Записки инженера

Доступным языком заметки по IT технологиям

- Главная
- Услуги
- ХТКЭМ

Отправить xml / json / jsonp методом POST / GET с помощью PHP / JavaScript, обработка и прием

В данном посте я собрал готовые простые решения, необходимые каждому web разработчику, сталкивающегося **с отправкой (POST/GET) и обработкой полученных данных (xml, json, …)**.

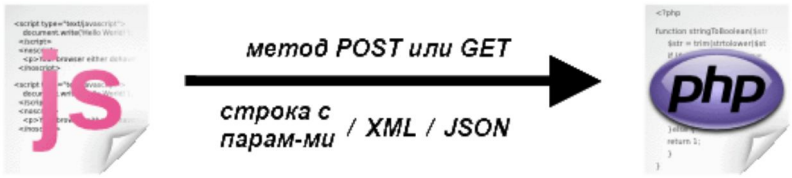
Здесь для себя вы найдете:

1. **как сформировать XML, JSON и отправить** данные с помощью **JavaScript (JQuery)** на сервер (локальный)
2. как **обработать полученные данные** на стороне сервера, т.е. **парсинг xml, json** с помощью **PHP**
3. как **обработать полученные ответ** с сервера (**парсинг XML, JSON** и как запустить полученный **javascript**) с помощью **JavaScript (JQuery)**
4. как **отправить данные** с помощью **JavaScript (JQuery)** на **удаленный сервер (JSONP)**
5. как **передать данные обратно браузеру** с удаленного сервера **(JSONP)** с помощью **PHP**
6. **как сформировать XML, JSON и отправить** данные с помощью **PHP** на другой сервер **(серверный скрипт)**

Что-бы было проще изложение готовых решений, рассмотрим **6-е крайних ситуации**, с которыми мы многие из нас встречаемся в работе:

Ситуация №1

“нужно отправить данные с браузера на локальный сервер” (т.е. сервер с которого загружается сайт)



Решение:

Для начала, я хочу сказать, что все что касается **JavaScript** мы будем делать через фреймворк **JQuery**, поэтому в html коде **необходимо подключить его**

```
<script type="text/javascript" src="jquery-1.7.1.min.js"></script>
```

Скачать файл jQuery сможете на официальном сайте

Для отправки **данных локальному серверу** в html коде страницы, должен быть следующий код JavaScript код

```
<script type='text/javascript'>
$.ajax({
   type: "POST",                                  //метод запроса, POST или GET (если опустить, то по умолчанию GET)
   url: "script.php",                             //серверный скрипт принимающий запрос
   data: "request=message&request2=message2",     //можно передать строку с параметрами запроса, ключ=значение
   //data: {request:"message",request2:"message2"},  //можно передать js объект, ключ:значение
   //data: {request:["message #A", "message #B"],request2:"message2"},  //можно передать массив в одном из параметре запроса
   success: function(res) {                       //функция выполняется при удачном завершение
     alert("Данные успешно отправлены на сервер");
   }
});
</script>
```

Если нам нужно отправить на локальный сервер данные в формате **XML** или **JSON**, то мы их можем передать **в одном из параметре** POST/GET запроса.

Например, следующий код **сформирует XML** и отправит его на локальный сервер, т.е. **script.php.**

```
<script type='text/javascript'>
//сформируем xml
var xmlData = '<?xml version="1.0" encoding="UTF-8"?>';
     xmlData+='<note>';
     xmlData+='<to sex="male">Petr</to>';
     xmlData+='<from>Marina</from>';
     xmlData+='<heading>Reminder</heading>';
     xmlData+='<body>Take out the trash</body>';
     xmlData+='</note>';
//отправим xml
$.ajax({
   type: "POST",                 //метод запроса, можно POST можно GET (если опустить, то по умолчанию GET)
   url: "script.php",            //серверный скрипт принимающий запрос
   data: {request:xmlData,request2:"message2"}, //можно передать переменную с xml в одном из параметре запроса
   success: function(res) {      //функция выполняется при удачном завершение
     alert("Данные успешно отправлены на сервер");
   }
});
</script>
```

Второй пример, **сформируем JSON** и также отправим его на локальный сервер, т.е. скрипту **script.php,** но для начала нужно подключить плагин **jquery-json**

```
<script type="text/javascript" src="jquery.json-2.4.min.js"></script>
```

