmail.com");

        JSONObject obj = new JSONObject();
        obj.put("nombre","Juana");
        obj.put("apellidp", new String("Rios"));
        obj.put("nacimiento",new Date());
        obj.put("mails",m);
        String strJson = obj.toJSONString();
        System.out.println("String JSon: " + strJson);

        Set<String> claves = obj.keySet();
        Iterator<String> it = claves.iterator();
        while (it.hasNext()) {
            clave = it.next();
            System.out.println(clave+ ":" + obj.get(clave));
        }
    }
}
```

Salida



Markers Properties Servers Data Source Explorer Problems Console Progress

<terminated> ParseandoJSON [Java Application] C:\Program Files\Java\jre1.8.0_73\bin\javaw.exe (29 sep. 2016 23:20:58)

String JSon: {"mails":["jrios@gmail.com","juanarios@hotmail.com"],"apellidp":"Rios","nombre":"Juana","nacimiento":Thu Sep 29 23:20:58 ART 2016}

Propiedades del JSon:

mails:[jrios@gmail.com, juanarios@hotmail.com]

apellidp:Rios

nombre:Juana

nacimiento:Thu Sep 29 23:20:58 ART 2016

AJAX

Envío de texto desde el Servidor al Cliente

```
package servlets;
// imports
@WebServlet("/RetornaTexto")
public class RetornaTexto extends HttpServlet {
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response)
        String text = "algun texto";
        response.setContentType("text/plain");
        response.getWriter().write(text);
    }
}
```



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Texto Plano</title>
<script src="http://code.jquery.com/jquery-latest.min.js"></script>
<script>
    function consultarServidor(){
        $.get("RetornaTexto", function(responseText) {
            document.getElementById("somediv").innerHTML=responseText;
        });
    };
</script>
</head>
<body>
    <button onClick="consultaServidor()">Presione para recibir texto plano</button>
    <div id="somediv"></div>
</body>
</html>
```

Esta función se ejecuta cuando se presiona el botón

Se ejecuta un request GET AJAX a la URL del servlet RetornaTexto y ejecuta la función AJAX con responseText

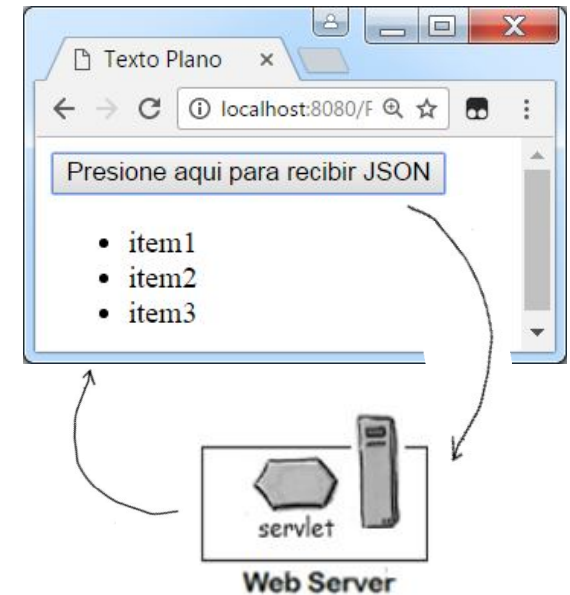
Ubica el elemento HTML DOM con ID "somediv" y setea su contenido con el texto de la respuesta

AJAX

Envío de JSON desde el Servidor al Cliente (List<String>)

```
package servlets;
// imports
@WebServlet("/RetornaJSON")
public class RetornaJSON extends HttpServlet {
    protected void doGet(HttpServletRequest request,
                          HttpServletResponse response)
        List<String> list = new ArrayList<String>();
        list.add("item1");list.add("item2");list.add("item3");
        String json = new Gson().toJson(list);
        response.setContentType("application/json");
        response.getWriter().write(json);
    }
}
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head>
<script src="http://code.jquery.com/jquery-latest.min.js"></script>
<script>
function consultarServidor(){
    $.get("RetornaJSON", function(responseJson) {
        var html = "<ul>";
        for(var i=0; i< responseJson.length; i++){
            html += "<li>"+responseJson[i]+"</li>";
        }
        html += "</ul>";
        document.getElementById("somediv").innerHTML=html;
    });
};</script></head>
<body>
    <button onClick="consultarServidor()">Presione aquí para recibir JSON</button>
    <div id="somediv"></div>
</body></html>
```



Esta función se ejecuta cuando se presiona el botón

Se ejecuta un request GET AJAX a la URL del servlet RetornaJSON y ejecuta la función AJAX con responseJson

Crea una variable upara ir armando el HTML. Le agrega un HTML y despues cada uno de los

