Universidade Cruzeiro do Sul

AJAX

(Asynchronous JavaScript and XML)

com JSP

AJAX

- **AJAX** (acrônimo em língua inglesa de Asynchronous Javascript And XML) é o uso metodológico de tecnologias como Javascript e XML;
- **AJAX** não é uma linguagem de programação mas um modelo de programação para chamadas assíncronas usando tecnologias conhecidas;
- **AJAX** permite a construção de aplicações Web mais dinâmicas e criativas, permitindo a criação de efeitos visuais só possíveis anteriormente em aplicações desktop.

Conceitos

- Com AJAX podemos fazer com que o Javascript se comunique diretamente com o servidor usando um objeto **XMLHttpRequest,** implementado nos atuais web browsers;
- Com esse objeto, podemos trazer dados do servidor sem a necessidade de atualizar a tela;
- AJAX usa transferência de dados assíncrona (HTTP request) entre o browser e o servidor, permitindo que apenas alguns objetos sejam atualizados ao invés da página toda;
- AJAX é utiliza os seguintes padrões web:
  - Javascript
  - XML
  - HTML
  - CSS

XML

- XML significa **EX**tensible **M**arkup **L**anguage, linguagem de marcação extensível
- Linguagem muito parecida com o HTML
- É uma linguagem para definir dados, o arquivo XML possui somente os dados, a visualização pode ser feita por diversas outras linguagens
- Não possui tags definidas, o desenvolvedor cria sua própria definição de tags.
- O XML não substitui o HTML, pois possuem objetivos diferentes:
  - XML foi projetado para transportar e armazenar dados, ou seja, está focado nos dados da aplicação
  - HTML foi projetado para estruturar os dados, ou seja, está focado na visualização dos dados

Compartilhamento de dados

- Quando falamos em base de dados, podemos nos remeter aos chamados SGBDs como por exemplo: Oracle, MySQL, SQL Server, etc.
- Um problema entre essas bases é que cada uma possui seus formatos de dados, o que gera incompatibilidades.
- O XML armazena os dados no formato texto, o que não depende de software e hardware, facilitando o transporte de dados entre aplicações.
- Um único arquivo .xml pode ser utilizado em diversas aplicações, por exemplo:
  - JavaScript/HTML/CSS
  - Java
  - PHP
  - JSP
  - VB.net
  - ....
- Como utiliza o formato texto, um arquivo XML também pode ser acessado por diferentes tipos de dispositivos, como uma aplicação que roda em para smartphone.

Exemplo

```
<?xml version="1.0" encoding=“ISO-8859-1"?>
<veiculos>
   <carro>
        <fabricante> Ford </fabricante>
        <modelo> Escort </modelo>
        <motorista> Pedro </motorista>
   </carro>
   <carro>
        <fabricante> Honda </fabricante>
        <modelo> Fit </modelo>
        <motorista> Júlio </motorista>
   </carro>
</veiculos>
```

Observações

- Um arquivo XML começa com a linha

  <?xml version="1.0" encoding="“ISO-8859-1"?>

- Nesta linha definimos a versão e a codificação que será usada para a criação dos dados

- Um documento XML possui sempre um elemento raiz, no exemplo anterior, o elemento raiz seria a tag <veiculos>....</veiculos>

- O elemento raiz pode ter vários elementos filhos, como a tag <carro>...</carro> que aparece duas vezes

- O elemento carro, por sua vez possui três elementos filhos, que seriam: <fabricante>, <modelo> e <motorista>

Algumas regras na criação do XML

- O documento XML deve ter:
  - Apenas UM elemento raiz
  - Os atributos devem estar entre aspas
  - Todos os elementos devem ser fechados
  - Os elementos devem estar corretamente aninhados
  - O nome das *tags* é *case-sensitive*, logo, diferencia letras maiúsculas de letras minúsculas.

DOM (Document Object Model)

- Para manipularmos os dados do arquivo XML, iremos utilizar o DOM, que define padrões para acesso aos objetos do HTML e aos objetos do XML.

Algumas propriedades

- Propriedades fundamentais no modelo DOM
  - **childNodes** **=> A coleção de nós filhos**
  - **firstChild** **=> O primeiro filho**
  - lastChild => O último filho
  - **nodeValue** **=> O Valor do nó**
  - nextSibling => O próximo nó
  - previousSibling => O nó anteiror
  - nodeName => O nome do nó

Métodos do DOM

- Métodos fundamentais do modelo DOM

- getElementByTagName(nome):

Captura as tags pelo nome

- hasChildNodes():

Retorna se existe filhos

- getAttribute(nome):

Retorna o valor do atributo

Objeto XMLHttpRequest

- Inicialmente criado pela Microsoft e adaptado posteriormente pelos demais *browsers*.

- No Internet Explorer é usado através do **ActiveXObject**, nos demais *browsers* é usada a classe **XMLHttpRequest**

Propriedades do objeto XMLHttpRequest

- **readyState:** retorna o estado da requisição que pode ser:

  0 – não iniciada
  1 – carregando
  2 – carregado
  3 – processando
  4 – completada

- **status:** código de status retornado pelo servidor, por exemplo:

  200 - teve sucesso
  404 - página não encontrada

Propriedades do objeto XMLHttpRequest

- **responseText:** resposta no formato texto (HTML por exemplo)
- **responseXml:** resposta no formato XML (neste caso devemos manipular com o DOM)
- **onreadystatechange:** evento acionado a cada troca de estado da propriedade **readyState**, aqui devemos associar a uma função em javascript.

