Introducción al Desarrollo de Sistemas

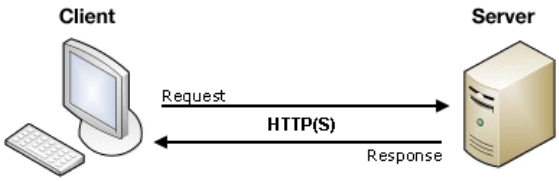
Envío de formularios

¿A dónde van los datos?

Aquí vamos a discutir lo que ocurre con los datos cuando se envía un formulario.

Sobre la arquitectura cliente / servidor

La web se basa en una arquitectura cliente / servidor muy básica que se puede resumir de la siguiente manera: un cliente (normalmente un navegador Web) envía una petición a un servidor (la mayoría de las veces un servidor web como Apache , Nginx , IIS , Tomcat , etc.), utilizando el protocolo HTTP . El servidor responde a la solicitud utilizando el mismo protocolo.



En el lado del cliente, un formulario HTML no es más que una manera fácil de usar conveniente para configurar una petición HTTP para enviar datos a un servidor. Esto permite al usuario para proporcionar información a ser entregada en la petición HTTP.

El elemento `<form>` define cómo se enviarán los datos. Todos sus atributos están diseñados para que pueda configurar la solicitud que se enviará cuando un usuario pulsa un botón de envío. Los dos atributos más importantes son `action` y `method`.

El atributo `action`.

Este atributo define el lugar donde los datos se envian. Su valor debe ser una dirección URL válida.

Si no se proporciona este atributo, los datos serán enviados a la dirección URL de la página que contiene el formulario.

En este ejemplo, los datos se envían a una dirección URL absoluta - `http://foo.com`

```
<form action="http://foo.com">
```

Aquí, nosotros usamos una URL relativa - los datos se envían a una dirección URL diferente en el servidor:

```
<form action= "/somewhere_else">
```

Nota: Es posible especificar una dirección URL que utiliza el protocolo HTTPS (HTTP seguro). Al hacer esto, los datos se cifran junto con el resto de la solicitud, incluso si el propio formulario está alojado en una página insegura se accede a través de HTTPS. Por otro lado, si el formulario está alojado en una página segura, pero se especifica una dirección URL HTTP insegura con el atributo `action` , todos los navegadores mostrarán una advertencia de seguridad para el usuario cada vez que intenten enviar datos, ya que estos no se

pueden cifrar.

El atributo `method`

Este atributo define cómo se envían los datos. El protocolo HTTP proporciona varias formas de realizar una solicitud; Los datos del formulario HTML se pueden transmitir a través de un número de diferentes queridos, los más comunes de los cuales son el método `GET` y el método `POST`.

Para entender la diferencia entre estos dos métodos, vamos a dar un paso atrás y examinar cómo funciona HTTP. Cada vez que desee llegar a un recurso en la Web, el navegador envía una petición a una URL. Una petición HTTP consta de dos partes: un encabezado que contiene un conjunto de metadatos mundial sobre las capacidades del navegador, y un cuerpo que puede contener la información necesaria paraque el servidor pueda procesar la petición específica.

