Jakarta Struts: Manually 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

Slides © Marty Hall, http://www.coreservlets.com, books © Sun Microsystems Press

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.

Slides © Marty Hall, http://www.coreservlets.com, books © Sun Microsystems Press

Overview

- **Distinguishing manual validation from automatic validation**
- **Performing validation in the Action**
  - Error messages in beans
- **Performing validation in the ActionForm**
  - Fixed error messages
  - Error messages with substitution
  - Separate error messages

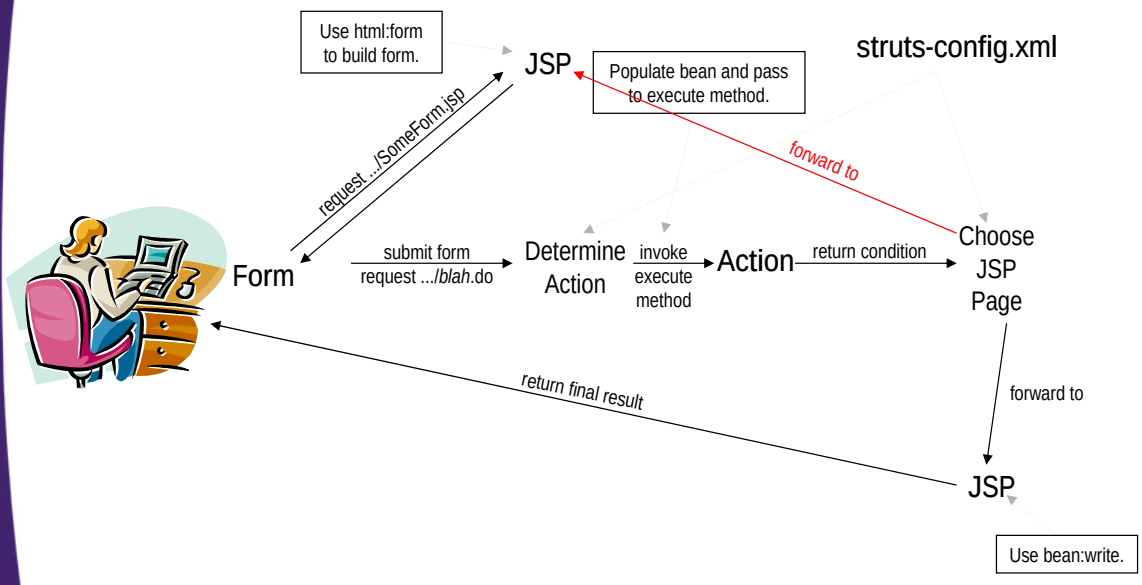
Options for Form Field Validation

- **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
  - See next lecture

Performing Validation in the Action's execute Method

Slides © Marty Hall, http://www.coreservlets.com, books © Sun Microsystems Press

Struts Flow of Control



Performing Validation in the Action

- **Start normally**
  - Cast ActionForm to specific type
  - Call getter methods to retrieve field values
- **For each missing or incorrect value**
  - Add an error message to a bean
    - Using the ActionForm bean is easiest
  - Use mapping.findForward to return error code
- **Use struts-config.xml to map the error code back to the input form**
- **Use bean:write to output error messages in input form**
  - Use filter="false" if error messages contain HTML tags

Example: Choosing Colors and Font Sizes for Resume

- **Input form**
  - Collects three font sizes (title, heading, body)
  - Collects two colors (foreground, background)
  - Uses bean:write to print out error messages
    - Error messages are empty strings by default
    - Remember the taglib entry for the "bean" library
- **ActionForm**
  - Represents form data
  - No error checking
  - Contains extra field for storing error messages
- **Action**
  - execute method checks if any params are missing. If so:
    - Returns error code
    - Puts warning message in form bean
- **struts-config.xml**
  - Maps error code back to input form

Example: Input Form

```
<CENTER>
<%@ taglib uri="http://struts.apache.org/tags-html"
           prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean"
           prefix="bean" %>
<html:form action="/actions/showSample">
  <bean:write name="formatFormBean"
              property="warning" filter="false"/><BR>
  Title size: <html:text property="titleSize"/><BR>
  Heading size: <html:text property="headingSize"/><BR>
  Body text size: <html:text property="bodySize"/><BR>
  Background color: <html:text property="bgColor"/><BR>
  Foreground color: <html:text property="fgColor"/><BR>
  <html:submit value="Show Sample"/>
</html:form>
</CENTER>
```


