02

Preparando ResourceBundle

Transcrição

ATENÇÃO

Ao final do vídeo 1 o arquivo `.properties` estava no singular (`message.properties`), porém ao início do segundo o mesmo estava no **plural** (`messages.properties`). Para evitar erros, estamos mantendo o padrão como `messages.properties` a partir desta atividade.

Terminamos a aula anterior com as validações já funcionando, mesmo que não exibindo as mensagens. Faremos isto agora de uma forma muito simples. Primeiro vamos abrir nosso arquivo `form.jsp` e adicionar uma nova `taglib` . Veja como está nosso `form.jsp` :

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Livros de Java, Android, iPhone, Ruby, PHP e muito mais - Casa do Código</title>
</head>
<body>
    <form action="/casadocodigo/produtos" method="post">
        <div>
            <label>Título</label>
            <input type="text" name="titulo" />
        </div>
        <div>
            <label>Descrição</descricao>
            <textarea rows="10" cols="20" name="descricao"></textarea>
        </div>
        <div>
            <label>Páginas</label>
            <input type="text" name="paginas" />
        </div>
        <c:forEach items="${tipos}" var="tipoPreco" varStatus="status">
            <div>
                <label>${tipoPreco}</label> <input type="text"
                    name="precos[${status.index}].valor" /> <input type="hidden"
                    name="precos[${status.index}].tipo" value="${tipoPreco}" />
            </div>
        </c:forEach>
        <button type="submit">Cadastrar</button>
    </form>
</body>
</html>
```

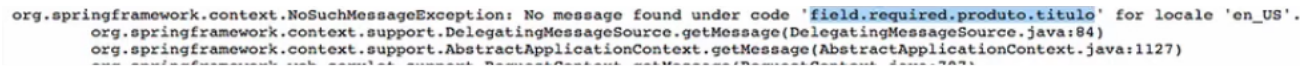
A `taglib` que adicionaremos será uma do próprio **Spring**. A `uri` da `taglib` será: `http://www.springframework.org/tags/form` e o `prefix` será `form`. Dessa forma teremos nosso `form.jsp` da seguinte forma:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Livros de Java, Android, iPhone, Ruby, PHP e muito mais - Casa do Código</title>
</head>
<body>
    <form action="/casadocodigo/produtos" method="post">
        [...]
    </form>
</body>
</html>
```

Com a `taglib` adicionada em nossa página JSP, podemos então usar uma **tag** dessa biblioteca para exibir as mensagens de erro. A tag usada para isso é a tag `form:errors` e ela tem um atributo chamado `path` que indica de qual atributo queremos obter a mensagem de erro. Sendo assim, podemos escrever algo do tipo: `<form:errors path="produto.titulo" />`. Simples! Não? Vamos fazer o mesmo para os outros campos, colocando a *tag* acima do campo a que o erro se refere. Vejamos como o código do formulário ficará:

```
<form action="/casadocodigo/produtos" method="post">
    <div>
        <label>Título</label>
        <form:errors path="produto.titulo" />
        <input type="text" name="titulo" />
    </div>
    <div>
        <label>Descrição</label>
        <form:errors path="produto.descricao" />
        <textarea rows="10" cols="20" name="descricao"></textarea>
    </div>
    <div>
        <label>Páginas</label>
        <form:errors path="produto.paginas" />
        <input type="text" name="paginas" />
    </div>
    <c:forEach items="${tipos}" var="tipoPreco" varStatus="status">
        <div>
            <label>${tipoPreco}</label> <input type="text"
                name="precos[${status.index}].valor" /> <input type="hidden"
                name="precos[${status.index}].tipo" value="${tipoPreco}" />
        </div>
    </c:forEach>
    <button type="submit">Cadastrar</button>
</form>
```

Agora nossas mensagens devem aparecer no formulário caso cadastremos um produto que não seja válido, ou seja, sem título, descrição ou sem páginas. Vamos testá-lo então. Submeta o formulário sem preencher os campos corretamente!



Deu erro? Como assim? Veja a mensagem de erro: **No message found under code field.required.produto.titulo**. O **Spring** não está encontrando nossas mensagens de erro. Em lugar nenhum definimos isso. Se você já conhece, existem os arquivos `properties` que são comuns de serem usados nesses casos. Vamos criar então um `properties` para deixarmos as mensagens para o formulário neste arquivo.

Na pasta `WEB-INF` vamos criar um arquivo de texto chamado `messages.properties` e associar as chaves dos erros aos valores, ou seja, associar as chaves dos erros às mensagens. Dentro desse arquivo, podemos ter algo do tipo `field.required.produto.titulo = O Campo título é obrigatório .`

Por hora, para testarmos, vamos deixar só a mensagem referente ao título. Depois adicionaremos o restante dos campos. Agora, precisamos configurar o **Spring** para que ele encontre esse nosso arquivos de mensagens. Já temos uma classe de configuração: a `AppWebConfiguration` .

Na classe `AppWebConfiguration` criaremos um novo método que carregará nossos arquivos de mensagens. Este método se chama `messageSource` e retorna um objeto do tipo `MessageSource` . Dentro deste método criaremos um objeto do tipo `ReloadableResourceBundleMessageSource` que chamaremos de `messageSource` . Neste objeto, iremos definir três propriedades: `setBaseName` com o valor `/WEB-INF/message` que terá o nome base dos nossos `resources` . O `setDefaultEncoding` com o valor `UTF-8` para evitar o problema de caracteres estranhos que já vimos outras vezes e o `setCacheSeconds` para que o **Spring** recarregue o arquivo de tempos em tempos com o valor `1` .

Esta última propriedade é muito útil em desenvolvimento pois poderemos ficar sempre modificando as mensagens sem se preocupar em ficar refazendo ***reload*** do arquivo manualmente. Nosso método fica da seguinte forma: **Observação**: Lembre-se de anotar esse método com a anotação `@Bean` para que o **Spring** possa reconhecer essa configuração.

```
@Bean
public MessageSource messageSource(){
    ReloadableResourceBundleMessageSource messageSource = new ReloadableResourceBundleMessageSou
    messageSource.setBasename("/WEB-INF/messages");
    messageSource.setDefaultEncoding("UTF-8");
    messageSource.setCacheSeconds(1);
    return messageSource;
}
```