Principais Métodos do objeto XMLHttpRequest

- **open**(“método”, “url”, boolean)

  método: **GET** ou **POST**

  url: endereço a ser chamado ou arquivo a ser carregado

  boolean: **true** (assíncrono) / **false** (síncrono).

- **send**("string")

  null para o método GET

  string ou variável com os dados a serem enviados via método POST

Exemplo – index.jsp (ExemploAjax1)

Universidade Cruzeiro do Sul

```
<body>
    <h1>Busca Dados do Aluno</h1>
    <table border="1">
        <tr>
            <td>RGM:</td>
            <td>
                <input type="text" name="rgm" id="rgm" size="7">
                <input type="button" value="Pesquisar" onClick="Javascript:pesquisar()">
            </td>
        </tr>
        <tr>
            <td>Nome:</td>
            <td><input type="text" name="nome" id="nome" size="50"></td>
        </tr>
        <tr>
            <td>Turma:</td>
            <td><input type="text" name="turma" id="turma" size="7"></td>
        </tr>
    </table>
</body>
```

Busca Dados do Aluno

RGM:	Pesquisar
Nome:	
Turma:	

```
var xmlHttp;
function pesquisar() {

    //Pega o RGM para pesquisar
    var jsrgm = document.getElementById("rgm").value;
    if(jsrgm == "") {
        alert("Entre com o RGM do aluno para pesquisar");
        return;
    }

    //Cria url para a página que faz a pesquisa
    var url = "buscaaluno.jsp?rgm=" + jsrgm;

    // code for IE6, IE5
    if (window.ActiveXObject) {
        xmlHttp = new ActiveXObject('Microsoft.XMLHTTP');
    }
    //IE7+, Firefox, Chrome, Opera, Safari
    else if (window.XMLHttpRequest) {
        xmlHttp = new XMLHttpRequest();
    }
    else {
        alert("Navegador não suporta AJAX");
    }

    xmlHttp.open('GET', url, true);
    xmlHttp.onreadystatechange = capturaeventos;
    xmlHttp.send(null);

}
```

url que gera o xml

Abre a URL

Determina função que trata os eventos

Envia requisição

```
function capturaeventos(){
    //Mostra imagem de carregando na caixa
    if (xmlHttp.readyState == 1) {
        document.getElementById("nome").value = "Procurando...";
    }
    //Quando terminar de carregar a resposta
    if (xmlHttp.readyState == 4) {
        if (xmlHttp.status == 200) {
            //Captura a resposta do AJAX
            xmlDoc = xmlHttp.responseXML;
            jsnome = xmlDoc.getElementsByTagName("nome")[0].childNodes[0].nodeValue;
            jsturma = xmlDoc.getElementsByTagName("turma")[0].childNodes[0].nodeValue;

            //Colocando a resposta no formulário
            document.getElementById("nome").value = jsnome;
            document.getElementById("turma").value = jsturma;
        }
     }
}
```

0 - Não iniciado (Uninitialised)
1 - Carregando (Loading)
2 - Carregado (Loaded)
3 - Interativo (Interactive)
4 - Completado (Completed)

Pode ser responseText

Status	Descrição
200	OK
404	Página não encontrada
403	Não autorizado
407	Autenticação Proxy solicitada
408	Tempo esgotado
500	Erro interno do servidor

Exemplo – conecta.jsp

Universidade Cruzeiro do Sul

```jsp
<%@page import="java.sql.*" %>

<%
        Connection con = null;
        Statement stmt = null;

        try {
            //Banco de Dados
            String serverName = "localhost";
            String mydatabase = "aula";

            //Login e senha do banco
            String username = "root";
            String password = "root";

            // Carregando o JDBC Driver
            String driverName = "com.mysql.jdbc.Driver";
            Class.forName(driverName);

            // Criando a conexão com o Banco de Dados
            String url = "jdbc:mysql://" + serverName + "/" + mydatabase; // a JDBC url
            con = DriverManager.getConnection(url, username, password);
            stmt = con.createStatement();

        }
        catch (ClassNotFoundException e){
            //Driver não encontrado
            out.println("Driver não encontrado: " + e.toString());
        }
        catch (SQLException e) {
            //Não está conseguindo se conectar ao banco
            out.println("Erro ao executar SQL: " + e.toString());
        }

%>
```

Exemplo – buscaaluno.jsp

Universidade Cruzeiro do Sul

```
<%@include file="conecta.jsp" %>
<%@page contentType="text/xml"%>
<%@page pageEncoding="ISO-8859-1"%>
<%
    response.setHeader("Cache-Control", "no-cache");
    response.setHeader("Pragma", "no-cache");

    String rgm = request.getParameter("rgm");
    String sql = "SELECT aluno.nome, turma.nome FROM aluno INNER JOIN turma ";
    sql += "ON (aluno.turma = turma.codigo) WHERE aluno.codigo='" + rgm + "'";
    ResultSet rs = stmt.executeQuery(sql);
    out.println("<resposta>");
    if(rs.next()) {
        out.println("<nome>" + rs.getString("aluno.nome") + "</nome>");
        out.println("<turma>" + rs.getString("turma.nome") + "</turma>");
    }
    else {
        out.println("<nome>Não encontrado</nome>");
        out.println("<turma>N/C</turma>");
    }
    out.println("</resposta>");
    rs.close();
%>
```

Indica que a resposta é em XML

Não guarda cache da página

Monta o XML de resposta