Example: ActionForm

```
package coreservlets;
import org.apache.struts.action.*;

public class FormatFormBean extends ActionForm {
  private String titleSize = "";
  private String headingSize = "";
  private String bodySize = "";
  private String bgColor = "";
  private String fgColor = "";
  private String warning = "";

  public String getTitleSize() {
    return(titleSize);
  }

  public void setTitleSize(String titleSize) {
    this.titleSize = titleSize;
  }
  ...
```

Example: ActionForm (Continued)

```
public String getStyleSheet() {
  return(
    "<STYLE TYPE=\"text/css\">\n" +
    "<!--\n" +
    "H1 { font-size: " + titleSize + "px; }\n" +
    "H2 { font-size: " + headingSize + "px; }\n" +
    "BODY { font-size: " + bodySize + "px;\n" +
    "       background-color: " + bgColor + ";\n" +
    "       color: " + fgColor + "; }\n" +
    "-->\n" +
    "</STYLE>");
  }
```


Example: ActionForm (Continued)

```
public String getWarning() {
  return(warning);
}

public void addWarning(String warning) {
  this.warning =
    this.warning + "<B><FONT COLOR=RED>" +
    warning + "!</FONT></B><BR>";
}

public boolean isMissing(String value) {
  return((value == null)||(value.trim().equals("")));
}
```

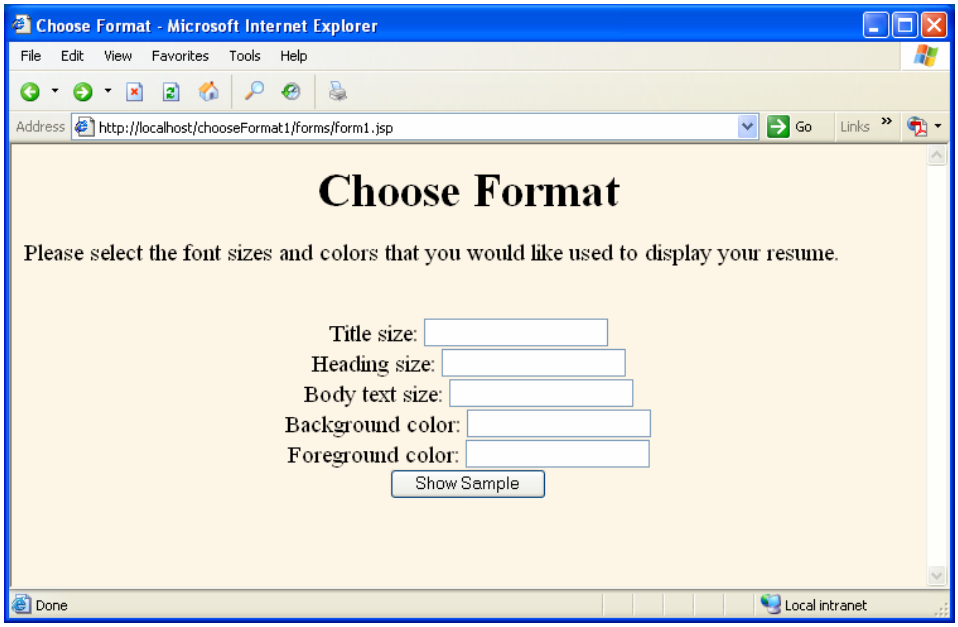
Example: Action

```
public class ShowSampleAction extends Action {
  public ActionForward execute(ActionMapping mapping,
                               ActionForm form,
                               HttpServletRequest request,
                               HttpServletResponse response)
      throws Exception {
    FormatFormBean formatBean = (FormatFormBean)form;
    ActionForward forward = mapping.findForward("success");
    if (formatBean.isMissing(formatBean.getTitleSize())) {
      formatBean.addWarning("Missing Title Size");
      forward = mapping.findForward("missing-data");
    }
    if (formatBean.isMissing(formatBean.getHeadingSize())) {
          formatBean.addWarning("Missing Heading Size");
          forward = mapping.findForward("missing-data");
    }
    …
    return(forward);
  }

}
```

