

Tiago Ramos

ramos.tiago@gmail.com

<http://tramos.railsplayground.net>

MIRANTE TECNOLOGIA

CURSO STRUTS2

Temas e Templates

- São a base das tags Struts
 - Templates: trechos de código geralmente escritos em Freemarker
 - (<http://struts.apache.org/2.0.11.1/docs/freemarker.html>)
 - Temas: conjunto de templates que fornecem funcionalidades em comum

Temas e Templates

- Struts2 possui quatro temas básicos
 - simple
 - Tema básico geralmente tomado como ponto de partida para construção de outros
 - xhtml
 - Criado a partir do simple tema
 - Label para os campos
 - Validação e exibição de erros
 - Validação utilizando javascript
 - Padrão de layout com duas colunas por tabela para formulário (tags struts)

Temas e Templates

- Struts2 possui quatro temas básicos
 - `css_xhtml`
 - Construído a partir do `xhtml`
 - Layout baseado em `div`
 - `ajax`
 - Ainda é experimental
 - Estende de `xhtml` tema
 - Validação com `ajax`
 - Mecanismos de de recarga dinâmica em `div's`
 - Eventos com `ajax`

Temas e Templates

- Definindo temas na sua aplicação
 - Defina o atributo theme para uma tag específica
 - Defina para todo o formulário
 - Defina para toda a aplicação no arquivo struts.properties
 - struts.ui.theme = simple | ajax | xhtml
 - Verifique
 - <http://struts.apache.org/2.0.11.1/docs/strutsproperties.html>
- Importando propriedades do tema para sua página

```
<head>  
  <s:head/>  
</head>
```

Temas e Templates

- Para construção de novos temas verifique
 - <http://struts.apache.org/2.0.11.1/docs/extending-themes.html>

Temas e Templates

- Tente agora você...

Validação

- Tem como base o mecanismos de validação da Xwork
- A validação é realizada antes da ação ser executada
 - Interceptors
 - validation
 - workflow
- Existem duas formas de validar
 - No servidor
 - No cliente

Validação no servidor

- Para sua classe de ação defina
 - <ActionClassName>-validation.xml
 - <ActionClassName>-<ActionAliasName>-validation.xml
- Temos vários validadores padrão
 - required, requiredstring, stringlength
 - int, long, short, double
 - date
 - expression, fieldexpression, regex, conversion
 - email, url, visitor

Validação no servidor

- Exemplo de validação

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE validators PUBLIC "-//OpenSymphony Group//XWork Validator 1.0.2//EN"
    "http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">
<validators>
  <field name="novaQuestao.pergunta">
    <field-validator type="requiredstring">
      <message>campo obrigatório</message>
    </field-validator>
  </field>
  <field name="novaQuestao.resposta">
    <field-validator type="requiredstring">
      <message>campo obrigatório</message>
    </field-validator>
  </field>
</validators>
```

Validação no servidor

- Para sua classe de ação defina
 - <ActionClassName>-validation.xml
 - <ActionClassName>-<ActionAliasName>-validation.xml
- Temos vários validadores padrão
 - required, requiredstring, stringlength
 - int, long, short, double
 - date
 - expression, fieldexpression, regex, conversion
 - email, url, visitor

Validação no servidor

■ Exemplos

```
<field-validator type="int">  
  <param name="min">6</param>  
  <param name="max">10</param>  
  <message>bar must be between ${min} and ${max}, current value is {bar}.</message>  
</field-validator>
```

```
<field-validator type="regex">  
  <param name="regex">[0-9],[0-9]</param>  
  <message>Formato v;alido é "x, y", onde x e y estão entre 0 e 9</message>  
</field-validator>
```

Validação no servidor

■ Exemplos

```
<field-validator type="date">  
  <param name="min">12/22/2002</param>  
  <param name="max">12/25/2002</param>  
  <message>The date must be between 12-22-2002 and 12-25-2002.</message>  
</field-validator>
```

```
<validator type="expression">  
  <param name="expression">p1 lt p2 </param>  
  <message>Parametro 1 deve ser superior ao 2. P1 = ${p1}, P2 = ${p2}.</message>  
</validator>
```

Validação no cliente

- Alteração ocorre na tag do formulário
 - `<s:form method="post" validate="true" action="/namespace/validation.action">`
 - Observe que o namespace deve ser informado juntamente com a action
- Para quem quer andar mais...
 - <http://struts.apache.org/2.0.11.1/docs/ajax-validation.html>

Validação com Annotations

- Anote a classe
 - @Validation
- Anote os métodos set's
 - @RequiredFieldValidator(type = ValidatorType.FIELD, message = "message")
 - @IntRangeFieldValidator(type = ValidatorType.FIELD, min = "6", max = "10", message = "message")
- ou os métodos de ação
 - @Validations(requiredFields = {}, requiredStrings = {}, ...)

Validação com Annotations

- Veja mais...

- <http://struts.apache.org/2.0.11.1/docs/annotations.html>

Annotation	Description
ConversionErrorFieldValidator Annotation	Checks if there are any conversion errors for a field.
DateRangeFieldValidator Annotation	Checks that a date field has a value within a specified range.
DoubleRangeFieldValidator Annotation	Checks that a double field has a value within a specified range.
EmailValidator Annotation	Checks that a field is a valid e-mail address.
ExpressionValidator Annotation	Validates an expression.
FieldExpressionValidator Annotation	Uses an OGNL expression to perform its validator.
IntRangeFieldValidator Annotation	Checks that a numeric field has a value within a specified range.
RegexFieldValidator Annotation	Validates a regular expression for a field.
RequiredFieldValidator Annotation	Checks that a field is non-null.
RequiredStringValidator Annotation	Checks that a String field is not empty.
StringLengthFieldValidator Annotation	Checks that a String field is of the right length.
StringRegexValidator Annotation	Invokes a regular expression to validate a String field.
UrlValidator Annotation	Checks that a field is a valid URL.
Validation Annotation	Marker annotation for validation at Type level.
Validations Annotation	Used to group validation annotations.
VisitorFieldValidator Annotation	Invokes the validation for a property's object type.
CustomValidator Annotation	Use this annotation for your custom validator types.