COM222 DESENVOLVIMENTO DE SISTEMAS WEB

Aula 14: Ajax, JSON e jQuery

Conteúdo

- AJAX
- JSON
  - Introdução
  - Estruturas
  - Exemplo de Manipulação
- jQuery, AJAX e JSON
  - Métodos
  - Exemplo 1
  - Exemplo 2
  - Exemplo 3
  - Exercício

Funcionamento típico da Web

- Comportamento típico de uma aplicação web consiste no carregamento de uma página, execução de uma tarefa – preenchimento de um formulário, por exemplo – e, eventualmente envio de dados ao servidor
- Trata-se de um procedimento sequencial , no qual páginas são requisitadas uma por vez, e deve-se esperar o carregamento de cada página para executar uma tarefa
- A necessidade de carregamento de uma nova página para toda comunicação cliente-servidor é uma desvantagem desse modelo

AJAX

- O uso de Javascript começou a quebrar este paradigma
  - Passou a ser possível, por exemplo, validar formulários ANTES de submete-los ao servidor
- Javascript, entretanto, não era capaz de se comunicar diretamente com o servidor até o surgimento do AJAX
- AJAX = **A**synchronous **J**avascript **A**nd **X**ML
  - Permite o uso de Javascript para se comunicar diretamente com o servidor sem submeter formulários ou exigir o carregamento de uma nova página
  - O termo assíncrono significa que o usuário não é obrigado a esperar que o servidor responda à uma requisição para continuar usando a aplicação

JSON

- JSON (JavaScript Object Notation) é um modelo para armazenamento e transmissão de dados no formato texto.
- As aplicações Web vem utilizando o JSON devido a sua simplicidade e capacidade de estruturar informações de forma mais compacta do que o XML.
- Apesar de conter a palavra JavaScript no seu nome, JSON não tem seu uso limitado apenas a essa linguagem (ActionScript, C / C++, C#, Cold Fusion, Delphi, E, Erlang, Java, Lisp, Perl, Objective-C, Objective CAML, PHP, Python, Rebol, Ruby, Scheme, Squeak, etc..)

JSON - Estruturas

- O modelo JSON é baseado nas seguintes estruturas de dados:
  - Objetos: conjunto de containers que contém um par de chave/valor (key/value), incorporados por { } e separados por vírgula. Cada par chave/valor é separado por dois pontos (:). Chaves são Strings e o valores são valores JSON (Strings, Array, Number, Boolean)

```
{
"nome": "Marcos",
"sobrenome": "Silva"
}
```

  - Array [ ]

```
"ordem":[ "primeiro", "segundo", "terceiro" ]
```

JSON - Estruturas

- Number:

```
42
3.1415926
```

- String " ":

```
"primeiro"
```

- Boolean:

```
true
false
```

- null:

```
null
```

JSON - Estruturas

- Exemplo de um objeto JSON contendo uma variedade de estruturas:

```
{
"name" : "Mark Shisk",
"age" : 22,
"student" : true,
"books" : [
    {
        "id" : 01 ,
        "author" : "Herbert Schildt",
        "coauthor": ["Andre", "Jacob"]
    },
    {
        "id" : 07 ,
        "author" : "E.Balagurusamy",
        "coauthor": ["Susan", "Mario"]
     }]
}
```

```
// Acessando dados JSON
// através do JavaScript
var json = { "name": "Mark Shisk",
                    .... }
// Nome
json.name      // "Mark Shisk"
json['name']   // "Mark Shisk"

// Books
json.books     // "[ {}, {} ]"


// Co-autores do primeiro livro
json.book[0]["coauthor"]
json.book[0].coauthor
json["book"][0]["coauthor"]
```

Exemplo de Maniputação

- Abra o seu navegador
- CTRL + SHIFT + i
- Crie a variável ‘ z = {}’
- Manipule a váriavel ‘z’ de modo que ela tenha um atributo do tipo string, number, array, null, object e boolean
- Utilize o acesso aos atributos através de colchetes ‘z[key]’ e através do ponto ‘z.key’