Скачать его можно здесь

Результирующий код

```
<script type='text/javascript'>
//сформируем JSON
var JsonData = {
  "test1":"value1",
  "test2":{
        "test2_in":"internal value test2"
  }
};
//отправим JSON
$.ajax({
   type: "POST",                 //метод запроса, POST или GET (если опустить, то по умолчанию GET)
   url: "script.php",            //серверный скрипт принимающий запрос
   data: {request:$.toJSON(JsonData),request2:"message2"},  //можно передать переменную с json в одном из параметре запроса
   success: function(res) {      //функция выполняется при удачном завершение
     alert("Данные успешно отправлены на сервер");
   }
});
</script>
```

Проверить, что мы, все таки отправляем можно с помощью следующего скрипта, сохраните его под именем “script.php”. Данный скрипт при получение POST/GET запроса сохраняет переданные данные в data.txt.

Ситуация №2

“нужно отправить данные с браузера на удаленный сервер” (на другой домен)

Решение:

Сначала я продемонстрирую решение **в упрощенном виде** (используем **$.getJSON**)

```
<script type='text/javascript'>
//формируем JSON
var JsonData = {
  "test1":"value1",
  "test2":{
            "test2_in":"internal value test2"
  }
};
//отправляем JSON, используем технологию JSONP для кроссдоменной передачи данных
$.getJSON("script.php?callback=?",    //вместо script.php можете указать url к скрипту
                                      // знак '?' указывает на то, что имя callback функции
                                      // генерируется jquert
    {request:$.toJSON(JsonData)}, //отсылаем JSON
    function(data) {
        console.log(data.request); //в консоле браузера выводим json в параметре request
                                   //т.е. то что нам отправил сервер в ответ
       //console.log(data.request.name1);  //мы можем вывести какой-то параметр полученного json, например name1
        });
</script>
```

Ответ сервера выводится в консоль (параметр request).

Теперь тоже самое но через **$.ajax**

```
<script type='text/javascript'>
var JsonData = {
  "test1":"value1",
  "test2":{
              "test2_in":"internal value test2"
   }
};

$.ajax({
url: 'script.php?callback=?', //вместо script.php можете указать url к скрипту
                              // знак '?' указывает на то, что имя callback функции
                              // генерируется jquert
data: {request:$.toJSON(JsonData)}, //отсылаем JSON
type: 'GET',             // если POST, то преобразовано все равно будет в GET
dataType: 'jsonp',       //ожидаем от сервера данные типа jsonp
crossDomain: true,
success: function(data) {
    console.log(data.request); //в консоле браузера выводим json в параметре request
                               //т.е. то что нам отправил сервер в ответ
   //console.log(data.request.name1); //мы можем вывести какой-то параметр полученного json
}
});
</script>
```

Какие преимущества через **$.ajax?** Да пожалуй только то, что можно поставить обработчик на событие **error.**

Ситуация №3

“серверу (php) по запросу нужно отправить данные обратно браузеру используя технологию JSONP”

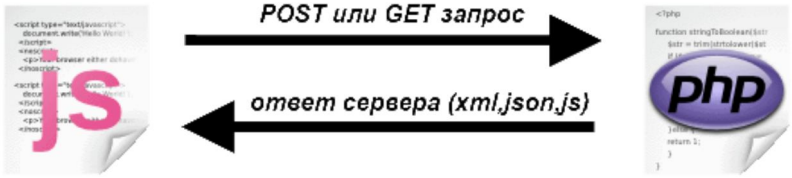
Для этого на стороне сервера должен быть следующий **php скрипт**

```
<?php
  //сформируем json, который вернем браузеру обратно
  $json_str.='{';
  $json_str.='"name1":"value1",';
```

```
  $json_str.='}';
  $callback = $_GET['callback'];  //узнаем имя функции callback
  echo $callback.'('."{'request':$json_str}".')'; //возвращаем ответ $json_str
?>
```

Ситуация №4

*“по запросу к серверу нужно получить ответ (**xml, json, script**) и обработать его”*



Решение:

1. Отправим **POST/GET запрос на сервер (локальный)** и обработаем полученный ответ - **xml**

Допустим сервер по запросу должен вернуть **следующий xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to sex="male">Petr</to>
  <from>Marina</from>
  <heading>Reminder</heading>
  <body>Take out the trash</body>
</note>
```

Тогда наш js код:

