



Servlets y JSP

Sesión 12: Comunicación con clientes ricos y AJAX



Puntos a tratar

- Clientes ricos
- AJAX



Conexión HTTP con servlets

- Podemos utilizar un objeto `URLConnection` para conectarnos desde nuestra aplicación Java a un servlet mediante HTTP
- Al crear la URL utilizamos la dirección a la que está mapeado el servlet
- Podremos hacer que el servlet genere contenido con cualquier codificación y formato
- La aplicación cliente deberá entender este formato de los datos, para poder leer la información que nos devuelve el servlet



Lectura de objetos

- Como ejemplo vamos a ver un servlet que devuelve como contenido un objeto Java

```
MiClase result = generaObjetoResultante();
response.setContentType("application/x-java-serialized-object");
OutputStream out = response.getOutputStream();
ObjectOutputStream oos = new ObjectOutputStream(out);
oos.writeObject(result);
oos.flush();
```

- Desde la aplicación cliente podremos leer el objeto devuelto

```
URL url = new URL("http://localhost:8080/aplic/servlet/MiServlet");
URLConnection con = url.openConnection();
InputStream in = con.getInputStream();
ObjectInputStream ois = new ObjectInputStream(in);
MiClase obj = (MiClase)ois.readObject();
```



Envío de objetos

- También podemos enviar objetos u otros datos desde la aplicación cliente al servlet
 - Mediante protocolo HTTP utilizando el bloque de contenido

```
URLConnection con = url.openConnection();
con.setDoOutput(true);
ByteArrayOutputStream baos = new ByteArrayOutputStream();
ObjectOutputStream oos = new ObjectOutputStream(baos);
oos.writeObject(result);
con.setRequestProperty("Content-Length",
    String.valueOf(baos.size()));
con.setRequestProperty("Content-Type",
    "application/x-java-serialized-object");
baos.writeTo(con.getOutputStream());
```



AJAX

- *AJAX (Asynchronous Javascript And Xml)*
 - Técnica de desarrollo que permite obtener información desde un navegador sin recargar la página
- Utiliza varias tecnologías:
 - HTML y CSS
Presentar la información
 - XML
Obtener la información
 - DOM y Javascript
Analizar la información



Pasos para utilizar AJAX

- Realizar una petición HTTP al servidor desde Javascript
 - Se utiliza el objeto `XMLHttpRequest`
- Obtener un documento XML como respuesta
 - Contendrá los datos solicitados al servidor
- Extraer la información del documento XML
 - Analizándolo mediante el DOM de Javascript
- Actualizar el documento HTML del navegador
 - Se puede utilizar el DOM para modificar este documento e introducir en él los datos recibidos



Crear petición HTTP

- Creamos el objeto para hacer la petición
 - Depende del tipo de navegador (Firefox / IE)

```
function verMensajes() {  
    //Preparar objeto para lanzar peticion  
    if (window.XMLHttpRequest) { //Firefox,etc  
        peticion = new XMLHttpRequest();  
    } else if (window.ActiveXObject) { //Explorer  
        peticion = new ActiveXObject("Microsoft.XMLHTTP");  
    }  
}
```

- Establecemos una función callback
 - Será llamada cuando se reciba la respuesta

```
//a quien llamar cuando el servidor responda  
peticion.onreadystatechange = atenderPeticion;
```




Efectuar la petición

- Efectuamos la petición con:

```
//lanzar la petición propiamente dicha
peticion.open("GET",
    "http://localhost:7001/ChatXml/chat/listaMensajesXml.jsp",
    true);
peticion.send(null);
```

- El método `open` toma como parámetros:
 - Método (GET, POST, ...)
 - URL a la que conectamos
 - Petición asíncrona (`true/false`)
 - Si no es asíncrona quedaría bloqueado hasta recibir la respuesta
- El método `send` toma como parámetro la URL a la que conectamos
 - Si ya se ha especificado esta URL en `open`, se puede poner `null`
- NOTA: Por motivos de seguridad, sólo se podrá conectar al mismo servidor del que se descargó la página que contiene el código Javascript



Recibir la respuesta

- Cuando se reciba la respuesta, se llamará a la función *callback* especificada

```
function atenderPetición() {  
    if (petición.readyState == 4) {  
        //analizar respuesta  
        if (petición.status!=200) {  
            alert("ha habido un error");  
            return;  
        }  
    }  
}
```

- Nos interesará sólo el caso en que:
 - La petición esté completada
readyState será 4
 - No se hayan producido errores
status tendrá código 200 (200 OK)

Estados de la petición (readyState)	
0	<i>No inicializada</i>
1	<i>Cargando</i>
2	<i>Cargada</i>
3	<i>Interactiva</i>
4	<i>Completada</i>



Obtención de la respuesta

- Podemos leer la respuesta de las siguientes propiedades:
 - `peticion.responseText`
Respuesta como una cadena de texto
 - `peticion.responseXML`
Respuesta como un objeto `XMLDocument`, que podrá ser analizado mediante el DOM
- Utilizaremos:
 - `responseText` cuando queramos incluir el contenido recibido directamente en el documento
 - `responseXML` cuando estemos intercambiando datos estructurados



Análisis del documento XML

- Analizaremos el documento XML recibido:

```
//mostrar mensajes
var areaMensajes = document.getElementById("mensajesChat");
var textoHTML = "";
docXml = peticion.responseXML;
var raiz = docXml.getElementsByTagName('mensajes');
mensajes = raiz[0].getElementsByTagName('mensaje');
for(i=0;i<mensajes.length;i++) {
    var nick=mensajes[i].getElementsByTagName('nick').item(0).firstChild.data;
    var texto=mensajes[i].getElementsByTagName('texto').item(0).firstChild.data;
    textoHTML += "<strong>&lt;" + nick + "&gt;</strong> " + texto + "<br/>";
}
areaMensajes.innerHTML = textoHTML;
}
```

- El documento sería de la siguiente forma:

```
<mensajes>
  <mensaje>
    <nick>Ana</nick>
    <texto>Hola</texto>
  </mensaje>
</mensajes>
```