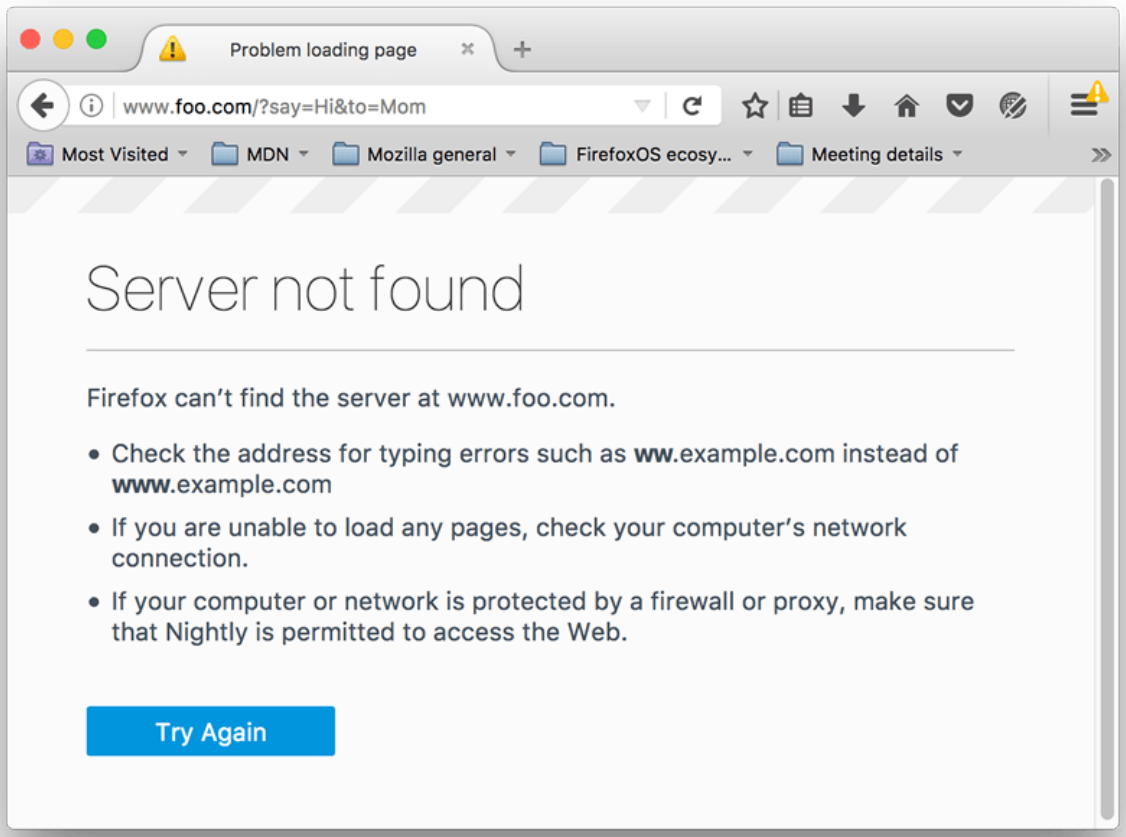
El método GET

El método `GET` es utilizado por el navegador para pedir al servidor que se envíe de vuelta un recurso dado: "Hey servidor, quiero conseguir este recurso." En este caso, el navegador envía un cuerpo vacío. Debido a que el cuerpo está vacío, si un formulario se envía utilizando este método, los datos enviados al servidor se anexan a la URL.

Considere la siguiente forma:

```
<form action="http://foo.com" method="get">
  <div>
    <label for="decir"> ¿Qué saludo quiere decir? </label>
    <input name="decir" id="decir" value="Hola">
  </div>
  <div>
    <label for="para"> ¿A quién se lo quiere decir? </label>
    <input name="para" value="mamá">
  </div>
  <div>
    <button> Botón enviar mis saludos </ button>
  </div>
</form>
```

Dado que el método `GET` ha conseguido el recurso, verá en la URL lo siguiente en la barra de direcciones del navegador `www.foo.com/?say=Hi&to=Mom` cuando se envía el formulario.



Los datos se añaden a la URL como una serie de pares de nombre / valor. Después que la dirección web URL ha terminado, se incluye un signo de interrogación (`?`) seguido de los pares de nombre / valor, cada uno separado por un signo (`&`). En este caso estamos pasando dos piezas de datos en el servidor:

- `say`, Que tiene un valor de `Hi`
- `to`, Que tiene un valor de `Mom`

La solicitud HTTP se ve así:

```
GET /? = Decir Hola & a = mamá HTTP / 1.1
Anfitrión: foo.com
```

El método POST

El `POST` método es un poco diferente. Es el método que el navegador utiliza para comunicarse con el servidor cuando se pide una respuesta que tenga en cuenta los datos proporcionados en el cuerpo de la petición HTTP: "Hey servidor, echa un vistazo a estos datos y envíame de vuelta un resultado apropiado." Si un formulario se envía utilizando este método, los datos se anexan al cuerpo de la petición HTTP.

Veamos un ejemplo - se trata de algo similar a como se vió en el método `GET`del apartado anterior, pero con el `method` establecido `post`.

```
<Form action = "http://foo.com" method = "post">
  <Div>
    <Label for = "dice"> Lo saludo qué quiere decir? </ Label>
    <Input name = "decir" id = "decir" value = "Hola">
  </ Div>
  <Div>
    <Label for = "para"> ¿Quién usted quiere decir que a? </ Label>
    <Input name = "a" value = "mamá">
  </ Div>
  <Div>
    <> Botón enviar mis saludos </ botón>
  </ Div>
</ Form>
```

Cuando el formulario se envía mediante el método `POST`, no se obtienen los datos adjuntos en la dirección URL, y la solicitud HTTP se parece a esto y los datos son incluidos en el cuerpo de la petición en su lugar:

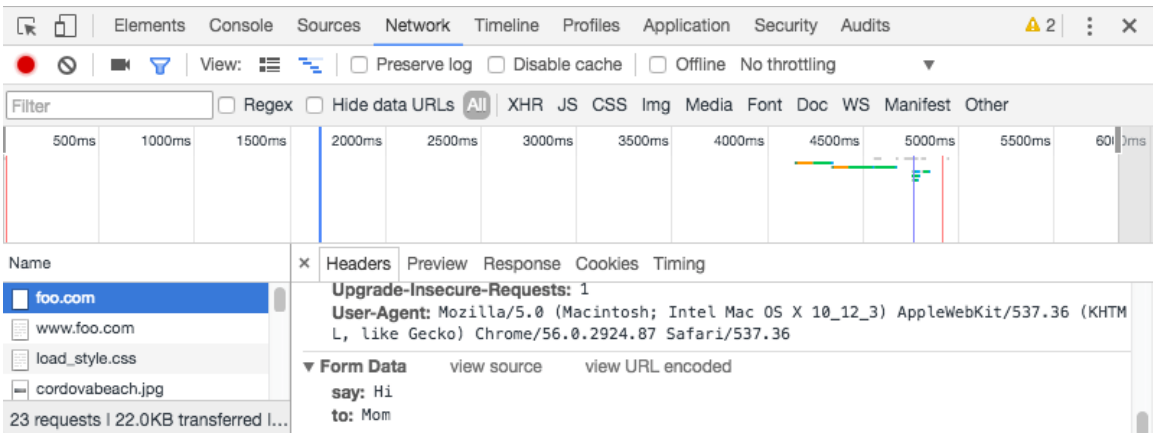
```
POST / HTTP/1.1
Anfitrión: foo.com
Content-Type: application / x-www-form-urlencoded
Content-Length: 13

decir = Hi & a = mamá
```

