SI401

Programação para a Web

2º Semestre - 2024

Comunicação Assíncrona: AJAX
Unidade 19

Prof. Guilherme Palermo Coelho
guilherme@ft.unicamp.br

Roteiro

- Introdução;
- O que é AJAX;
- A API *XMLHttpRequest*;
- Um exemplo simples usando JavaScript e PHP;
- Referências.

INTRODUÇÃO

Introdução

- Até aqui no curso, todas as aplicações que envolviam *front-end* e *back-end* interagiram de forma **síncrona**:

Introdução

- Em muitas situações, não queremos *sair da página atual*, mas precisamos que algum tipo de processamento seja feito no *back-end*;
- Exemplos:
  - Atualizar quantidade de itens em um carrinho de compras;
  - Arquivar uma mensagem em seu *webmail*;
  - Marcar um item como “lido” em seu leitor de notícias;
  - ...

Como fazer isso? → AJAX

O QUE É AJAX?

O que é AJAX?

Não, não é esse.

AJAX
Asynchronous Javascript And XML

- **AJAX** é um termo, criado em 2005, que descreve uma forma de se utilizar tecnologias já existentes (HTML, CSS, JavaScript, XML ...) [2]
  - Permite atualizações pontuais na interface, sem recarregar **toda** a página.

O que é AJAX?

- AJAX **não é** uma linguagem de programação!
- É uma combinação de tecnologias que permitem acessar um *servidor web* a partir de uma página:
  - Permite **ler** dados do servidor, mesmo após a página ter sido carregada;
  - Permite **enviar** dados ao servidor, em *background*;
  - Permite **atualizar** o conteúdo de uma página sem recarrega-la.
- AJAX utiliza principalmente as APIs **XMLHttpRequest (XHR)** e **Fetch**;
- Apesar do "X" vir de XML, não é necessário enviar/receber dados apenas neste formato: é possível utilizar texto e JSON (o mais comum), dentre outros.

O que é AJAX?

- Como AJAX funciona (usando XHR):

A comunicação continua usando o protocolo HTTP!

O cliente não precisa ficar "parado" esperando

Via JavaScript:
- Cria um objeto XMLHttpRequest;
- Envia um HTTPRequest.

Via JavaScript:
- Processa os dados retornados;
- Atualiza conteúdo da página.

Via PHP (nesta disciplina):
- Processa a requisição;
- Cria a resposta e envia os dados (caso existam) ao *browser*.

A API *XMLHttpRequest*

XMLHttpRequest (XHR)

- XHR [3] é uma API [1] que permite a interação entre páginas dinâmicas e servidores *web*.
  - API: *Application Programming Interface*
    - Conjunto de rotinas e padrões de programação que permitem o acesso a um *software* ou plataforma*;*
    - No nosso contexto, via *web*;

XMLHttpRequest (XHR)

- Todos os navegadores modernos possuem suporte a *objetos* XHR, que podem ser criados (via JavaScript) da seguinte forma:

```
variavel = new XMLHttpRequest();
```

- A partir do objeto criado, pode-se:
  - Disparar requisições HTTP (HTTP *requests*), de maneira síncrona ou assíncrona;
  - Detectar quando a requisição foi atendida;
  - Processar os dados recebidos.

XMLHttpRequest (XHR)

- Envio de requisições (sendo "*variavel*" o objeto XHR criado):

```
variavel.open('GET', 'http://www.example.org/some.file', true);
variavel.send();
```

- Método *open*() - inicializa a requisição:
  - **Primeiro parâmetro**: método de requisição HTTP a ser utilizado (GET, HEAD, POST ou outro suportado pelo servidor [4]);
  - **Segundo parâmetro**: URL para onde a requisição será enviada (por questões de segurança, você **só pode fazer requisições para o mesmo domínio** em que está hospedado o documento atual);
  - **Terceiro parâmetro (opcional)**: define se a requisição será *assíncrona (default = true),* ou seja, se a execução continuará enquanto o servidor não responde.

