Jakarta Struts: Automatically Validating Input

Struts 1.2 Version

Core Servlets & JSP book: www.coreservlets.com
More Servlets & JSP book: www.moreservlets.com
Servlet/JSP/Struts/JSF Training: courses.coreservlets.com

For live Struts training, please see JSP/servlet/Struts/JSF training courses at http://courses.coreservlets.com/.

Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization.

Overview

- **Distinguishing manual validation from automatic validation**
- **Distinguishing client-side validation from server-side validation**
- **Using automatic validation**
  - Declare application-wide properties file
  - Add messages to properties file
  - Turn on the automatic validator
  - Put validation rules in validation.xml
  - Put <html:errors/> in input page
  - Enable JavaScript validation

Options for Form Field Validation

- **Do validation in the Action**
  - Most powerful; has access to business logic, DB, etc.
  - May require repetition in multiple Actions
  - Must manually map conditions back to input page
  - Must write validation rules yourself
- **Do validation in the form bean**
  - In individual setter methods
    - Not really validation, but can be used to modify values
  - Using the validate method
    - Not quite as powerful
    - Does not require repetition in multiple Actions
    - Will automatically redisplay input page
    - Still requires you to write validation rules yourself
- **Use automatic validator**
  - Handles many common cases; includes JavaScript
  - You can combine approaches in the same application

Manual Validation
(See Previous Section for Details and Examples)

- **Option 1: Put validation code in the Action**
  - Return custom conditions from Action
  - Map certain conditions back to the input form
  - Embed the messages in the form bean
- **Option 2: Put validation code in bean**
  - Insert <html:errors/> in input form
  - Use validate method in ActionForm class

```
public ActionErrors validate(ActionMapping mapping,
                             HttpServletRequest request) {
  ActionErrors errors = new ActionErrors();
  if (somethingWrongWith(someField)){
    errors.add("someField",
               new ActionMessage("errors.someField"));
  }
  return errors;
}
```

