Struts 2

Contents

- How to access Servlet API
- Internationalization
- Input Validation

How to Access Servlet API

- access the static parameters
- Accessing Session Object
  - **org.apache.struts2.interceptor.SessionAware** : Actions that need access to the user's HTTP session should implement this interface.
- To access the request object
  - use the ***ActionContext***
  - implement the ***org.apache.struts2.interceptor.ServletRequestAware*** interface
- To access the response object
  - use the ***ActionContext***
  - implement the ***org.apache.struts2.interceptor.ServletResponseAware*** interface.

Internationalization

Internationalization

- *Internationalization* is the process of designing an application so that it can be adapted to various languages and regions without engineering changes.

Resource files

- Create a resource file for a default language
  - MyApplication.properties
  - Contains the messages in the default language for your server
  - If your default language is English, you might have an entry like this:
    - prompt.hello=Hello
- Create resource files for each language you want to support
  - MyApplication_xx.properties
  - Contains the same messages in the language whose language code is "xx"

Input Validation

Input Validation

- Struts2 Validation Framework allows us to separate the validation logic from actual Java/JSP code, where it can be reviewed and easily modified later.
- it can handle both **server side** as well as **client side** form validation.
- create your own validation logic by implementing java interface com.opensymphony.xwork2.Validator

Client Validation

```
<s:form action="Login" validate="true">
        <s:textfield key="username"/>
        <s:password key="password" />
        <s:submit/>
</s:form>
```

Input Validation

- There are two types of Validators in Struts2 Validation Framework.
  - Field Validators
  - Non-field validators

Field validators

```
<validators>
    <field name="bar">
            <field-validator type="required">
                    <message>You must enter a value for bar.</message>
            </field-validator>
  </field>
</validators>
```

Non-field validators

```
<validator type="required">
   <param name="fieldName">bar</param>
   <message>You must enter a value for
   bar.</message>
</validator>
```

Defining Validation Rules

- Per Action class
  - in a file named <ActionName>-validation.xml
- Per Action alias:
  - in a file named <ActionName-alias>-validation.xml
- Inheritance hierarchy and interfaces implemented by Action class
  - Searches up the inheritance tree of the action to find default validations for parent classes of the Action and interfaces implemented
- Using Java 5 annotations.

Built-in Validators

```
<validators>
        <validator name="required" class=".."/>
        <validator name="requiredstring" class=".."/>
        <validator name="int" class=".."/>
        <validator name="double" class=".."/>
        <validator name="date" class=".."/>
        <validator name="expression" class=".."/>
        <validator name="fieldexpression" class=".."/>
        <validator name="email" class=".."/>
        <validator name="url" class=".."/>
        <validator name="visitor" class=".."/>
        <validator name="conversion" class="../>
        <validator name="stringlength" class=".."/>
        <validator name="regex" class=".."/>
</validators>
```

Int validator

- The int Field Validator checks whether the given integer is within a certain range or not.
- parameters:
  - fieldName: The field name to be validated. Required if using Plain-Validator Syntax otherwise not required
  - min: the minimum value (if none is specified, it will not be checked).
  - max: the maximum value (if none is specified, it will not be checked).

Int validator

```
<validators>
    <field name="userinput">
      <field-validator type="int">
        <param name="min">10</param>
        <param name="max">80</param>
        <message>Number needs to between ${min} and ${
  max}
        </message>
      </field-validator>
    </field>
  </validators>
```

RequiredString validator

- RequiredStringValidator checks the String field is not-null and its length is > 0. (i.e. it isn't "").
- parameters:
  - fieldName: the field name to be validated. Required if using Plain-Validator Syntax otherwise not required.
  - trim: the field name of value before validating (default is true).

RequiredString validator

```
<validators>
    <field name="username">
      <field-validator type="requiredstring">
        <param name="trim">true</param>
        <message>User Name is required</message>
      </field-validator>
    </field>
</validators>
```

Double Validator

- This Field Validator checks, if the given number is double and specified within the specified range. If your input text is valid or within specified range then success. Otherwise, display error message.
- parameters:
  - fieldName: the field name to be validated.
  - minInclusive: the minimum inclusive value as FloatValue format specified by Java language (if none is specified, it will not be checked)

Double Validator

- maxInclusive - This is the maximum inclusive value as FloatValue format specified by Java language (if none is specified, it will not be checked)
- minExclusive - This is the minimum exclusive value as FloatValue format specified by Java language (if none is specified, it will not be checked)
- maxExclusive - This is the maximum exclusive value as FloatValue format specified by Java language (if none is specified, it will not be checked)

Double Validator

```
<validators>
    <field name="percentagemarks">
        <field-validator type="double">
          <param name="minInclusive">20.1</param>
          <param name="maxInclusive">50.1</param>
          <message>Percentage marks need to between
    ${minInclusive} and ${maxInclusive}</message>
        </field-validator>
      </field>
</validators>
```

Date Validator

- The Date validator checks whether the supplied date lies within a specific range or not. If the value supplied does not lie in the specified range, it generates an error message.
- parameters:
  - fieldName: the field name of the validator.
  - min: the min date range. If not specified, it will not be checked.
  - max: the max date range. If not specified, it will not be checked.

Date Validator

```
<validators>
      <field name="joiningdate">
        <field-validator type="date">
          <param name="min">01/01/1990</param>
          <param name="max">01/01/2000</param>
          <message>Joining date must be supplied between ${min} and $
    {max}</message>
        </field-validator>
      </field>
</validators>
```

Email validator

- The email validator checks whether a given String field is empty or not and contains a valid email address or not. If the entered value does not match with the email type then the e-mail validator generates an error message.
- parameters:
  - fieldName: the field name to be validated.

Email validator

```
<validators>
      <field name="myEmail">
        <field-validator type="email">
           <message>Please enter a valid email</message>
        </field-validator>
      </field>
</validators>
```

URLValidator

- The URLValidator checks whether the String contained within the given field is a valid URL or not. If the entered value is not a valid URL, it generates an error message.
- parameters:
  - fieldName: the field name to be validated.

URLValidator

```
<validators>
      <field name="url">
        <field-validator type="url">
           <message>Please enter a valid url</message>
        </field-validator>
      </field>
</validators>
```