XMLHttpRequest (XHR)

- Envio de requisições (sendo *variavel* o objeto XHR criado):

```
variavel.open('GET', 'http://www.example.org/some.file', true);
variavel.send();
```

- Método *send*() – envia a requisição:
  - **Requisição Assíncrona**: método retorna assim que a requisição é enviada (não aguarda a resposta);
    - Aqui está o "A" de AJAX;
  - **Requisição Síncrona**: método aguarda a resposta do servidor;
    - Este modo deve ser usado com cuidado, pois tende a prejudicar muito a usabilidade.
  - **Parâmetro:** conteúdo a ser enviado ao servidor (depende do método).

XMLHttpRequest (XHR)

- Uma vez enviada uma requisição (assíncrona), deve-se definir qual função será chamada quando o servidor responder (sendo "*variável*" o objeto XHR criado):

```
variavel.onreadystatechange = nomeDaFuncao;
```

  - O nome da função deve ser passado sem parêntesis;
  - Tal função deve **verificar o estado da requisição** e, caso seja a constante XMLHttpRequest.DONE, o processamento pode continuar;

```
if (variavel.readyState === XMLHttpRequest.DONE) {
    // Requisição terminada, processamento pode continuar.
} else {
    // Ainda não terminou.
}
```

XMLHttpRequest (XHR)

- Além disso, mesmo que o estado da requisição seja XMLHttpRequest.DONE, é preciso verificar se o **código de resposta HTTP não é uma mensagem de erro**:

```
if (variavel.readyState === XMLHttpRequest.DONE) {
    if (variavel.status === 200) {
        // Tudo certo! Código OK retornado.
    } else {
        // Houve um problema com a requisição
        // Ex.: 404 (not found) ou 500 (internal server error)
    }
} else {
    // Ainda não terminou.
}
```

XMLHttpRequest (XHR)

- Por fim, uma vez verificado um *status* 200, pode-se processar os dados recebidos da maneira que for necessário;
- Para isso, existem dois métodos:
  - ***variavel*.responseText**: retorna a resposta do servidor como uma *string*;
  - ***variavel*.responseXML**: retorna a resposta do servidor como um objeto **XMLDocument**, que pode ser manipulado com funções próprias para isso.

UM EXEMPLO SIMPLES COM JAVASCRIPT E PHP

Um exemplo simples com JavaScript e PHP

- Para ilustrar a comunicação assíncrona entre um *front-end* baseado em JavaScript e um *back-end* em PHP, vamos considerar uma aplicação bem simples:
  - Uma *string* é fornecida pelo usuário no documento HTML;
  - Esta *string* é enviada ao servidor, via AJAX;
  - O servidor responde com um objeto JSON [5], que contém a mesma *string* enviada e uma nova *string* gerada.
    - JSON (*JavaScript Object Notation*) é um padrão para troca de dados, muito utilizado atualmente por ser de fácil leitura e escrita tanto para humanos quanto para máquinas;
    - Não discutirei os detalhes de JSON aqui – recomendo a leitura de [5].

Um exemplo simples com JavaScript e PHP

- Documento HTML:

```
<!DOCTYPE html>
<html lang="PT">
    <head>
        <title>Exemplo de AJAX</title>
        <meta charset="UTF-8">

        <script src="funcoes_ajax.js"></script>
    </head>

    <body>
        <h1>Teste de AJAX</h1>

        <label>Digite seu nome:
            <input type="text" id="ajaxTextbox" />
        </label>

        <button id="ajaxButton" onclick="enviarDados()">Clique aqui!</button>

    </body>

</html>
```

Um exemplo simples com JavaScript e PHP

- JavaScript (parte 1):

```
let xhttp;

function enviarDados() {
    let nomeUsuario = document.getElementById("ajaxTextbox").value;
    xhttp = new XMLHttpRequest();

    if (!xhttp) {
        alert('Não foi possível criar um objeto XMLHttpRequest.');
        return false;
    }
    xhttp.onreadystatechange = mostraResposta;
    xhttp.open('POST', 'teste.php', true);
    xhttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
    xhttp.send('userName=' + encodeURIComponent(nomeUsuario));
}
```

- Criação da variável que manterá o objeto XHR;
  - O comando *let* é mais atual que o *var,* e cria variáveis com escopo bem definido (global ou local).

