1 Xestionando peticións GET e POST

O comando **load()** serve tanto para facer peticións GET ou POST dependendo de como configuremos a petición.

¿Cando deberemos usar un método ou outro?

Pois por temas de seguridade e versatilidade ó enviar datos ó servidor usaremos o método POST.

Nalgúns casos por facilidade de programación e sempre e cando non hai ningún conflicto na seguridade poderemos empregar o método GET. O método GET tamén ten a limitación da cantidade máxima de datos que se pode enviar (estaría limitada pola cantidade máxima de texto que poderemos escribir na URL).

1.1 Sumario

1.2  Sobre o método GET

Un dato importante é que moitos navegadores cando fan a caché de páxinas, si estamos empregando o método GET, pode darse o caso de que en vez de baixar de novo a páxina do servidor, amósanos a páxina da caché, dando lugar a confusións moitas veces imcomprensibles.

1.2.1 Obtendo datos con jQuery

```
jQuery.get( url, [data], [callback], [type] )
```

Cando queremos obter datos dende o servidor e decidir o que queremos facer con eles (en vez de deixar que o comando load() o asigne como contido dun elemento HTML), podemos empregar o comando **$.get()**:

A sintaxe é a seguinte:

$.get(url,parametros,retorno)

Como parámetros introduciremos os datos que se enviarán á url. Exemplos de uso:

```
// Chama á páxina test.php sin pasar ningún parámetro adicional.
   $.get("test.php");

   // Chama á páxina test.php pasando como datos o nome e a hora.
   $.get("test.php", { nome: "Carlos", hora: "4pm" } );

// No seguinte exemplo pasamos un array de datos ó servidor, e ignoramos a resposta
// devolta polo servidor.
   $.get("test.php", { 'choices[]': ["Jon", "Susan"]} );

// Chamamos á páxina test.php, e ó rematar a execución amosará unha alerta cos datos devoltos
// pola páxina test.php e que se atopan na variable data.
   $.get("test.php", function(data){
        alert("Datos devoltos: " + data);
```

```
   });

// Chama á páxina test.cgi pasando o nome e hora como párametros polo método GET, e xestiona a
// resposta mediante a función e amosa unha alerta cos resultados devoltos.
   $.get("test.cgi", { nome: "Carlos", hora: "8pm" },
         function(data){
                alert("Datos devoltos: " + data);
         });
```

Exemplo completo:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Ejemplo: $.get()</title>
</head>
<body>
<button type="button" id="testButton">Pulsame!</button>

 <script src="http://code.jquery.com/jquery-2.1.3.min.js"></script>
 <script>
  $(function(){
    $('#testButton').click(function(){
      $.get(
        'reflectData.php',
        {a:1, b:2, c:3},
        function(data) { alert(data); }
        );
    });
  });
</script>
</body>
</html>
```

1.2.2 Obtendo datos JSON

1.3  ¿Qué é JSON?

JSON, é un acrónimo de **JavaScript Object Notation**, é un formato lixeiro para o intercambio de datos. **JSON** é un subconxunto da notación literal de obxectos de JavaScript que non precisa do uso de XML.

[Máis información sobre JSON]

Cando a páxina á que lle enviamos datos nos devolve o resultado en formato XML, pode ser unha tarefa complicada o xestionar eses datos. O XML ten como vantaxes a flexibilidade e é un formato axeitado cando traballamos con estructuras xerárquicas de datos.

Cando o XML non nos aporta vantaxes no seu uso, temos á nosa disposición un novo formato que pode sustituilo: o JSON.

O JSON tamén conta coa vantaxe de que representa mellor a estructura dos datos e require menos codificación e procesamento.

Características máis importantes de JSON:

1. Representación de información xerarquizada dende javascript.
2. É a alternativa máis extendida ó XML, quizais porque é moito máis simple (ainda que non moi humano), e sobre todo é máis rápido de parsear.
3. Existen montón de servizos ofertados en JSON (en lugar de con XML): Google (kml), Yahoo(pipes), Flickr...
4. É a forma máis rápida de servir a jQuery datos estructurados nunha soa petición.

Exemplo: {

```
  clave : 'hola',
  clave2 : ['mundo','luna','marte']
```

}

Exemplo de uso de JSON:

```
// Partindo deste JSON como resposta do noso servidor:
{
    clave : 'hola',
    clave2 : ['mundo','luna','marte']
}

$.getJSON( data.php ,{id:$(this).attr( id ),function(j) {
if (j.error) {
    alert( Error [ +j.error+ ] ;
    return;
}

$( #input1 ).val(j.clave);

for(var idx in j.clave2) {
    $( #textarea1 ).append(j.clave2[idx]);
}
});
```

Outro exemplo de representación de datos en formato JSON:

```
{
"firstName": "John",
"lastName": "Smith",
"address": {"streetAddress": "21 2nd Street", "city": "New York", "state": "NY", "Code": 10021 },
"phoneNumbers": ["212 555-1234","646 555-4567"]
 }
```

Con jQuery dispomos dispomos da función **$.getJSON(url, parametros, retorno,tipo)**:

O formato dos parámetros é idéntico á función anterior **$.get()** salvo que temos un parámetro adicional **tipo**, que é opcional e indicará o tipo de datos que devolverá a función de retorno: **"xml", "html", "script", "json", "jsonp", ou "text"**.

Outro exemplo de uso:

```
$.getJSON("test.php", function(datos){
  alert("Datos obtidos en formato JSON: " + datos.address[3].name);
});
```

1.4  MOI INTERESANTE

Unha das restriccións de seguridade máis importantes no uso de AJAX, é que soamente permite facer peticións a páxinas localizadas no mesmo dominio. Pois ben con este método $.getJSON , si na función de retorno especificamos unha funcion [JSONP] de retorno, poderemos obter datos JSON dun dominio externo ó noso.

1.4.1 Facendo peticións con POST

Se queremos face-la chamada enviando os datos polo método POST empregaremos esta función:

$.post(url, parametros, retorno, tipo)

O formato dos parámetros é idéntico á función anterior **$.get()**.

IMPORTANTE:

O parámetro opcional **tipo**, indicará o tipo de datos que devolverá a función de retorno: **"xml", "html", "script", "json", "jsonp", ou "text"**. Ésto é necesario si dende o servidor devólvese un objeto JSON empregando a cabeceira: application/json; charset=utf-8 Deste xeito non teremos que facer o

$.parseJSON() ou JSON.parse() cando recibamos ese obxecto JSON.

Exemplos:

```
$.postJSON = function(url, data, callback) {
$.post(url, data, callback, "json");
};
```

```
$.post("test.php", { name: "John", time: "2pm" },
  function(data){
    process(data);
  }, "xml");
```

```
// Obtén os datos devoltos pola páxina test.php en formato JSON.
// En PHP para devolve-los datos en formato JSON:
// <?php echo json_encode(array("name"=>"John","time"=>"2pm")); ?>
// A función de PHP json_encode está dispoñible a partir da versión 5.2

$.post("test.php", { func: "obterNomeHora" },
  function(data){
    alert(data.name); // John
    alert(data.time); //  2pm
  }, "json");
```

--Veiga (discusión) 14:16 5 mar 2015 (CET)