02

AJAX: por páginas mais dinâmicas

Melhorando a interface com AJAX

Vamos agora deixar nossa aplicação um pouco mais elegante. Relembremos daqueles sites onde clicamos e o conteúdo muda, sem a página "piscar", ou seja, sem a necessidade de reatualizar o navegador.

Para que isso aconteça, precisamos fazer uso de **AJAX**. AJAX é o nome que se dá ao uso de Javascript e XML para possibilitar requisições por baixo dos panos no navegador e deixar as páginas mais interativas.

Preparação da *action* e JSP

Em primeiro lugar, implementamos a ação no controlador de produtos que removerá um produto a partir do seu *id*:

```
public void remove(Produto produto) {
    produtos.remove(produto);
}
```

Agora vamos fazer o link na `lista.jsp` . Dentro da tabela, colocamos um link:

```
<td><a href="">Remover</a></td>
```

O problema é que se inserirmos o link normalmente, o navegador fará uma *requisição síncrona*, igual a que estamos acostumados. Vamos ter que fazer uso de AJAX para isso.

Instalação do jquery

Para fazer uma requisição AJAX, faremos uso de uma biblioteca chamada **jQuery**. Ela facilita bastante o trabalho do desenvolvedor. A biblioteca já está no projeto, na pasta `WebContent/js/jquery-1.6.1.min.js` . A versão mais atual você pode ser baixada pelo link: (http://jquery.com/)http://jquery.com (http://jquery.com)

Para usar a biblioteca do jQuery é preciso incluí-la no arquivo `lista.jsp` , dentro da tag `<head>` :

```
<script type="text/javascript" src="js/jquery-1.6.1.min.js"></script>
```

Para não ter problema com o contexto, usaremos a tag `c:url` :

```
<script type="text/javascript" src="<c:url value="/js/jquery-1.6.1.min.js" />"></script>
```

Enviando uma requisição com *jquery*

Vamos ao código javascript que envia uma requisição HTTP GET usando a função `$.get` do jquery. Nela, será necessário definir a URL com o parâmetro. O segundo parâmetro é uma função que é chamada automaticamente quando a requisição foi executada com sucesso:

```
<script type="text/javascript">
```

```
function remover(id) {
    $.get('remove?produto.id=' + id, function() {
        alert('removido com sucesso');
    });
}
</script>
```

No link, chamamos a função `remover`. Repare que passamos a ID do produto como parâmetro da função:

```
<td><a href="javascript:void(0);" onclick="remover(${produto.id})">Remover</a></td>
```

Por último, é preciso mexer mais uma vez na classe `ProdutoController` para passar o que a *action* deve retornar. Na verdade ela não retornará nada, certo? Ela não retorna um HTML convencional, não queremos chamar um JSP. Isso é indicado ao VRaptor pelo método `nothing()` do objeto `Result`:

```
public void remove(Produto produto) {
    produtos.remove(produto);
    result.nothing();
}
```

Testando a aplicação

Vamos até a listagem, clicamos em "Remover". O *alert* apareceu! Vamos atualizar a página para garantir que foi, de fato, excluído.

Repare que a requisição foi até o servidor, o VRaptor processou essa requisição, e devolveu ao javascript. Tudo isso sem a necessidade de "piscar" a tela do navegador.

Atualização da tela com jquery

Porém, o ideal seria fazer com que a linha da tabela sumisse. Para isso, vamos usar um pouco mais de javascript. Primeiro, damos um nome para cada uma das linhas:

```
<tr id="produto-${produto.id}">
```

Agora, fazemos com que, quando a requisição ajax voltar, a linha se apague:

```
<script type="text/javascript">

function remover(id) {
    $.get('remove?produto.id=' + id, function() {
        $('#produto-' + id).hide();
        alert('Produto removido com sucesso');
    });
}

</script>
```

Pronto. Agora é só testar novamente no navegador.