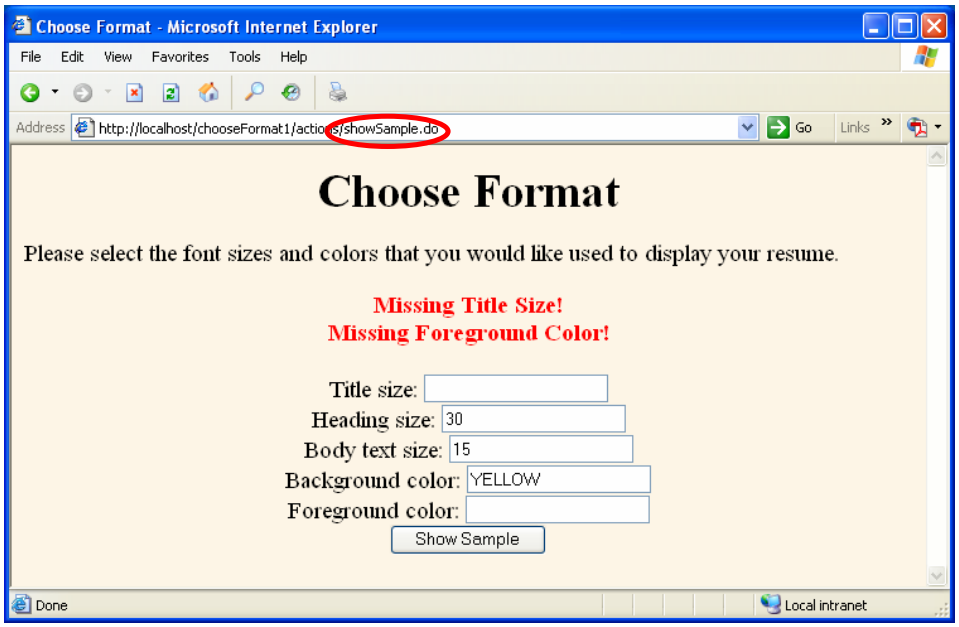

Example: struts-config.xml

```
<struts-config>
  <form-beans>
    <form-bean name="formatFormBean"
               type="coreservlets.FormatFormBean"/>
  </form-beans>
  <action-mappings>
    <action path="/actions/showSample"
            type="coreservlets.ShowSampleAction"
            name="formatFormBean"
            scope="request">
      <forward name="success"
               path="/WEB-INF/results/sample.jsp"/>
      <forward name="missing-data"
               path="/forms/form1.jsp"/>
    </action>
  </action-mappings>
</struts-config>
```

