10

Para saber mais: Forward e Redirect. Qual a diferença?

Após submetermos um formulário (quando clicamos em um `h:commandLink` também estamos submetendo um formulário) é necessário redirecionar para uma outra página para mostrar o devido *feedback* ao usuário. Como foi visto neste capítulo, podemos optar em utilizar o `forward` ou o `redirect`. Não se preocupe se você está um pouco confuso nesses conceitos, pois agora vamos falar um pouco mais sobre eles:

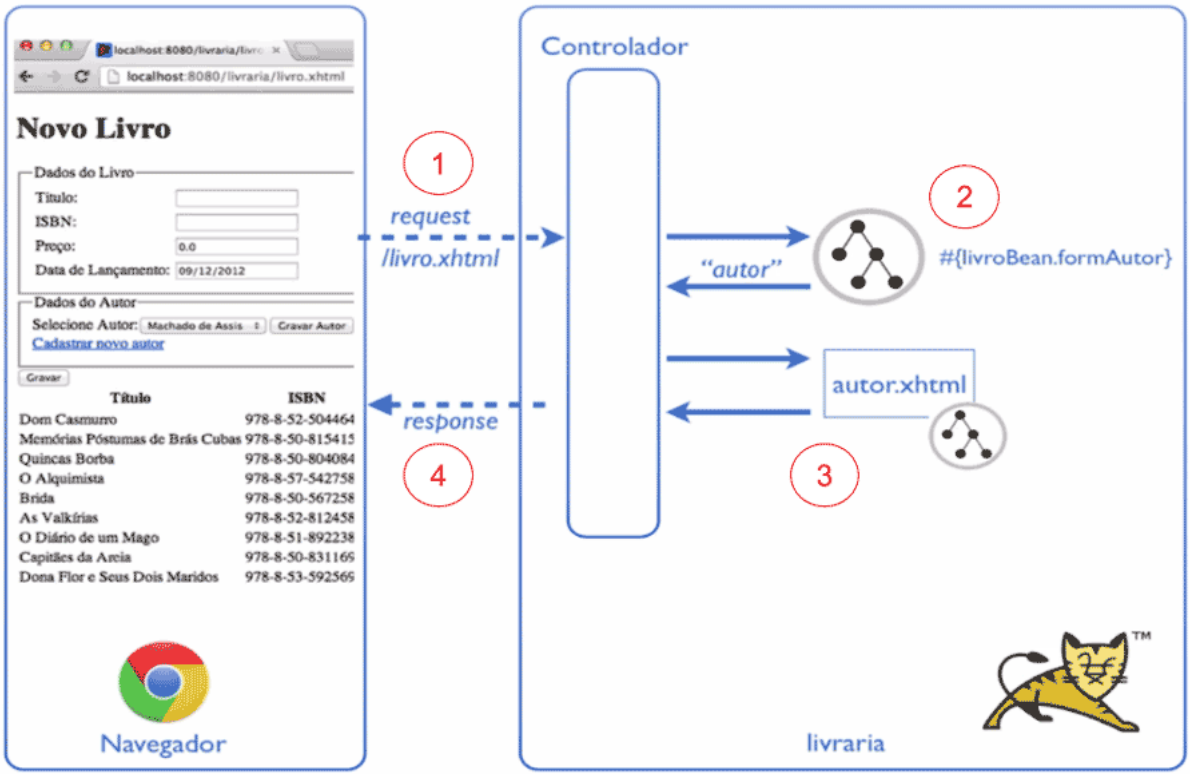
Redirecionamento no lado do servidor (forward)

Após o usuário realizar a requisição, o controlador do JSF irá chamar o método do nosso `ManagedBean` e aguardará o seu retorno. Após isso o controlador irá renderizar a página, cujo nome foi retornado pelo método e a enviará prontinha para o cliente como resposta.

Vamos analisar o fluxo na nossa aplicação? O que acontece ao clicar no botão **Cadastrar novo autor**?

```
<!-- livro.xhtml -->
<h:commandLink value="Cadastrar novo autor" action="#{livroBean.formAutor}" immediate="true"/>
```

```
// LivroBean.java
public String formAutor() {
    System.out.println("Chamanda o formulario do Autor");
    return "autor";
}
```



1) O cliente dispara uma requisição (submete o formulário) para o controlador;

2) O método `formAutor()` é chamado pelo controlador que armazena seu retorno;

3) O controlador utilizará a `String` retornada pelo método para buscar a `View` que deverá ser renderizada. Ao encontrá-la, ele irá realizar o processo de renderização;

4) Após a renderização, o seu HTML será retornado como resposta da requisição feita pelo usuário no passo 1 (a submissão do formulário).

