

FÁBRICA DE DESENVOLVIMENTO

FÁBRICA DE DESENVOLVIMENTO - 2018

Prof. Thiago T. I. Yamamoto

#16 - AJAX E JSON



thiagoyama



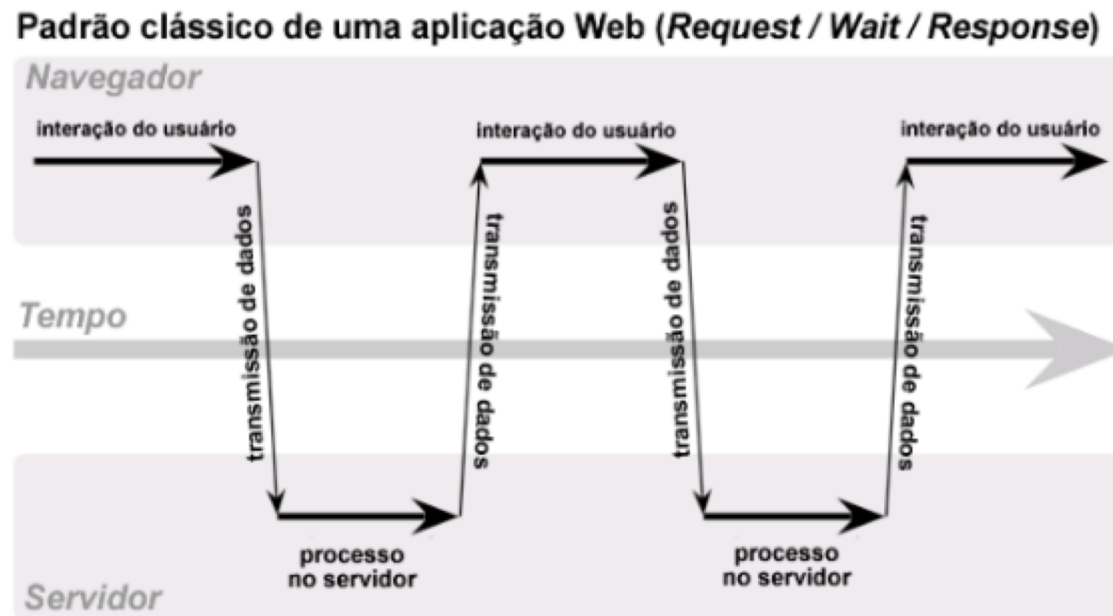
thiagoyama@gmail.com

#16 - AJAX E JSON

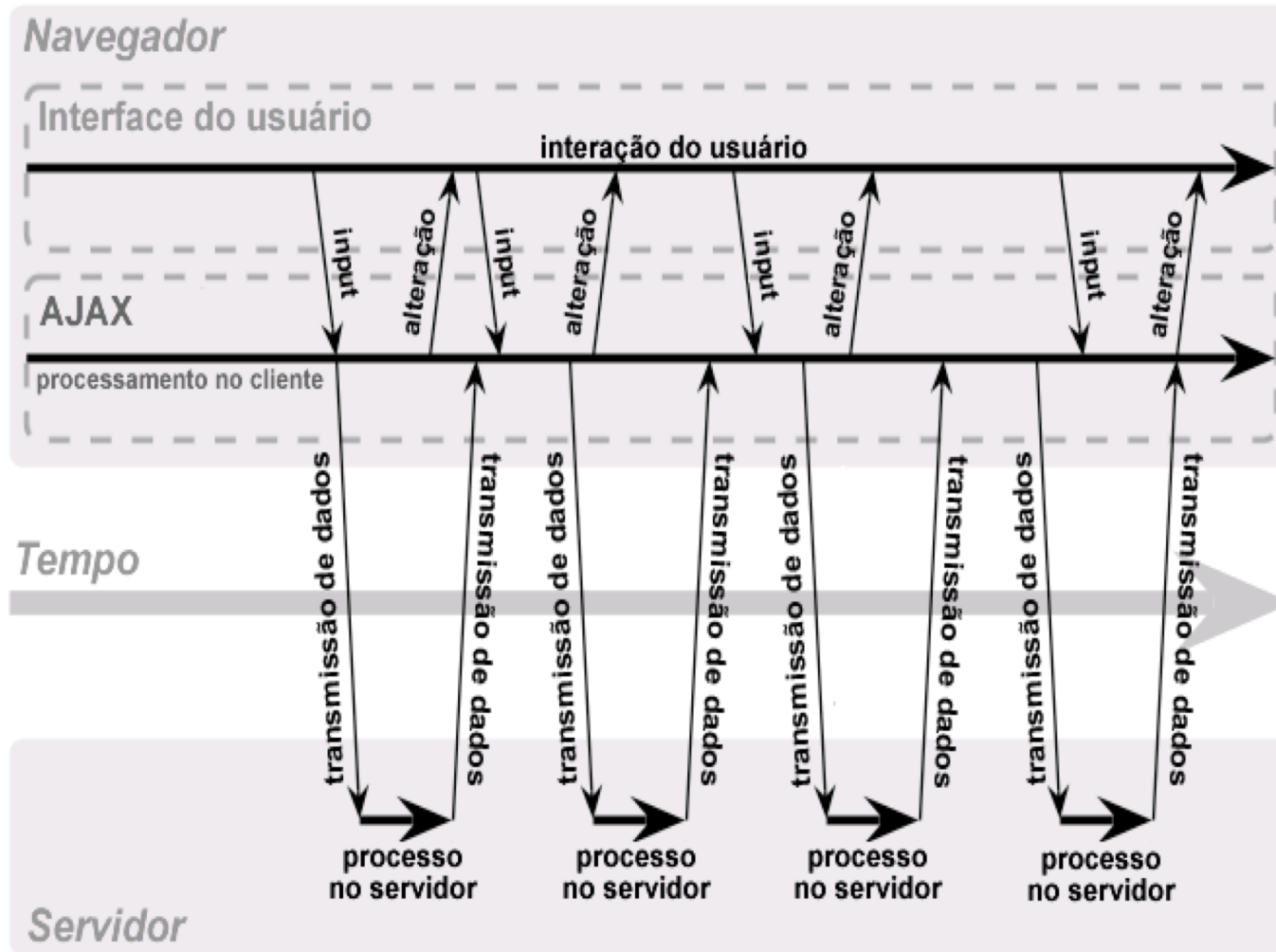
- Ajax
- Ajax Helpers
- JSON
- JQuery

AJAX

- Não é uma nova linguagem de programação;
- Utiliza padrões e tecnologias existentes para realizar troca de dados entre a página e o servidor de forma assíncrona, sem a necessidade de atualizar toda a página.