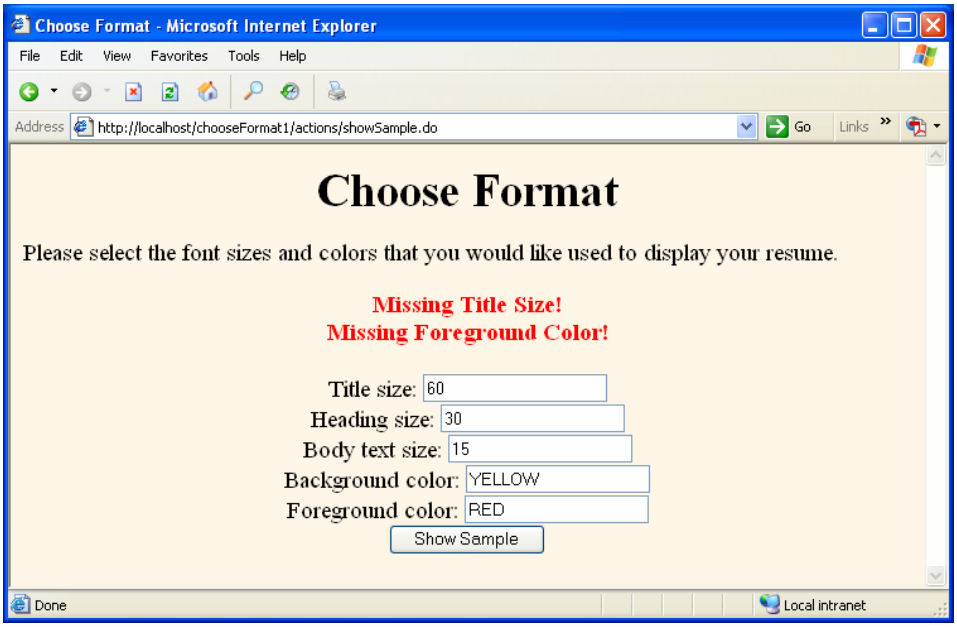
Result: Initial Form



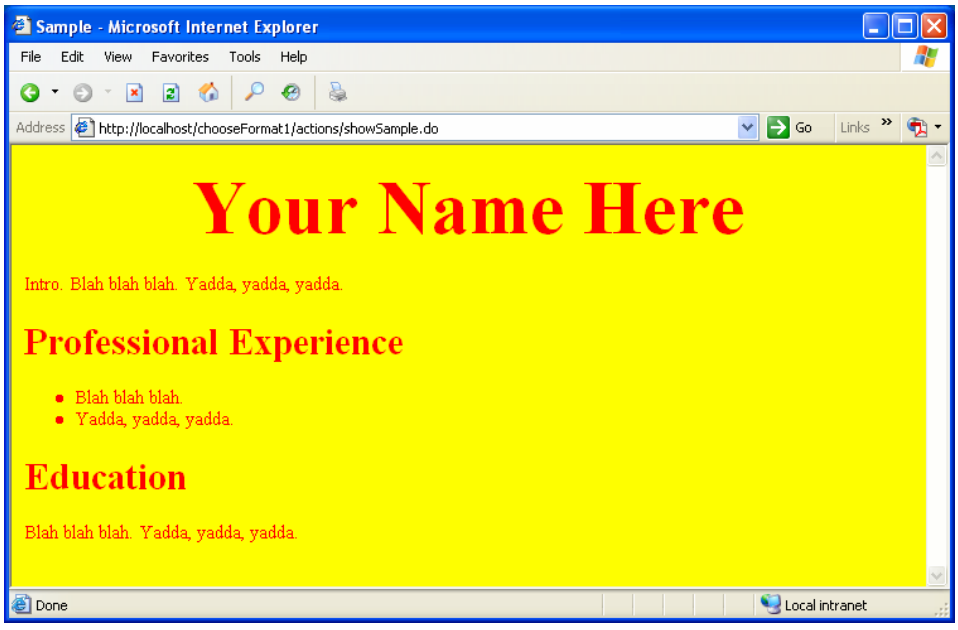
Result: Incomplete Data



Result: Rentering Data



Result: Complete Data



Variation: Redirecting Instead of Forwarding

- **In the usual scenario, the address listed by <forward ... path="..."/> is accessed by RequestDispatcher.forward**

```
<action path="..." type="..." name="..."
        scope="request">
  <forward name="..." path="..."/>
```

  - User sees URL of Action, not of JSP page
  - Data can be request-scoped
- **By specifying redirect="true", you can tell the system to use response.sendRedirect instead**

```
<action path="..." type="..." name="..."
        scope="session">
  <forward name="..." path="..." redirect="true"/>
```

  - User sees URL of JSP page
  - Data must be session-scoped
  - You must clear out error messages each time

Redirecting to Input Page

- **Advantages**
  - User sees a familiar URL
    - The one that was entered when the form was originally accessed
  - Bookmarks work normally
    - Can bookmark before or after the form is redisplayed. With forwarding, you cannot bookmark the input form once it is redisplayed (since the URL is the .do address of the Action).
  - User can hit reload on input form
- **Disadvantages**
  - Requires your data to be session-scoped
    - Per-request data like error messages must be reset each time!
    - Session data is expensive in clustered servers
  - Requires you to add entries to struts-config.xml
  - Does not directly correspond to other validation schemes
    - Using the validate method of the ActionForm
    - Using the automatic validation framework

Redirecting to Input Page: Example

- **Input form**
  - Unchanged except for action address
- **Action**
  - Unchanged
- **ActionForm**
  - Added code to reset error messages each time