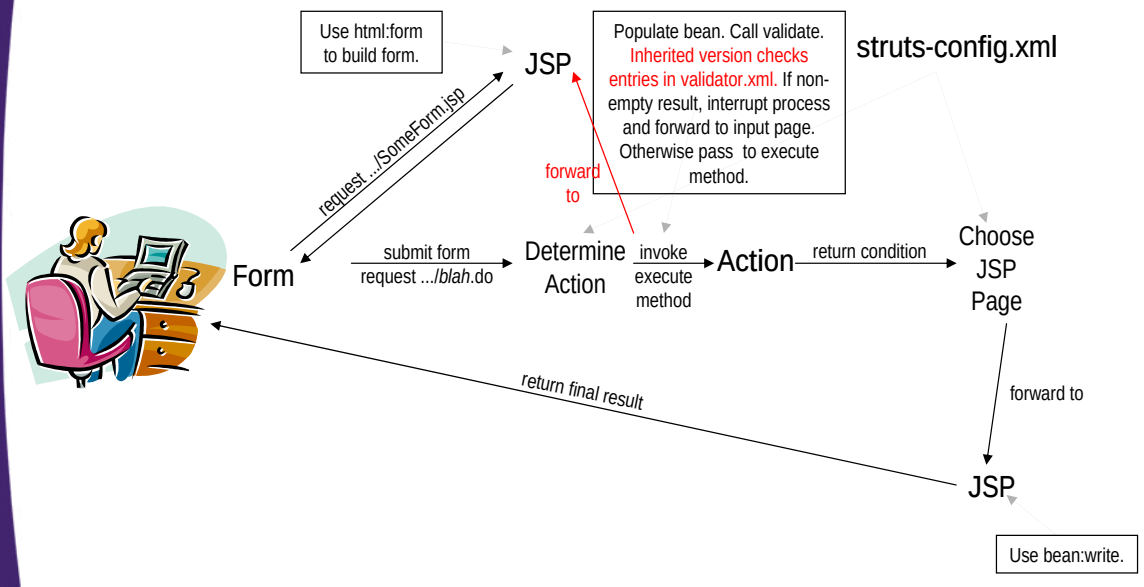

Manual vs. Automatic Validation

- **Manual validation**
  - Most flexible
  - Has full access to bean and to business logic and database
  - Repeats same logic many times
  - Runs only on server if you use existing framework
    - Client-side validation requires writing lots of JavaScript
  - Tedious
  - Embedded in Java code
    - Which violates Struts strategy of having as much as possible in editable XML files
- **Automatic validation**
  - Consolidates validation code
  - Lets you use standard validation rules
  - Runs on server; can optionally also run on client
  - Described by XML files

Client-Side vs. Server-Side Validation

- **Client-side validation**
  - JavaScript code verifies format of fields
  - Dialog box warns users of illegal values
  - Submission blocked if invalid
  - Pro:
    - Fast
  - Cons:
    - Can be deliberately or accidentally bypassed
    - Cannot do validation that requires much application logic
- **Server-side validation**
  - Java code on server verifies format of fields
  - Form is redisplayed (with warnings) if illegal values
  - You *must* do this regardless of whether or not you do client-side validation!

Struts Flow of Control



Steps in Using Automatic Validation (General)

1. **Configure struts-config.xml**
   - List the address of the input form
   - List the properties file (resource bundle)
   - Turn on the automatic validator
2. **Edit the properties file**
   - Put errors.footer, errors.header for html:errors as before
   - Edit standard validator messages (errors.invalid, etc)
   - Create names to replace {0}, {1} in standard messages
3. **Put validation rules in validation.xml**
   - For each field, specify one or more validation rules
   - Find the name of the corresponding error message
   - Look in properties file to see how many args needed
   - Supply arg0 ... argN as necessary
4. **Have form bean extend ValidatorForm**
   - Instead of ActionForm
5. **Use <html:errors> as before**

Steps in Using Automatic Validation (Details on Third Step)

1. **For each field, specify one or more validation rules from the list of builtin choices**
   - required, mask, email, intRange, maxLength, etc.
2. **Find the name of the error message that will be generated if the rule fails**
   - Usually errors.*ruleName*, but given in validator-rules.xml.
3. **Look in properties file to see what {} arguments the error message needs**
   - errors.invalid={0} is invalid.
   - errors.maxlength={0} cannot be greater than {1} characters.
4. **Supply arg0-argN for each placeholder**

Using Automatic Validation

1. Configure struts-config.xml

1. List the address of the input form

```
<action path="..." type="..." name="..." scope="request"
        input="...inputFormAddress.jsp">
```

2. List properties file (resource bundle).
   - `<message-resources parameter="MessageResources"/>`
     - Refers to WEB-INF/classes/MessageResources.properties
3. Turn on the automatic validator
   - Don't enter by hand: uncomment the entry from struts-blank

```
<plug-in
    className="org.apache.struts.validator.ValidatorPlugIn">
  <set-property property="pathnames"
                value="/WEB-INF/validator-rules.xml,
                       /WEB-INF/validation.xml"/>
</plug-in>
```


Using Automatic Validation

2. Edit properties file

- Edit "error" entries for formatting error messages
  - Same as in manual validation

```
errors.header=<UL>
errors.prefix=<LI><B><FONT COLOR="RED">
errors.suffix=</FONT></B></LI>
errors.footer=</UL>
```

- Edit standard "validator" error messages if desired
  - struts-blank has several typos you will want to fix
    - "an" long, "an" byte, "can not", etc.

```
errors.invalid={0} is invalid.
errors.maxlength={0} cannot be greater than {1} characters.
```

- Add prompts/messages that will be substituted into error messages for {0}, {1}. etc

```
inputForm.firstName=First name
inputForm.lastName=Last name
inputForm.zipCode=5-digit ZIP Code
```

Using Automatic Validation (Continued)