Aplicação web utilizando AJAX

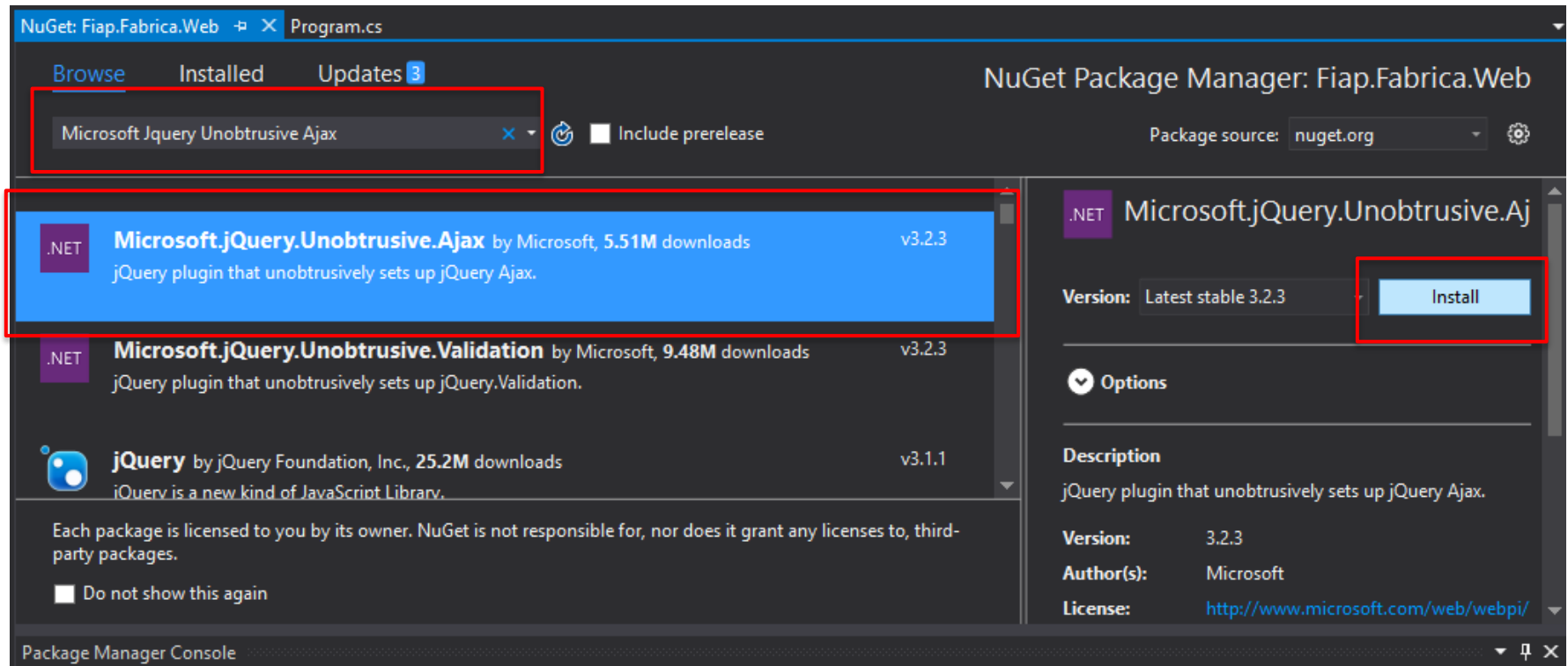


- Podemos utilizar Ajax Helpers para criar um formulário que será enviado via Ajax;

```
@using (Ajax.BeginForm(  
    new AjaxOptions  
    {  
        HttpMethod = "Get",  
        UpdateTargetId = "tabela",  
        InsertionMode = InsertionMode.Replace  
    })  
{  
    <input type="text" name="nome" />  
    <input type="submit" value="Enviar" />  
}
```

CONFIGURAÇÃO NUGET

- Precisamos instalar o **Microsoft JQuery Unobtrusive Ajax**;

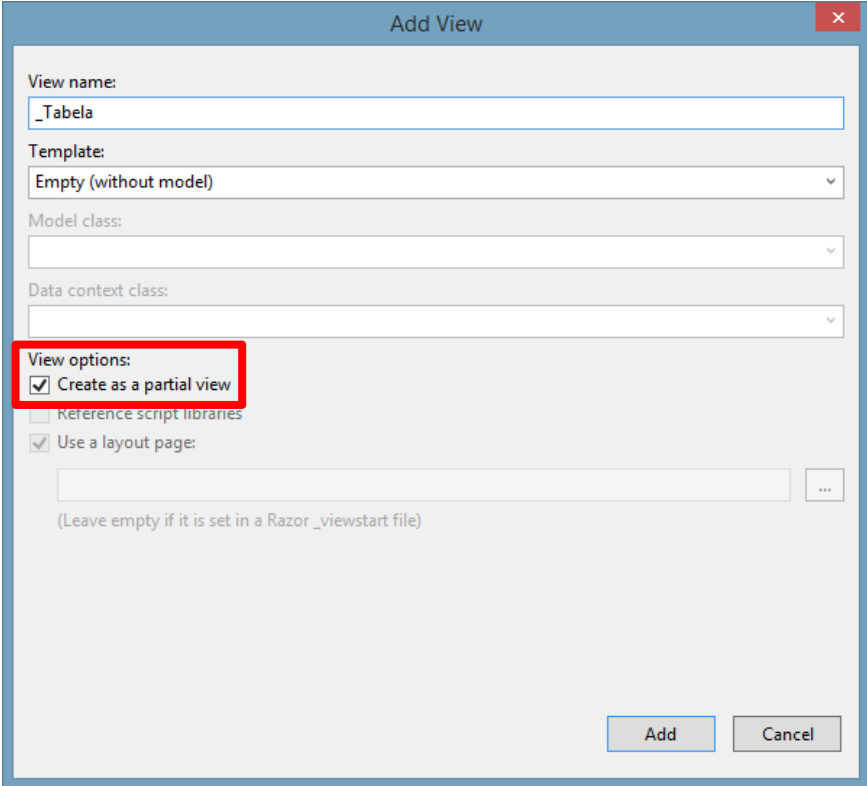


- E adicionar o .js:

```
<script src="~/Scripts/jquery.unobtrusive-ajax.min.js"></script>
```