```
<script type='text/javascript'>
$.ajax({
  type: "POST", //метод запроса, можно POST можно GET (если опустить, то по умолчанию GET)
  url: "script.php",
  data: {request:"message",request2:"message2"},     //отправим данные, если нужно
  success: function(res) {                          //функция выполняется при удачном завершение
    var return_value=$.parseXML(res);               //конвертируем xml строку в xml документ
    console.log($(return_value).find('to').text());  //выведем в консоль значение тега to
    //console.log($(return_value).find('to').attr('sex'));  //выведем в консоль значение атрибута 'sex'
  }
});
</script>
```

2. Отправим **POST/GET запрос на сервер (локальный)** и обработаем полученный ответ - **json**

Допустим сервер по запросу должен вернуть **следующий json**

```
{
   "test1":"value1",
   "test2":{
      "test2_in":"internal value test2"
    }
};
```

Тогда наш js код:

```
<script type='text/javascript'>
$.ajax({
  type: "POST", //метод запроса, можно POST можно GET (если опустить, то по умолчанию GET)
  url: "script.php",
  data: {request:"message",request2:"message2"}, //отправим данные, если нужно
  success: function(data) {                      //функция выполняется при удачном завершение
    console.log($.parseJSON(data).test1);        //выведем в консоль содержимое test1
    //console.log($.parseJSON(data).test2.test2_in); //выведем в консоль содержимое test2_in
  }
});
</script>
```

P.S. Отправлять **POST/GET** запросы и обрабатывать ответы помимо выше указанных вариантов можно с помощью функций **$.post()** и **$.get()**, это сокращает код. В в качестве аргументов данные функции принимают: “url адрес отправки”, данные {ключ:значение} и

функция обработчик. Например также как и выше отправим два параметра **{request:"message",request2:"message2"}** и выведем значение параметра **test1** полученного **json**

```
<script type='text/javascript'>
$.post("script.php",{request:"message",request2:"message2"},
    function(data){
      console.log($.parseJSON(data).test1); //выведем в консоль содержимое test1
    }
);
</script>
```

Для **GET** запроса все аналогично, только вместо **$.post(…)** используем **$.get(…)** .

3. Отправим **GET запрос на сервер (на удаленный или локальный не важно)** и выполним полученный **javascript**

Допустим сервер по **GET** запросу возвращает следующий js код:

```
alert('Полученный скрипт');
```

Тогда **наш js** код должен выглядеть так, сначала упрощенный вариант:

```
<script type='text/javascript'>
$.getScript("script.php"); //загружаем и выполняем js код
                           //в данном варианте нельзя отправить данные на сервер
</script>
```

Тоже самое но, через **$.ajax**

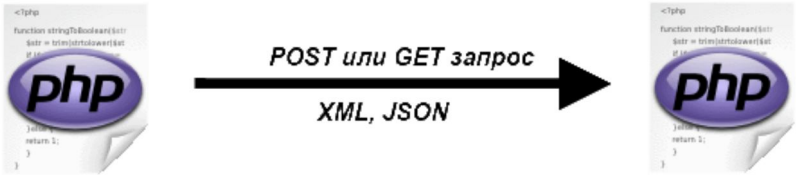
```
<script type='text/javascript'>
$.ajax({
  type: "GET", //разрешается только GET
  url: "script.php",
  data: {request:"message",request2:"message2"}, //посылаем данные, если нужно
  dataType: "script"      // тип возвращаемых данных
})
</script>
```

Преимущество использования **$.ajax** в данном случае перед **$.getScript,** в возможности отправки данных на сервер. С помощью **$.ajax** можно, с помощью **$.getScript** нельзя.

При выполнение **любого из вариантов** указанных выше, будет **исполнен загруженный с сервера js** скрипт, т.е. будет выведено окно “Полученный скрипт”.

Ситуация №5

*“нужно отправить POST/GET запрос с данными (**xml, json**) с помощью PHP на другой сервер (локальный или удаленный не важно)”*



Для отправки HTTP запроса любому другому серверу мы будем использовать **PHP** функцию **fsockopen()**.

Первый делом я опишу как сформировать **XML** и **JSON,** а потом предоставлю **PHP** скрипт который посылает **HTTP** запрос любому другому серверу.

1. Сформируем **XML**

Сформировать **XML** можно “ручным” способом, т.е. создать строку и наполнить ее **XML** содержимым