3. **Put validation rules in validation.xml**
   - Use <form name="..."> to identify the bean
     - <form name="*beanNameFromStrutsConfig*">
   - Use <field property="..." depends="..."> to identify the bean property to check and the rule to use to check it
     - <field property="*propName*" depends="*ruleName*">
     - See http://struts.apache.org/userGuide/dev_validator.html for all available rules and specifics on each
   - Use <arg*N*...> to give values for error messages.
     - <field property="*propName*" depends="*ruleName*">
     - <arg0 key="*key.Name*"/>
     - </field>
   - Look in properties file to see what args needed
   - The name of the error message is usually errors.*ruleName*, but see validator-rules.xml to be sure

validation.xml: Structure

- **<form-validation> and <formset>**
  - Main enclosing elements
- **<form name="*beanName*">**
  - Matches form-bean name from struts-config.xml
- **<field property="firstName"**
  - Matches HTML form parameter (ie, bean property) name
- **depends="required">**
  - Matches name of predefined validator rule
    - **required**: must be non-empty
    - **mask**: must match a given regular expression
    - **email**: must be an email address
    - **creditCard**: must be a legal credit card number
      - (Use 4111111111111111 for testing)
- **<arg0 key="property.subname"/>**
  - Replaces {0} in error message from properties file

Using Automatic Validation (Continued)

3. **Put validation rules in validation.xml: example**
   - Bean is named orderFormBean in the form-beans section of struts-config.xml
   - Bean has property called firstName
   - One of the standard rules is "required"
   - In the properties file, errors.required is ... {0} ...
   - There is a custom message called inputForm.firstName

```
<formset>
  <form name="orderFormBean">
    <field property="firstName"
           depends="required">
      <arg0 key="inputForm.firstName"/>
    </field>
  </form>

…
</formset>
```


Using Automatic Validation (Continued)

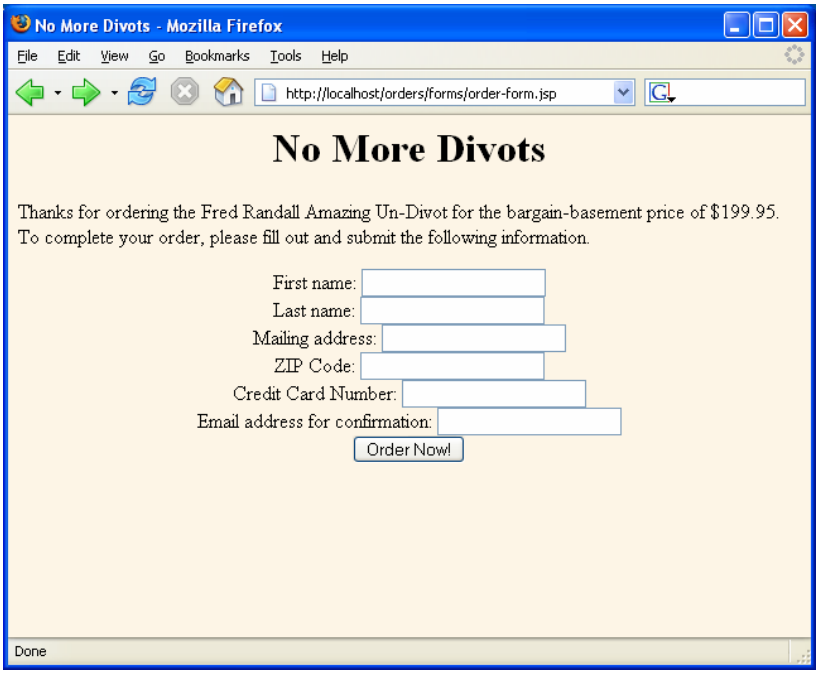
4. **Have your form bean extend ValidatorForm, not ActionForm directly**

```
import org.apache.struts.validator.*;

public class OrderFormBean
  extends ValidatorForm { … }
```