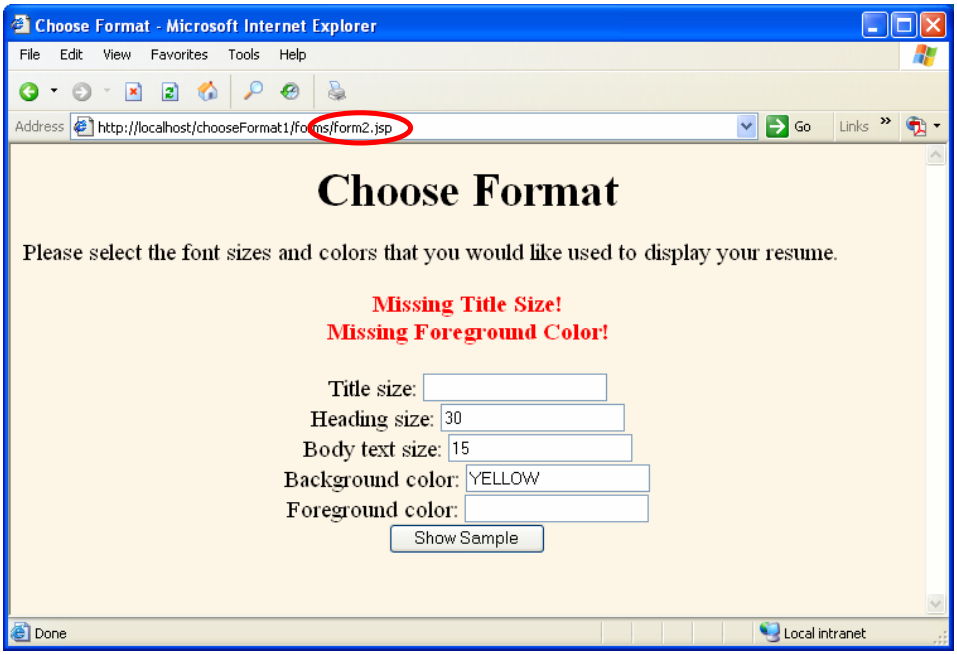
```
public class FormatFormBean extends ActionForm {
  ...
  public void resetWarning() {
    warning = "";
  }

  public void reset(ActionMapping mapping,
                    HttpServletRequest request) {
    resetWarning();
  }
}
```


Redirecting to Input Page: Example (Continued)

- **struts-config.xml**

```
<action path="/actions/showSample2"
        type="coreservlets.ShowSampleAction"
        name="formatFormBean"
        scope="session">
  <forward name="success"
           path="/WEB-INF/results/sample.jsp"/>
  <forward name="missing-data"
           path="/forms/form2.jsp"
           redirect="true"/>
</action>
```

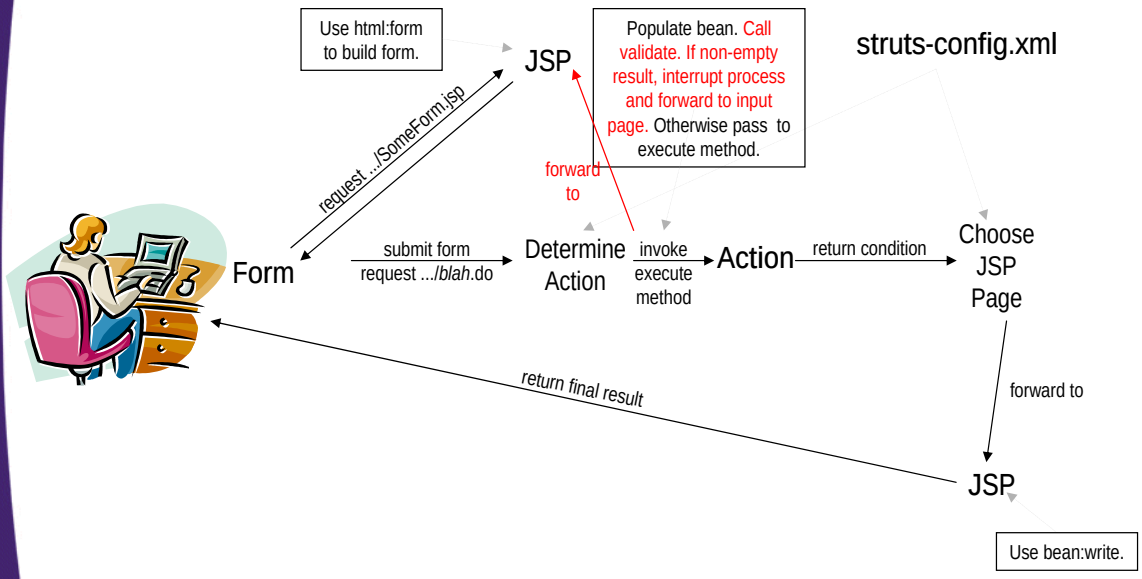
Redirecting to Input Page: Example (Results)