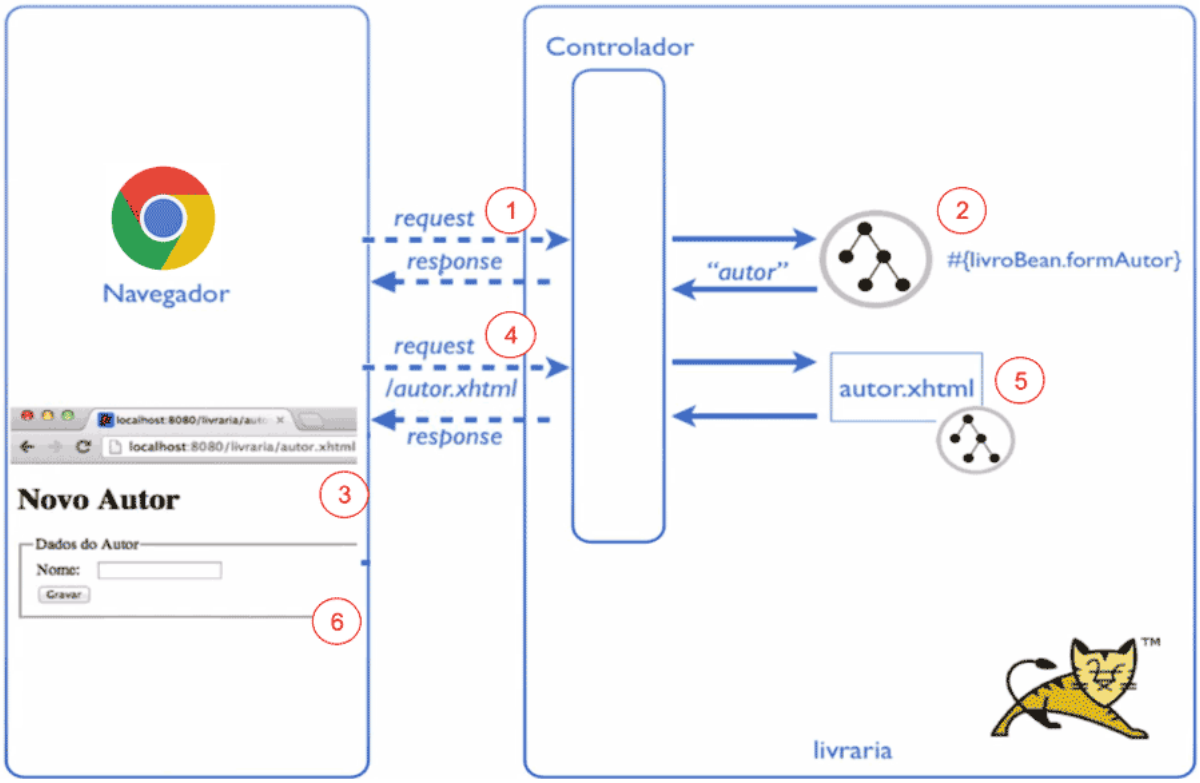
Como toda requisição deve haver uma resposta, ao submetermos o formulário também precisamos de uma resposta. Só que nesse caso a resposta será o HTML de uma **outra página**. Por conta disso, o navegador nem sequer percebe que houve uma mudança de página e a URL na barra de endereços continua a mesma de antes da submissão.

Redirecionamento no lado do cliente (redirect)

Ao delegarmos o redirecionamento ao cliente, o servidor não devolverá mais a página renderizada como resposta mas pedirá ao **cliente** para que acesse a nova página.

Para dizer ao JSF que queremos realizar um redirecionamento no lado do cliente, precisamos adicionar o parâmetro `faces-redirect=true` :

```
// LivroBean.java
public String formAutor() {
     System.out.println("Chamanda o formulario do Autor");
     return "autor?faces-redirect=true";
}
```



1) O cliente dispara uma requisição (submete o formulário) para o controlador;

2) O método `formAutor()` é chamado pelo controlador;

3) O Controlador do JSF retorna a resposta ao navegador com o status 302 *(moved temporarily)*. Esse código diz ao browser que ele precisará acessar a URL contida no *header* location da resposta.

4) O navegador, portanto, irá acessar a página contida no header Location

5) O controlador identifica a View e realiza o processo de renderização

6) O HTML da página é retornado ao browser.

Dessa forma o próprio navegador fica responsável por realizar o redirecionamento acessando outra página e por conta disso, a URL da barra de endereços é a da nova página que recebemos como resposta.

Podemos citar alguma vantagem em utilizar o **redirecionamento no lado do cliente** ao submeter um formulário de cadastro? Pesquise sobre a técnica *Always redirect after Post* e responda com suas próprias palavras.

Responda

INSERIR CÓDIGO | FORMATAÇÃO

[sua resposta]