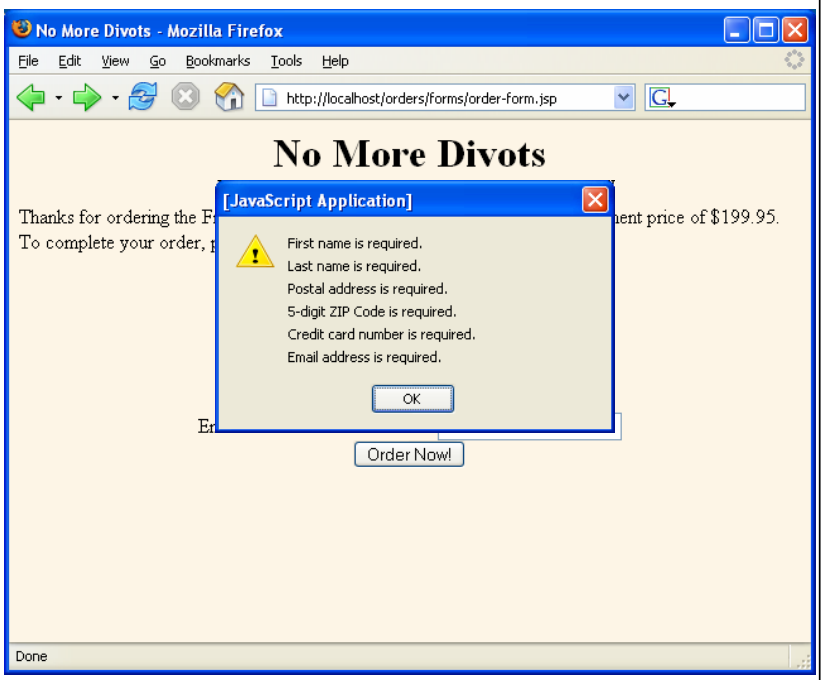
5. **Put <html:errors/> in input page**
   - Edit properties file to customize form of error message
6. **(Optional) Enable JavaScript validation**
   - Add <html:javascript formName="*beanName*"/> anywhere
   - Add onsubmit="return validate*BeanName*(this);" to html:form

Using Automatic Validation: Example (Goal)



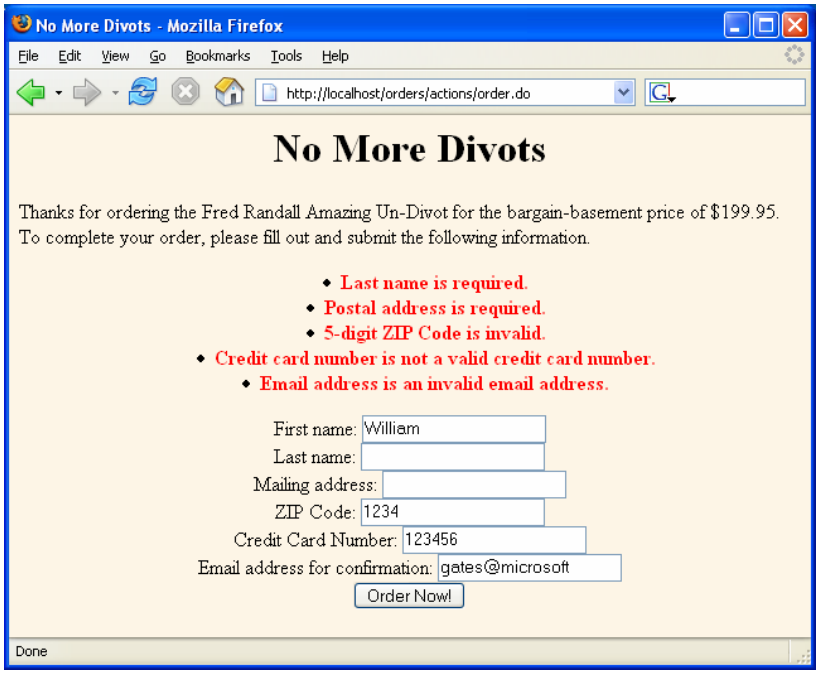
Using Automatic Validation: Example (Goal)