Performing Validation in the ActionForm's validate Method

Slides © Marty Hall, http://www.coreservlets.com, books © Sun Microsystems Press

Struts Flow of Control



Performing Validation in the ActionForm

- **Create an ActionForm method called validate**
  - If no errors, return null or an empty ActionErrors object
  - For each error, add ActionMessage entries to ActionErrors
    - ActionErrors.add takes a name and an ActionMessage
    - ActionMessage constructor takes key
      - Key corresponds to entry in a property file
      - Or, supply extra value of false to supply error message directly
  - If you return a non-empty ActionErrors object, the system will automatically forward user to the input form
    - Page listed by input attribute of action in struts-config.xml
- **Create a property file with error messages**
  - Property names should match keys used in ActionMessage
  - Also define how multiple error messages are output
  - Use struts-config to declare properties file
- **Use <html:errors/> in input form**
  - Prints list of all error messages. Empty if no errors.
- **No validation logic needed in Action**

The validate Method

```
public ActionErrors validate(ActionMapping mapping,
                             HttpServletRequest request) {
  ActionErrors errors = new ActionErrors();
  if (isSomeProblem(getSomeProperty())) {
    errors.add("someName",
               new ActionMessage("some.key"));
    errors.add("someOtherName",
              new ActionMessage("actual message", false));
  }
  ...
  return(errors);
}
```


Specifying Input Page in struts-config.xml

```
<action path="/somePath/someActionName"
        type="somePackage.SomeClass"
        name="someFormBean"
        scope="request"
        input="/somePath/original-form.jsp">
```

Preventing Validation

- The same bean might be used with multiple actions
- What if some actions want validation to occur, and others do not?
- Solution: specify validate="false" (in struts-config.xml) for actions that do not want validation
  - true is the default

```
<action path="..."
        type="somePackage.SomeClass"
        name="someFormBean"
        scope="request"
        input="/somePath/original-form.jsp"
        validate="false">
  <forward name="..." path"..."/>
</action>
```


Properties File

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

# -- Custom validation messages --
some.key=Some Message
some.other.key=Some Other Message
```

Example: Choosing Colors and Font Sizes for Resume (Take 2)

- **Input form**
  - Collects three font sizes (title, heading, body)
  - Collects two colors (foreground, background)
  - Uses <html:errors/> to print out error messages
    - Error message list is empty by default
- **ActionForm**
  - Represents form data
  - The validate method checks if any params are missing. If so:
    - Creates ActionMessage keyed to name from properties file
    - Adds ActionMessage to the ActionErrors that is returned
- **Action**
  - No validation code
- **struts-config.xml**
  - Lists path to input form

Example: ActionForm

```
package coreservlets;

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

public class FormatFormBean extends ActionForm {
  private String titleSize = "";
  private String headingSize = "";
  private String bodySize = "";
  private String bgColor = "";
  private String fgColor = "";

  public String getTitleSize() {
    return(titleSize);
  }

  public void setTitleSize(String titleSize) {
    this.titleSize = titleSize;
  }
  ...
```

Example: ActionForm (Continued: validate method)

```
public ActionErrors validate(ActionMapping mapping,
                             HttpServletRequest request) {
  ActionErrors errors = new ActionErrors();
  if (isMissing(getTitleSize())) {
    errors.add("title",
               new ActionMessage("titleSize.required"));
  }
  if (isMissing(getHeadingSize())) {
    errors.add("heading",
               new ActionMessage("headingSize.required"));
  }

  ...

  if (isMissing(getFgColor())) {
    errors.add("fg",
               new ActionMessage("fgColor.required"));
  } else if (getFgColor().equals(getBgColor())) {
    errors.add("fg",
               new ActionMessage("colors.notMatch"));
  }
  return(errors);
}
```


Example: Properties File (WEB-INF/classes/MessageResources.properties)

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

# -- Custom validation messages --
titleSize.required=Title size required.
headingSize.required=Heading size required.
bodySize.required=Body text size required.
bgColor.required=Background color required.
fgColor.required=Foreground color required.
colors.notMatch=Foreground and background
                colors must be different.
```

Example: struts-config.xml

```
<struts-config>
  <form-beans>
    <form-bean name="formatFormBean"
               type="coreservlets.FormatFormBean"/>
  </form-beans>
  <action-mappings>
    <action path="/actions/showSample"
            type="coreservlets.ShowSampleAction"
            name="formatFormBean"
            scope="request"
            input="/forms/index.jsp">
      <forward name="success"
               path="/WEB-INF/results/sample.jsp"/>
    </action>
  </action-mappings>
  <message-resources parameter="MessageResources"
                     null="false"/>
</struts-config>
```