Um exemplo simples com JavaScript e PHP

- JavaScript (parte 1):

```
let xhttp;

function enviarDados() {
    let nomeUsuario = document.getElementById("ajaxTextbox").value;
    xhttp = new XMLHttpRequest();

    if (!xhttp) {
        alert('Não foi possível criar um objeto XMLHttpRequest.');
        return false;
    }
    xhttp.onreadystatechange = mostraResposta;
    xhttp.open('POST', 'teste.php', true);
    xhttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
    xhttp.send('userName=' + encodeURIComponent(nomeUsuario));
}
```

- Obtém a *string* fornecida pelo usuário no <input>;
- Cria o objeto XHR.

Um exemplo simples com JavaScript e PHP

- JavaScript (parte 1):

```
let xhttp;

function enviarDados() {
    let nomeUsuario = document.getElementById("ajaxTextbox").value;
    xhttp = new XMLHttpRequest();

    if (!xhttp) {
        alert('Não foi possível criar um objeto XMLHttpRequest.');
        return false;
    }
    xhttp.onreadystatechange = mostraResposta;
    xhttp.open('POST', 'teste.php', true);
    xhttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
    xhttp.send('userName=' + encodeURIComponent(nomeUsuario));
}
```

- Verifica se o objeto XHR foi criado corretamente:
  - Caso não tenha sido, avisa o usuário.

Um exemplo simples com JavaScript e PHP

- JavaScript (parte 1):

```
let xhttp;

function enviarDados() {
    let nomeUsuario = document.getElementById("ajaxTextbox").value;
    xhttp = new XMLHttpRequest();

    if (!xhttp) {
        alert('Não foi possível criar um objeto XMLHttpRequest.');
        return false;
    }
    xhttp.onreadystatechange = mostraResposta;
    xhttp.open('POST', 'teste.php', true);
    xhttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
    xhttp.send('userName=' + encodeURIComponent(nomeUsuario));
}
```

- Associa a função “mostraResposta()” ao evento de alteração do estado da requisição.

Um exemplo simples com JavaScript e PHP

- JavaScript (parte 1):

```
let xhttp;

function enviarDados() {
    let nomeUsuario = document.getElementById("ajaxTextbox").value;
    xhttp = new XMLHttpRequest();

    if (!xhttp) {
        alert('Não foi possível criar um objeto XMLHttpRequest.');
        return false;
    }
    xhttp.onreadystatechange = mostraResposta;
    xhttp.open('POST', 'teste.php', true);
    xhttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
    xhttp.send('userName=' + encodeURIComponent(nomeUsuario));
}
```

- Monta a requisição assíncrona, usando o método POST
  - Temos a definição do cabeçalho da mensagem HTTP e, na sequência, o envio da *string* codificada como conteúdo da mensagem.

Um exemplo simples com JavaScript e PHP

- JavaScript (parte 2):

```
function mostraResposta() {
    try {
        if (xhttp.readyState === XMLHttpRequest.DONE) {
            if (xhttp.status === 200) {
                let resposta = JSON.parse(xhttp.responseText);
                alert(resposta.stringModificada + ' - Seu username é: ' +
resposta.userName);
            }
            else {
                alert('Um problema ocorreu.');
            }
        }
    }
    catch (e) {
        alert("Ocorreu uma exceção: " + e.description);
    }
}
```