- **JavaScript validation enabled**

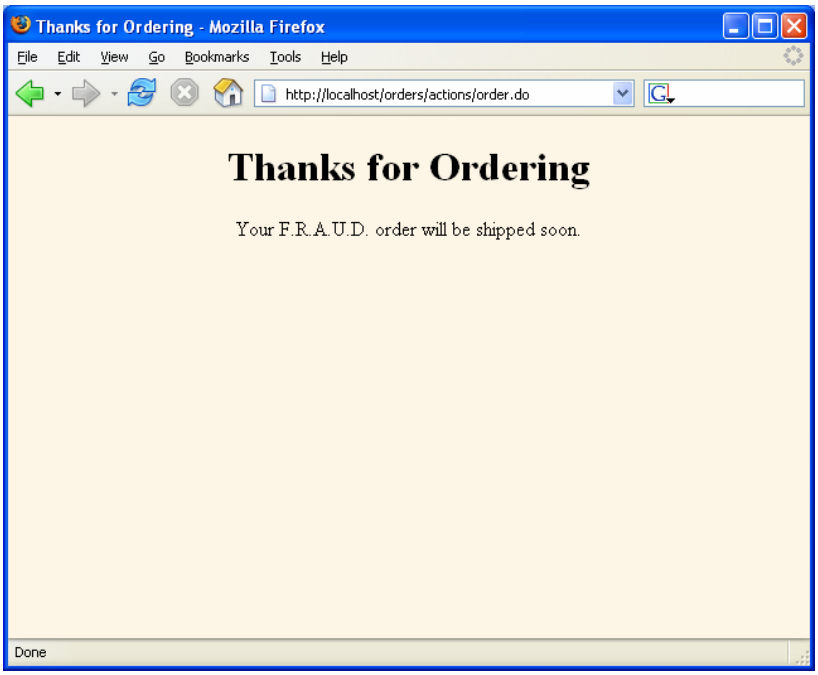


Using Automatic Validation: Example (Goal)

- **JavaScript validation disabled**



Using Automatic Validation: Example (Goal)



Implementing the Example (Background)

```
package coreservlets;

import javax.servlet.http.*;
import org.apache.struts.action.*;

public class Order extends Action {
  public ActionForward execute(ActionMapping mapping,
                               ActionForm form,
                               HttpServletRequest request,
                               HttpServletResponse response)
      throws Exception {
    return(mapping.findForward("success"));
  }
}
```


Step 1: Configure struts-config.xml

```
<form-beans>
  <form-bean name="orderFormBean"
             type="coreservlets.OrderFormBean"/>
</form-beans>
<action-mappings>
  <action path="/actions/order"
          type="coreservlets.Order"
          name="orderFormBean"
          scope="request"
          input="/forms/order-form.jsp">
    <forward name="success"
             path="/WEB-INF/results/order-confirmation.jsp"/>
  </action>
</action-mappings>

<message-resources parameter="MessageResources"/>

<plug-in
   className="org.apache.struts.validator.ValidatorPlugIn">
  <set-property
    property="pathnames"
    value="/WEB-INF/validator-rules.xml,
           /WEB-INF/validation.xml"/>
</plug-in>
```

Step 2: Edit Properties File (MessageResources.properties)

```
# -- Custom messages for this application --
inputForm.firstName=First name
inputForm.lastName=Last name
inputForm.address=Postal address
inputForm.zipCode=5-digit ZIP Code
inputForm.creditCardNumber=Credit card number
inputForm.email=Email address

# -- Standard errors --
errors.header=<UL>
errors.prefix=<LI><B><FONT COLOR="RED">
errors.suffix=</FONT></B></LI>
errors.footer=</UL>

# -- validator --
errors.invalid={0} is invalid.
errors.maxlength={0} cannot be greater than {1} characters.
errors.minlength={0} cannot be less than {1} characters.
errors.range={0} is not in the range {1} through {2}.
errors.required={0} is required.
...
```