JSON

- Devido a essa facilidade, muitos serviços de WEB vem utilizando o JSON como formato de texto para troca de informações.
- O que trafega entre um servidor e um cliente não é um objeto JSON, mas sim um objeto JSON convertido em String.
- O servidor back-end incorpora os dados a serem enviados para o cliente em uma estrutura JSON e por fim converte (encode) essa estrutura para String e envia ao cliente a resposta.

jQuery, AJAX e JSON

- Para a troca de dados com o servidor, jQuery possui uma coleção de funções simples para este uso:

Method	Operation
`$.ajax()`	Realiza uma requisição AJAX
`$.get()`	Carrega os dados do servidor usando requisição AJAX HTTP GET
`$.getJSON()`	Carrega os dados no modelo JSON usando HTTP GET
`$.post()`	Envia os dados através do método HTTP POST.

- As funções **`$.get()`**, **`$.getJSON()`** e **`$.post()`** são extensões da função **`$.ajax()`**, sendo cada função pré-configurada para um tipo de requisição

jQuery e AJAX

- Usando a função **$.get()** com o exemplo :

```
var myURL = "telltime.php";
var myRand = parseInt(Math.random()*999999999999999);

// A função $.get() irá lançar a requisição:
// telltime.php?rand=609846947836027

$.get( myURL,{ rand: myRand }).done(function(){
    // Quando terminar a requisição
    // Faça a atualização do relógio
});
```

jQuery e AJAX

- Usando a função **$.ajax()**:

```
var myURL = "telltime.php";
var myRand = parseInt(Math.random()*999999999999999);

$.ajax({
  dataType: 'html',    // Define o formato de resposta da requisição
  type: 'GET',          // Define o tipo de requisição
  url: myUrl,           // URL a qual será feito a requisição
  data: { rand: myRand },// Dados a serem enviados
  success: update       // função que será chamada após obter sucesso
  error: displayError   // função que será caso ocorra um erro
});
```

- O parâmetro dataType pode ser preenchido por: (xml, json, script, ou html)

Primeiro Exemplo

- Retornando as horas em formato JSON
  - Crie um arquivo “request.php”

```php
<?php
    $h = date("H"); // get hour
    $m = date("i"); // get minutes
    $s = date("s"); // get seconds

    // Estrutura JSON em PHP
    $data = array('hour'=> $h, 'minute'=> $m, 'second'=> $s);

    // Transformando essa estrutura em String
    $data = json_encode($data);

    echo ($data);
?>
```

Primeiro Exemplo

- Crie um arquivo “index.html”

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title> First WebSite </title>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.2/jquery.min.js">
</script>
      <style>
         body {
            background-color: #CCCCCC;
            text-align: center;
          }
        .displaybox {
            margin: auto;
            width: 150px;
            background-color: #FFFFFF;
            border: 2px solid #000000;
            padding: 10px;
            font: 1.5em normal verdana, helvetica, arial, sans-serif;
          }
        </style>
```

```
        <script>
            $(document).ready(function(){
            var url = "request.php";

            function getAndDisplay(){
               $.get( url ).done(function( data ){
                  console.log("get:", data);
                  console.log("get + parse", $.parseJSON(data));
                });
            }

            function getJSONAndDisplay(){
               $.getJSON( url ).done(function( data ) {
                  console.log("getJSON:",data);
                  setTime(data);
                });
            }

            // Preenche o campo de horas
            function setTime(obj){
               var text = obj.hour + ":" + obj.minute + ":" + obj.second
               $("#showtime").html(text);
            }

            // Atribui ao evento de click do
            // botao "btn_click" à função teste()
            $("#bt_click").click(function(){
               getAndDisplay();
               getJSONAndDisplay();
            });
         });
        </script>

    </head>
    <body>
      <button id="bt_click">Get the Time</button>
      <div id="showtime" class="displaybox"></div>
    </body>
</html>
```

Envio de dados do servidor

- A resposta armazenada pela função `$.get()` pode ser qualquer texto que Javascript seja capaz de processar como uma string (código JavaScrtip, HTML, String, JSON, XML)
- A função `$.getJSON()` já trata a resposta da requisição como uma String que contém o formato JSON

Exemplo 2 – Live search

- Veremos agora um exemplo no qual a busca é feita no processo de digitação do texto
  - A cada nova letra digitada é feita uma requisição de busca e que contém o conteúdo digitado é retornado
  - Similar à busca implementada pelo Google recentemente
- Usaremos PHP no back-end
  - Lista contendo um conjunto de nomes
  - Dados digitados no cliente são enviados ao back end, onde é feita comparação de strings com a lista de nomes
    - Nomes que contém a substring digitada são retornados

HTML


