Presentación

Nombre: Andy Manuel

Apellidos: Reyes Del Rosario

Matricula: 100488534

Carrera: Licenciatura en Informática

Asignatura: INF5170

Sección: 01

Maestro: Radhames Silverio González

AJAX

AJAX (JavaScript Asíncrono y XML) es un término nuevo para describir dos capacidades de los navegadores que han estado presentes por años, pero que habían sido ignoradas por muchos desarrolladores Web, hasta hace poco que surgieron aplicaciones como Gmail, Google suggest y Google Maps.

Las dos capacidades en cuestión son:

- La posibilidad de hacer peticiones al servidor sin tener que volver a cargar la página.
- La posibilidad de analizar y trabajar con documentos XML.

Para realizar una petición HTTP usando JavaScript, es necesario crear una instancia de una clase que provea esta funcionalidad. Esta clase fue inicialmente introducida en Internet Explorer como un objeto ActiveX, llamado `XMLHTTP`. Después Mozilla, Safari y otros navegadores lo siguieron, implementando una clase `XMLHttpRequest` que soportaba los métodos y las propiedades del objeto ActiveX original.

Como resultado, para crear una instancia de la clase requerida que funcione en todos los navegadores, es necesario poner:

```
if (window.XMLHttpRequest) { // Mozilla, Safari, ...
    http_request = new XMLHttpRequest();
} else if (window.ActiveXObject) { // IE
    http_request = new ActiveXObject("Microsoft.XMLHTTP");
}
```

(El código mostrado es una versión simplificada con fines ilustrativos. Para un ejemplo más realista ver el paso 3 de este artículo.)

Algunas versiones de los navegadores Mozilla no funcionan correctamente si la respuesta del servidor no tiene la cabecera mime de tipo XML. En ese caso es posible usar un método extra que sobreescriba la cabecera enviada por el servidor, en caso que no sea `text/xml`.

```
http_request = new XMLHttpRequest();
http_request.overrideMimeType('text/xml');
```

El próximo paso es decidir qué se hará después de recibir la respuesta del servidor a la petición enviada. A estas alturas sólo es necesario decirle al objeto HTTPrequest qué función de JavaScript se encargará de procesar la respuesta. Para esto se asigna la propiedad onreadystatechange del objeto al nombre de la función de JavaScript que se va a utilizar:

```
http_request.onreadystatechange = nameOfTheFunction;
```

Es importante notar que no hay paréntesis después del nombre de la función y no se pasa ningún parámetro. También es posible definir la función en ese momento, y poner en seguida las acciones que procesarán la respuesta:

```
http_request.onreadystatechange = function(){
    // procesar la respuesta
};
```

Después de especificar qué pasará al recibir la respuesta es necesario hacer la petición. Para esto se utilizan los métodos open() y send() de la clase HTTP request, como se muestra a continuación:

```
http_request.open('GET', 'http://www.example.org/algun.archivo', true);
http_request.send();
```

- El primer parámetro de la llamada a open() es el método HTTP request – GET, POST, HEAD o cualquier otro método que se quiera usar y sea aceptado por el servidor. El nombre del método se escribe en mayúsculas, sino algunos navegadores (como Firefox) podrían no procesar la petición. Para más información sobre los métodos HTTP request visitar W3C specs
- El segundo parámetro es el URL de la página que se esta pidiendo. Por medida de seguridad no es posible llamar páginas en dominios de terceras personas. Se debe saber el dominio exacto de todas las páginas o se obtendrá un error de 'permiso denegado' al llamar open(). Una falla común es acceder al sitio por domain.tld e intentar llamar las páginas como www.domain.tld.
- El tercer parámetro establece si la petición es asíncrona. Si se define TRUE, la ejecución de la función de JavaScript continuará aún cuando la respuesta del servidor no haya llegado. Por esta capacidad es la A en AJAX.

El parámetro en el método send() puede ser cualquier información que se quiera enviar al servidor si se usa POST para la petición. La información se debe enviar en forma de cadena, por ejemplo:

```
name=value&anothername=othervalue&so=on
```

Si se quiere enviar información de esta forma, es necesario cambiar el tipo MIME de la petición usando la siguiente línea:

```
http_request.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
```

Al enviar la petición HTTP es necesario indicar el nombre de la función JavaScript que procesará la respuesta.

```
http_request.onreadystatechange = nameOfTheFunction;
```

A continuación, se verá lo que esta función realiza. En primer lugar, necesita revisar el estado de la petición. Si el estado tiene el valor 4, significa que la respuesta completa del servidor ha sido recibida y es posible continuar procesándola.

```
if (http_request.readyState == 4) {
    // todo va bien, respuesta recibida
} else {
    // aun no esta listo
}
```

La lista completa de valores para la propiedad `readyState` es:

- 0 (no inicializada)
- 1 (leyendo)
- 2 (leido)
- 3 (interactiva)
- 4 (completo)