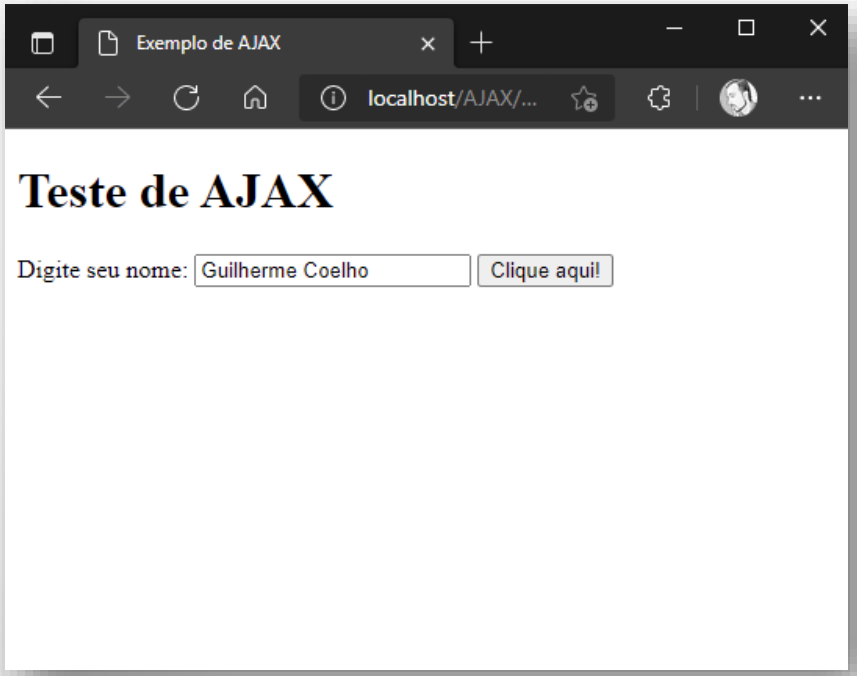
Um exemplo simples com JavaScript e PHP

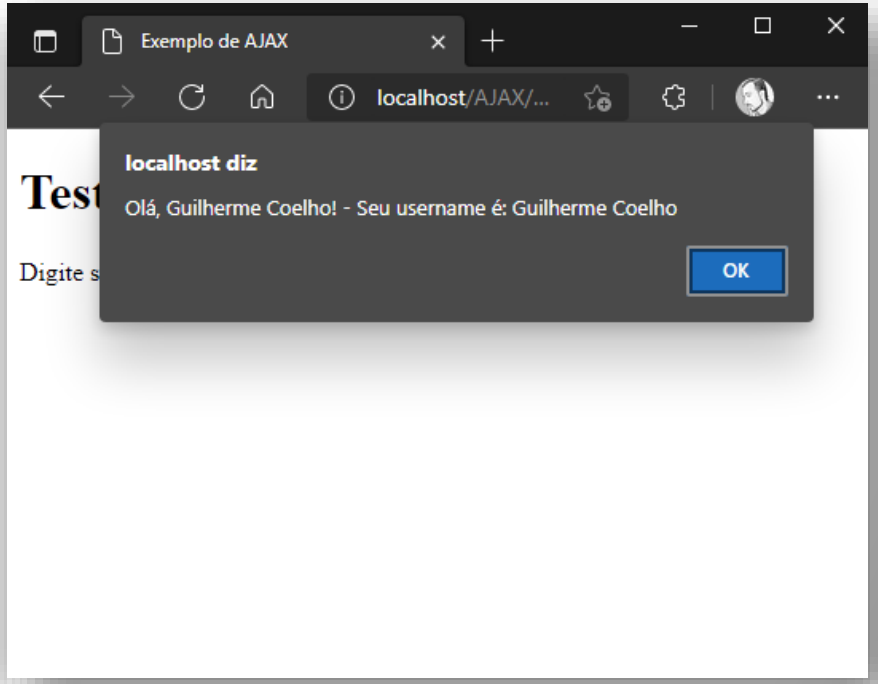
- PHP:

```
<?php
    $name = (isset($_POST['userName'])) ? $_POST['userName'] : 'nome vazio';
    $computedString = "Olá, " . $name . "!";
    $array = ['userName' => $name, 'stringModificada' => $computedString];
    echo json_encode($array);
?>
```

Um exemplo simples com JavaScript e PHP

- Execução:





Referências

Referências

[1] Vídeos sobre a definição de APIs:

https://www.youtube.com/watch?v=s7wmiS2mSXY (em inglês)

https://www.youtube.com/watch?v=3LHSyha0xN0 (em português)

[2] Mozilla Developer Network: AJAX
https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX

[3] Mozilla Developer Network: XMLHttpRequest

https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest

[4] Métodos de Requisição HTTP 1.1

https://www.rfc-editor.org/rfc/rfc9110.html#name-methods

[5] JSON: *JavaScript Object Notation*

https://www.json.org/json-en.html