La cabecera `Content-Length` indica el tamaño del cuerpo, y la cabecera `Content-Type` indica el tipo de recurso que se envía al servidor. Discutiremos estas cabeceras más adelante.

Visualización de peticiones HTTP

Las peticiones HTTP nunca se muestran al usuario (si quieres verlos, es necesario utilizar herramientas como el Monitor de red Firefox o las herramientas de desarrollo de Chrome). A modo de ejemplo, los datos del formulario se muestran a continuación en la pestaña de Chrome red:



Lo único que se muestra al usuario es la dirección URL llamada. Como mencionamos anteriormente, con una petición `GET` del usuario,se verán los datos en su barra de direcciones, pero con una petición `POST` no será de esta manera. Esto puede ser muy importante por dos razones:

1. Si necesita enviar una contraseña (o cualquier otra pieza sensible de los datos), nunca utilice el método `GET` o se arriesga a mostrar en la barra de direcciones, lo que sería muy inseguro.
2. Si necesita enviar una gran cantidad de datos, el método `POSt` es preferible debido a que algunos navegadores limitan los tamaños de las direcciones URL. Además, muchos servidores limitan la longitud de las URL que aceptan.

En el lado Servidor: la recuperación de los datos

Sea cual sea el método HTTP que elija, el servidor recibe una cadena que será analizada con el fin de obtener los datos como una lista de pares clave / valor. La forma de acceder a esta lista depende de la plataforma de desarrollo que use y de las estructuras específicas que pueda estar usando con él. La tecnología se utiliza también determina cómo se manejan claves duplicadas; A menudo,se da prioridad al valor recibido más recientemente para una clave dada .

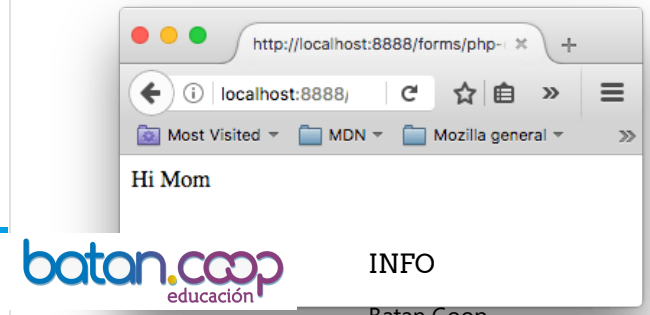
Ejemplo: PHP Raw

PHP ofrece algunos objetos globales para acceder a los datos. Suponiendo que usted ha utilizado el método `POST`, el siguiente ejemplo sólo toma los datos y lo muestra al usuario. Por supuesto, lo que se hace con los datos depende de usted. Es posible visualizarlos, almacenarlos en una base de datos, enviarlos por correo electrónico, o procesarlos de alguna otra manera.

```
<? Php
  // La variable global $ _POST que permite acceder a los datos enviados con el método POST por su nombre
  // Para acceder a los datos enviados con el método GET, puede usar $ _GET
  $ = Decir htmlspecialchars ($ _POST [ 'decir']);
  $ A = htmlspecialchars ($ _POST [ 'a']);

  echo $ digamos, '', $ a;
?>
```

Este ejemplo muestra una página con los datos que enviamos. Esto se puede ver en acción en nuestro archivo ejemplo php-example.html - que contiene un ejemplo similar en forma como el que hemos visto antes, con un `method` con parámetro `post` y un `action` con parámetro `php-example.php` Cuando se envía, envía los datos del formulario al script php-ejemplo.php , que contiene el código de PHP que se ha visto en el bloque anterior. Cuando se ejecuta este código, la salida en el navegador es `Hi Mom`.



Nota : Este ejemplo no funcionará cuando se carga en un navegador localmente - los navegadores no pueden interpretar código PHP, por lo que cuando se envía el formulario en el navegador sólo se puede ofrecer la descarga del archivo PHP para usted. Para conseguir que funcione, es necesario ejecutar el ejemplo a través de un servidor PHP de algún tipo. (Ver parte del curso dedicada a la instalación del servidor Apache).

Fuente: https://developer.mozilla.org/es/docs/Learn/HTML/Forms/Sending_and_retrieving_form_data

Última modificación: miércoles, 15 de julio de 2020, 18:00