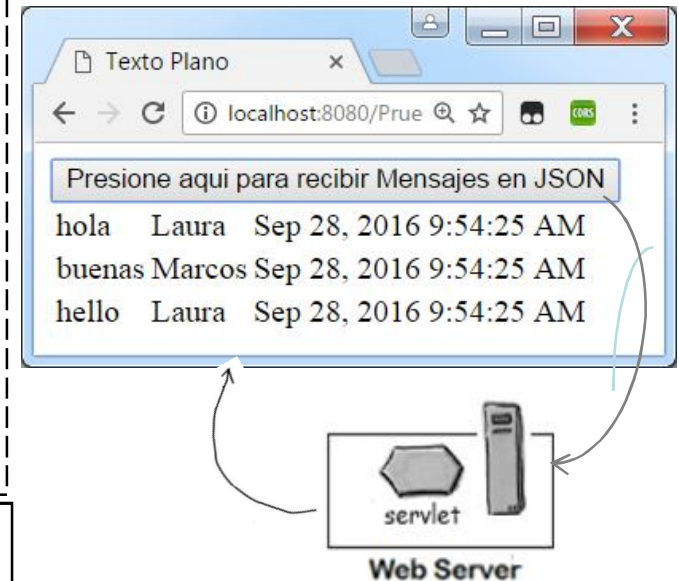
Pone el fragmento HTML creado como contenido del div "somediv"

AJAX

Envío de JSON desde el Servidor al Cliente(List<Mensaje>)

```
package servlets;
@WebServlet("/RetornaMensajesJSON")
public class RetornamensajesJSON extends HttpServlet {
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response)
        List<Mensaje> mensajes = new ArrayList<Mensaje>();
        mensajes.add(new Mensaje("hola","Laura", new Date()));
        mensajes.add(new Mensaje("buenas","Marcos", new Date()));
        mensajes.add(new Mensaje("hello","Laura", new Date()));
        String json = new Gson().toJson(mensajes);
        response.setContentType("application/json");
        response.setCharacterEncoding("UTF-8");
        response.getWriter().write(json); }}
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN". . . <head>
<script src="http://code.jquery.com/jquery-latest.min.js"></script>
<script>
function consultarServidor(){
$.get("RetornaMensajesJSON", function(responseJson) {
    var html = "<table>";
    for(var i=0; i< responseJson.length; i++){
        var mensaje = responseJson[i];
        html += "<tr>";
        html += "    <td>"+mensaje.msj+"</td>";
        html += "    <td>"+mensaje.usuario+"</td>";
        html += "    <td>"+mensaje.dia+"</td>";
        html += "</tr>";
    }
    html += "</table>";
    document.getElementById("somediv").innerHTML=html;
});
};/script></head>
<body> <button id="somebutton">Presione para recibir texto plano</button>
    div id="somediv"></div>
</body></html>
```



Esta función se ejecuta cuando se presiona el botón

Se ejecuta un request GET AJAX a la URL del servlet RetornaJSON y ejecuta la función AJAX con responseJson

Crea una variable upara ir armando el HTML. Le agrega un HTML y despues cada uno de los

Pone el fragmento HTML creado como contenido del div "somediv"

AJAX

Envío de JSON desde el Cliente al Servidor (Arreglo)

```
package servlets;
// imports
@WebServlet("/RecibeJSON")
public class RecibeJSON extends HttpServlet {
    protected void doGet(HttpServletRequest request,
                          HttpServletResponse response)
    //JsonObject data = new Gson().fromJson(request.getReader(), JsonObject.class);
    JSONArray data = new Gson().fromJson(request.getReader(), JSONArray.class);
    for (Object o : data) {
        System.out.println("Datos leídos por el Servlet "+o);
    }
}
```

```
<%@ page contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head>
<title>Texto Plano</title>
<script src="http://code.jquery.com/jquery-latest.min.js"></script>
<script>
    var params = [{ nombre: "Juana"}, {apellido: "Rios"};
    $.post( "RecibeJSON",
            JSON.stringify(params)
    ).done(function() {
        alert( 'Datos guardados correctamente' );
    }).fail(function() {
        alert( 'Ocurrió un error' );
    });
</script>
</head><body>
    <button id="somebutton">Presione para recibir texto plano</button>
    <div id="somediv"></div>
</body></html>
```

Se hace un requerimiento por POST al servlet RecibeJson

El doner() es para especificar que se hace si si el requerimiento es exitoso y el fail si falla

Archivos JSON

Lectura de datos JSON desde un Servlet contra una URL

```
package todoJson;
...
@WebServlet("/LeeJSONClima")
public class LeeJSONClima extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException {

        URL url = new URL("http://clima.info.unlp.edu.ar/last");
        BufferedReader in = new BufferedReader(new InputStreamReader(url.openStream(), "UTF-8"));
        String documentoJSON = in.readLine();

        JSONParser jp = new JSONParser();
        try {
            JSONObject json = (JSONObject) jp.parse(documentoJSON);

            Set<String> claves = json.keySet();
            Iterator<String> it = claves.iterator();
            while (it.hasNext()) {
                String clave = it.next();
                System.out.println(clave + ":" + json.get(clave));
            }
        } catch (ParseException e) {
            e.printStackTrace();
        }
    }
}
```