```
<!DOCTYPE html>
<html>
<head>
    <title>jQuery Demonstration</title>
    <script
src="https://ajax.googleapis.com/ajax/libs
/jquery/2.2.2/jquery.min.js"></script>


</head>
<body>
<h1>jQuery Live Search</h1>

<p>
    Search for: <input type="text"
id="searchstring"/>
</p>

<div id="results">
    <ul id="list">
        <li>Results will be displayed
here.</li>
    </ul>
</div>

</body>
</html>
```

```
<script>
   $(document).ready(function(){
      var url = "names.php";

      function getJSONAndList(value){
         $.getJSON(url,{"query" : value}).done(
           function( data ) {
            console.log("getJSON:",data);
            setList(data);
          });
      }


      // Preenche a lista de Nomes
      function setList(obj){
         var result = obj.result;
         var list = $("#list");
         list.empty(); // limpando a lista

         for (var i = 0; i < result.length ; i++) {
            var li = $("<li>"+ result[i] +"</li>");
            list.append(li);
          }
      }

      $("#searchstring").keyup(function(){
         getJSONAndList(this.value)
      })
   });
</script>
```

Back end PHP (names.php)


```
<?php
   $people = array(
   "Abraham Lincoln", "Martin Luther King", "Jimi Hendrix",
   "John Wayne", "John Carpenter", "Elizabeth Shue", "Benny Hill",
    "Lou Costello", "Bud Abbott", "Albert Einstein", "Rich Hall",
    "Anthony Soprano", "Michelle Rodriguez", "Carmen Miranda",
    "Sandra Bullock", "Moe Howard", "Ulysses S. Grant", "Stephen Fry",
    "Kurt Vonnegut", "Yosemite Sam", "Ed Norton", "George Armstrong Custer",
    "Alice Kramden", "Evangeline Lilly", "Harlan Ellison"
    );

   $query = $_GET['query'];
   $temp = array();

   while (list($k,$v) = each ($people)){
        if (stristr ($v, $query)){
            array_push($temp, $v);
        }
    }
   $data = array("query" => $query, "result" => $temp);
   echo (json_encode($data));
?>
```

Exemplo 3 – Uso de APIs externas

- Requisições podem ser feitas de várias formas e para vários URLs, nesse exemplo, iremos fazer uma requisição para o serviço de geolocalização do google, o qual provê uma API (Application program interface) pública (https://developers.google.com/maps/documentation/geocoding/intro)
- Neste exemplo iremos fazer uma pesquisa de um endereço através de um campo texto de entrada. Ao clicar no botão de pesquisa, iremos executar um consulta com uso da função `getJSON()` através do padrão informado pela API

http://maps.googleapis.com/maps/api/geocode/***output*?*parameters***

https://maps.googleapis.com/maps/api/geocode/json?address=Itajuba

https://jsfiddle.net/3bugojsg/2/

Exercício

- Criar banco de dados e tabela Usuario com as seguintes colunas:
  - Código (chave primária)
  - Nome
  - Endereço
- Modificar o script PHP para acessar o banco e fazer a busca, retornando como resposta o modelo JSON a seguir:

Exercício


```
{
"query": "He"
"result": [
        {
        "id": 1,
        "nome" : "Helton",
        "endereco" : {
                        "rua": "Simão Mauad",
                        "numero": 123,
                        "bairro": "BPS",
                        "cidade": "Itajubá"
                      }
        },
        {
        "id": 15,
        "nome" : "Hellen",
        "endereco" : {
                        "rua": "Nelson Cruz",
                        "numero": 324,
                        "bairro": "Varginha",
                        "cidade": "Itajubá"
                      }
        }]
}
```

Bibliografia

- [1] http://w3techs.com/technologies/overview/javascript_library/all
- [2] http://spacetelescope.github.io/understanding-json-schema/
- https://www.metaweather.com/api/location/search/?query=san