```
<?php
$xml_str = '<?xml version="1.0" encoding="UTF-8"?>';
$xml_str.='<note>';
$xml_str.='<to sex="male">Petr</to>';
$xml_str.='<from>Marina</from>';
$xml_str.='<heading>Reminder</heading>';
$xml_str.='<body>Take out the trash</body>';
$xml_str.='</note>';

/* далее код для отправки данных */
```

```
-
```

или более правильным, но медленным способом используя **PHP** класс **domDocument**

```
<?php
$xml = new domDocument("1.0", "utf-8");
$note = $xml->createElement("note");    //создаем элемент note
$xml->appendChild($note);               //добавляем xml дочерний элемент note
$to = $xml->createElement("to","Petr"); //создаем элемент note
$to->setAttribute("sex","male");        //добавляем атрибут sex с значением male
$note->appendChild($to);                //добавляем note дочерний элемент to
$from = $xml->createElement("from","Marina");
$note->appendChild($from);
$heading = $xml->createElement("heading","Reminder");
$note->appendChild($heading);
$body = $xml->createElement("body","Take out the trash");
$note->appendChild($body);
$xml_str=$xml->saveXML(); //сохраняем xml в переменную $xml_str
?>
```

Оба способа в данном примере формируют **идентичный XML,** вы можете выбрать любой.

2. Сформируем **JSON**

Создадим **JSON** ”ручным” способом, т.е. создадим строку и наполнить ее **JSON** содержимым

```
<?php
$json_str.='{';
$json_str.='"test1":"value1",';
$json_str.='"test2":{';
$json_str.='"test2_in":"internal value test2"';
$json_str.='}';
$json_str.='}';
?>
```

Другой пример, создадим **JSON** с помощью функции **json_encode()**

```
<?php
$request_data = array('test1' => 'value1', 'test2' => array('test2_in' => 'internal value test2'));
$json_str = json_encode($request_data);
?>
```

3. Отправим наш **XML** или **JSON** на какой либо сервер

В ниже расположенном скрипте, **нужно только** выбрать метод **POST** или **GET**, **ip адрес** или **доменное имя сервера** куда шлем, **имя принимающего скрипта** на сервере и значение параметра **request =>**, т.е. какие данные слать

```
<?php
$method="GET";             // "POST" передача данных методом POST, "GET" методом GET
$serv_addr = '127.0.0.1';  // ip адрес или доменное имя сервера, куда шлем данные
$serv_port = 80;           // номер порта
$serv_page = 'script.php'; // серверный скрипт принимаюший запрос
$timelimit = 2;            // время ожидания ответа в сек., по умолчанию - 30 сек.

/* передаваемые данные в формате: название переменной => значение */
$data = array(
  'request' => $json_str,  //! ЕСЛИ НУЖНО ОТПРАВИТЬ XML, ТО ЗАМЕНЯЕМ
                           //! НА ДРУГУЮ ПЕРЕМЕННУЮ,Т.Е. НА $xml_str
  'request2' => 'message 2'
);
/* генерируем строку с запросом */
$post_data_text = '';
  foreach ($data AS $key => $val)
    $post_data_text .= $key.'='.urlencode($val).'&';

/* убираем последний символ & из строки $post_data_text */
$post_data_text = substr($post_data_text, 0, -1);
/* прописываем заголовки, для передачи на сервер
    последний заголовок должен быть обязательно пустым,
    так как тело запросов отделяется от заголовков
    пустой строкой (символом перевода каретки "\r\n") */

// заголовок для метода POST
$post_headers = array('POST /'.$serv_page.' HTTP/1.1',
                      'Host: '.$serv_addr,
                      'Content-type: application/x-www-form-urlencoded charset=utf-8',
                      'Content-length: '.strlen($post_data_text),
                      'Accept: */*',
                      'Connection: Close',
                      '');

// заголовок для метода GET
$get_headers = array('GET /'.$serv_page.'?'.$post_data_text.' HTTP/1.1',
                      'Host: '.$serv_addr,
                      'Accept: */*',
                      'Connection: Close',
```