PARTIAL VIEW

- Para atualizar parte da pagina, vamos criar uma partial view;
- Add -> View
- Marque a opção:
 - Create as partial view



The screenshot shows the 'Add View' dialog box. The 'View name' field contains '_Tabela'. The 'Template' dropdown is set to 'Empty (without model)'. The 'Model class' and 'Data context class' fields are empty. In the 'View options' section, the checkbox 'Create as a partial view' is checked and highlighted with a red box. Other options like 'Reference script libraries' and 'Use a layout page' are unchecked. The 'Add' button is at the bottom right.

- Na view, adicione a partial view:

```
@Html.Partial("_Tabela", Model)
```

- Na action, temos que retornar a partial view;

```
public ActionResult Buscar(string nome)
{
    var lista =
        _unit.ClienteRepository.SearchByName(nome);
    return PartialView("_Tabela", lista);
}
```

- Podemos utilizar Jquery para realizar requisições no servidor (Controller);

```
$.ajax({  
    url: "http://localhost:3262/Cliente/EmailJaExiste",  
    type: "GET",  
    data: { email: $("#Email").val() },  
    success: function (data) {  
        if (!data.ok) {  
            alert("OK!");  
        } else {  
            alert("Erro!");  
        }  
    },  
    error: function () {  
        alert("Erro ao requisitar o servidor");  
    }  
});
```

- A action pode retornar um JSON;
- **JsonRequestBehavior.AllowGet:** permiti responder uma chamada GET com JSON;

```
public ActionResult EmailJaExiste(string email)
{
    var busca = _unit.ClienteRepository.SearchFor(c => c.Email == email);
    return Json(new {ok = busca.Any()}, JsonRequestBehavior.AllowGet);
}
```

- **JSON - JavaScript Object Notation.**

- Formato simples e leve para transferência de dados.
- Uma alternativa para o XML

Exemplo:

```
{  
  "show": "Oasis",  
  "preco": 150,  
  "local": "São Paulo"  
}
```

Validador de formato Json: <http://jsonlint.com/>

Exemplo de uma lista de Shows:

```
{
  "shows": [
    {
      "show": "Oasis",
      "preco": 150,
      "local": "São Paulo"
    },
    {
      "show": "Link Park",
      "preco": 250,
      "local": "Rio de Janeiro"
    },
    {
      "show": "Jorge e Mateus",
      "preco": 200,
      "local": "São Paulo"
    }
  ]
}
```

{JSON}

Os colchetes [] limitam o array

- Podemos utilizar o JQuery para criar elementos HTML;
- Sintaxe: **\$("tag")**:
 - **\$("<tr>")**: cria um elemento tr;
 - **\$("<p>")**: cria um elemento p;
- Depois de criar o elemento, precisamos adiciona-lo na página, para isso podemos utilizar a função **append()**:
 - **\$("table > tbody").append(\$("<tr>"))**;
- A função **empty()** é utilizada para remover todos os filhos do elemento:
 - **\$("table > tbody").empty()**;

EXEMPLO

- Controller que retorna uma lista em JSON:
- **JsonRequestBehavior.AllowGet:** permiti responder uma chamada GET com JSON;

```
public ActionResult BuscarPlanos()
{
    var lista = _unit.PlanoRepository.List().Select(
        c => new {
            id = c.PlanoId,
            nome = c.Nome,
            valor = c.Valor });
    return Json(lista , JsonRequestBehavior.AllowGet);
}
```

- JQuery faz uma chamada Ajax e cria elementos , adicionando-os na lista:

```
$.ajax({  
  url: "http://localhost:3262/Cliente/BuscarPlanos",  
  type: "GET",  
  success: function (data) {  
    $("ul").empty();  
    $.each(data, function(index, plano){  
      var linha = $("<li>").text(plano.nome);  
      $("ul").append(linha);  
    });  
  }  
});
```

Copyright © 2018 - Prof. Me. Thiago T. I. Yamamoto

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).

“O primeiro passo rumo ao sucesso é dado quando você se recusa ao ser um refém do ambiente em que se encontra”