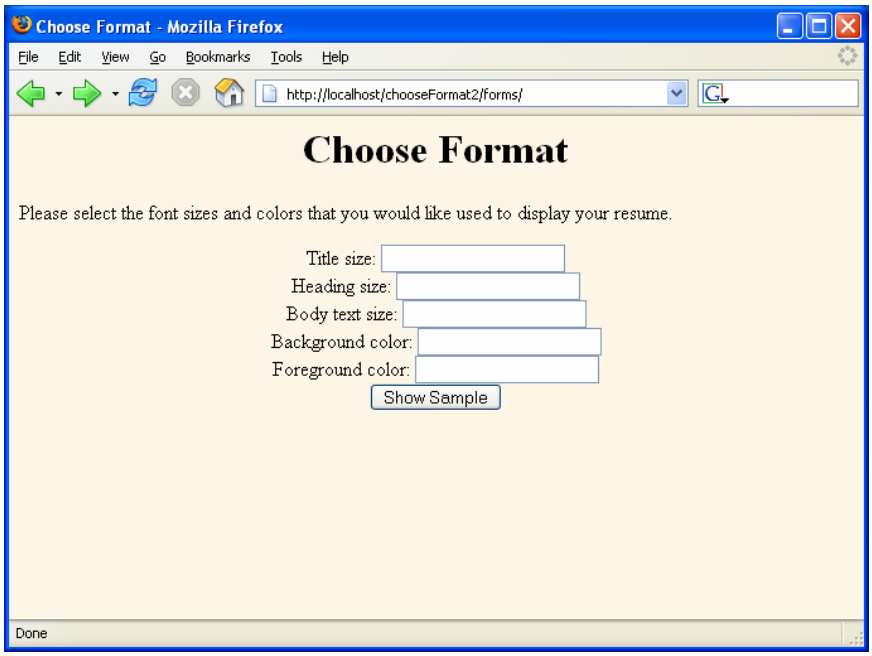

Example: Input Form

```
...
<CENTER>
<%@ taglib uri="http://struts.apache.org/tags-html"
           prefix="html" %>
<html:errors/>
<html:form action="/actions/showSample">
  Title size: <html:text property="titleSize"/><BR>
  Heading size: <html:text property="headingSize"/><BR>
  Body text size: <html:text property="bodySize"/><BR>
  Background color: <html:text property="bgColor"/><BR>
  Foreground color: <html:text property="fgColor"/><BR>
  <html:submit value="Show Sample"/>
</html:form>
</CENTER>
...
```