Step 3: Put Validation Rules in validation.xml

```
  <formset>
    <form name="orderFormBean">
      <field property="firstName"
             depends="required">
        <arg0 key="inputForm.firstName"/>
      </field>
      …
      <field property="zipCode"
             depends="required,mask">
        <arg0 key="inputForm.zipCode"/>
        <var>
          <var-name>mask</var-name>
          <var-value>^\d{5}$</var-value>
        </var>
      </field>
      <field property="creditCardNumber"
             depends="required,creditCard">
        <arg0 key="inputForm.creditCardNumber"/>
      </field>
… </form></formset>
```

Step 4: Have Form Bean Extend ValidatorForm

```
package coreservlets;

import org.apache.struts.validator.*;

public class OrderFormBean extends ValidatorForm {
  private String firstName = "";
  …
  public String getFirstName() {
    return(firstName);
  }

  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }
  …
}
```


Steps 5 & 6: Output <html:errors/> and Enable JavaScript validation

```
…
<%@ taglib uri="http://struts.apache.org/tags-html"
           prefix="html" %>
<html:errors/>
<html:form action="/actions/order"
           onsubmit="return validateOrderFormBean(this);">
  First name: <html:text property="firstName"/><BR>
  Last name: <html:text property="lastName"/><BR>
  Mailing address: <html:text property="address"/><BR>
  ZIP Code: <html:text property="zipCode"/><BR>
  Credit Card Number:
  <html:text property="creditCardNumber"/><BR>
  Email address for confirmation:
  <html:text property="email"/><BR>
  <html:submit value="Order Now!"/>
</html:form>
<html:javascript formName="orderFormBean"/>
…
```

Other Validator Capabilities

- **More predefined validators**
  - required
  - mask
    - Note that the associated message property name is errors.invalid, not errors.mask. See the "msg" entry in validator-rules.xml
  - intRange, floatRange, doubleRange (note uppercase "R")
    - Note that associated error message is "range"
  - maxlength, minlength (note lowercase "l")
  - integer, float, double, long, short, byte
  - date
  - creditCard
  - email
  - url
- **Specific guidance on using these**
  - http://struts.apache.org/userGuide/dev_validator.html

Combining Manual and Automatic Validation

- **Automatic validators do generic checks**
  - Email address, URL, credit card, missing, etc.
- **The validate method does application-specific checks**
  - That selected health plan was one of available options, etc
- **The validate method must call super.validate for this to work**

```
public ActionErrors validate
                (ActionMapping mapping,
                 HttpServletRequest request) {
  ActionErrors errors =
    super.validate(mapping, request);
  ...
  return(errors);
}
```

Rolling Your own Pluggable Validators

- **Builtin rules not sufficient**
  - For example, one field may depend on another
- **You can assign both server-side and client-side code**
- **Declare validators in validator-rules.xml**
- **Very powerful**
  - Can be easily reused throughout your application
- **Not easy!**
  - Complicated
  - Poorly documented

Summary

- **Modify struts-config.xml**
  - List the input form
  - Load the properties file (resource bundle)
  - Enable validator
- **Update the properties file**
  - Edit standard error messages
  - List formatting rules for error messages
  - Add custom messages for substitution into error messages
- **Put validation rules in validation.xml**
- **Extend ValidatorForm instead of ActionForm**
- **Put <html:errors/> in input page**
- **Enable JavaScript validation if desired**
  - <html:javascript formName="*beanName*"/>

Questions?

Core **Servlets & JSP book: www.coreservlets.com**
More **Servlets & JSP book: www.moreservlets.com**
Servlet and JSP Training Courses: courses.coreservlets.com