```
                        '');

if ($method=="POST") {
  $headers=$post_headers;
}
if ($method=="GET") {
  $headers=$get_headers;
}
/* сложим элементы массива в одну переменную $headers_txt
/* и добавим в конец каждой строки, знак "\r\n" - перевода каретки */
$headers_txt = '';
foreach ($headers AS $val) {
  $headers_txt .= $val.chr(13).chr(10);
}

// при POST запросе в конец заголовка добавляем наши данные
// для GET нет данной необходимости, т.к. данные уже в заголовке
if ($method=="POST") {
  $headers_txt = $headers_txt.$post_data_text.chr(13).chr(10).chr(13).chr(10);
}

// открытие сокета
$sp = fsockopen($serv_addr, $serv_port, $errno, $errstr, $timelimit);

// в случае ошибки, вернем ее
if (!$sp)
  exit('Error: '.$errstr.' #'.$errno);

// передача HTTP заголовка
  fwrite($sp, $headers_txt);

// если соединение, открытое fsockopen() не было закрыто сервером
// код while(!feof($sp)) { ... } приведет к зависанию скрипта
// в коде ниже - эта проблема решена
$server_answer = '';
$server_header= '';

$start = microtime(true);
$header_flag = 1;
while(!feof($sp) && (microtime(true) - $start) < $timelimit) {
  if ($header_flag == 1) {
    $content = fgets($sp, 4096);
    if ($content === chr(13).chr(10))
      $header_flag = 0;
    else
      $server_header .= $content;
  }
  else {
      $server_answer .= fread($sp, 4096);
  }
}

 // закрываем дескриптор $sp
 fclose($sp);

 /* для отладки, раскомментируйте строку ниже
    печать передаваемого HTTP запроса */
 //echo $headers_txt;
?>
```

Ситуация №6

“серверу (PHP) нужно принять данные (просто параметры, xml, json) и обработать их”

1. Тут все просто, если мы хотим просто **принять параметры POST/GET запроса**, то это сделает следующий PHP код:

```
<?php
$request_post = $_POST['request']; // request имя параметра при POST запросе
$request_get = $_GET['request'];   // request имя параметра при GET запросе
?>
```

2. Если нам в одном из параметров запроса **передали массив**, то работать мы с ним можем **таким образом:**

```
<?php
$post_request = $_POST['request'][1]; // если поле массив, то можем указать индекс сразу
                                      // или работать с $post_request как с массивом далее
?>
```

3. Если в одном из параметров находится **XML**, то работать с ним будет удобно используя расширения **SimpleXML** (встроено в **php** с версии **5**), а конкретнее, мы будем использовать функцию **simplexml_load_string().**

Пусть на php скрипт приходит **GET** запрос, с параметром **request,** который содержит следующий **XML**

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to sex="male">Petr</to>
  <from>Marina</from>
  <heading>Reminder</heading>
  <body>Take out the trash</body>
</note>
```

Тогда **PHP** скрипт обрабатывающий данный **XML** может выглядеть так:

```
<?php
$post_request = $_GET['request'];
$xml = simplexml_load_string($post_request); //разбор xml
echo $xml->to;          //печатаем содержимое тега to
//echo $xml->to['sex']; //печатаем содержимое атрибута sex тега to
?>
```

При запросе к нему, он вернет значение тега **to** (или значение атрибута если раскоментировать строку ниже).

4. Если в одном из параметров запроса находится **JSON,** то работать с ним будет удобнее используя функцию **json_decode()**.

Пусть на php скрипт приходит **POST** запрос, с параметром **request,** который содержит следующий **JSON**

```
{
   "test1":"value1",
   "test2":{
      "test2_in":"internal value test2"
    }
};
```

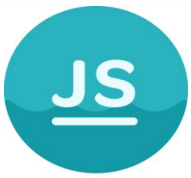
Тогда **PHP** скрипт обрабатывающий данный **JSON** может выглядеть так:

```
<?php
$data = json_decode($_POST['request']);
echo $data->test1;              // параметр test1
//echo $data->test2->test2_in;  // параметр test2_in вложенный в test2
?>
```

При запросе к нему, он вернет значение параметра **test1** (или значение вложенного параметра **test2_in** если раскоментировать строку ниже).

Вам будет интересно:

Отлавливаем POST/GET запрос



Структура и синтаксис JavaScript (выжимка)

Буду признателен если вы поделитесь данным постом