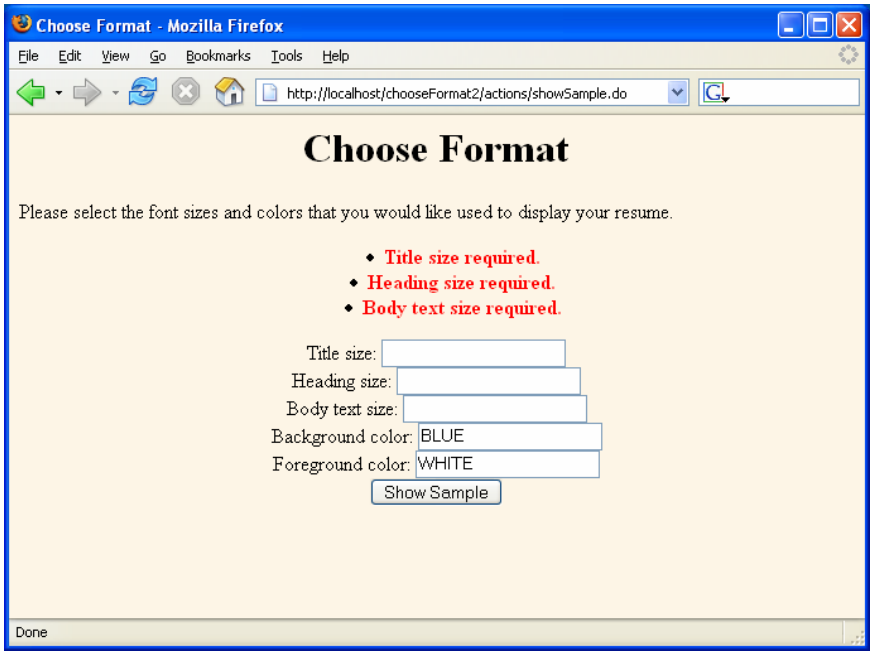
Example: Action

```
package coreservlets;

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

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

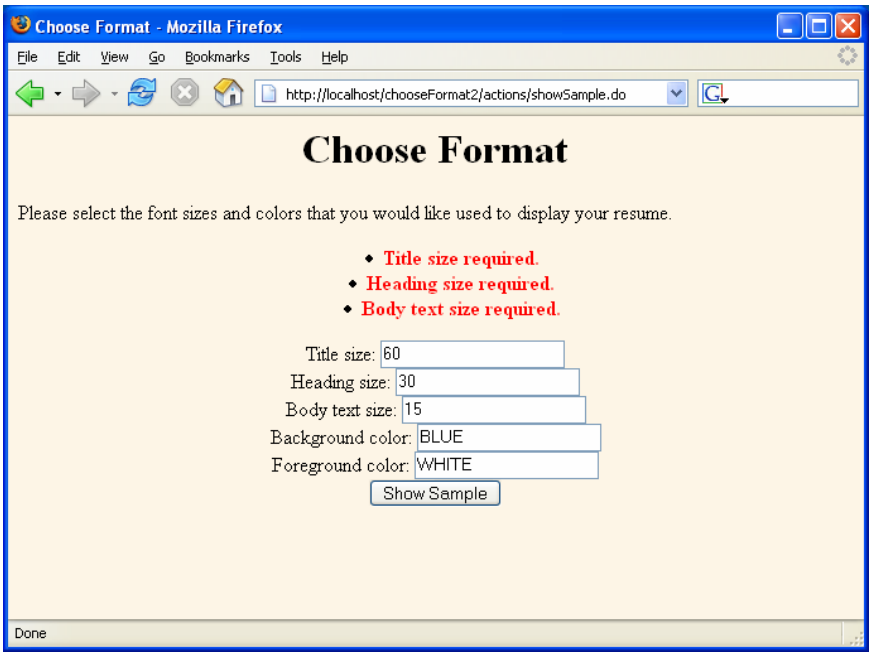

Result: Initial Form



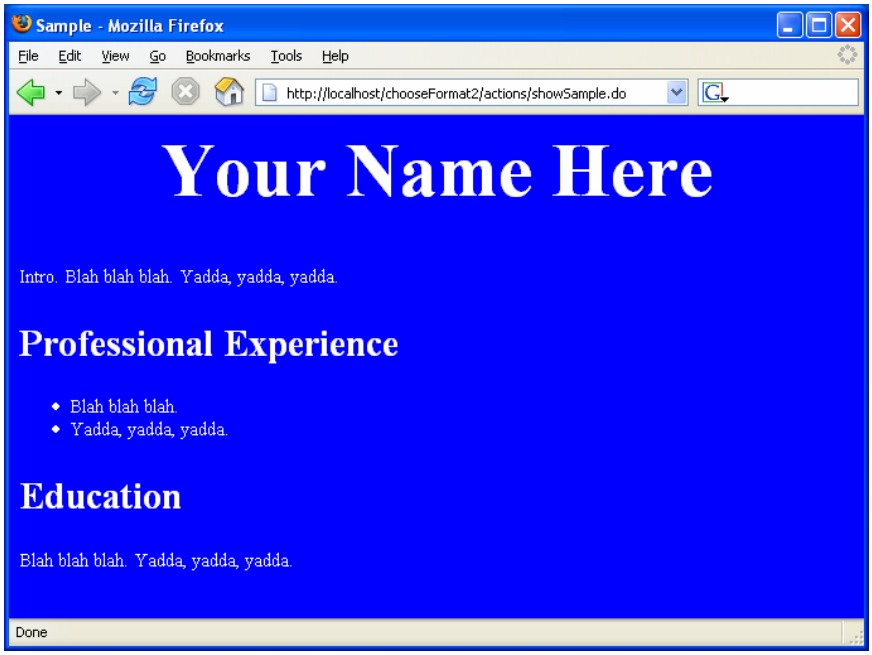
Result: Incomplete Data



Result: Rentering Data



Result: Complete Data



Using Parameterized Error Messages

- **Benefits**
  - Error messages reflect runtime values
  - Less repetition of error messages
  - More meaningful error messages for situations other than missing-data
- **Properties file**
  - Insert placeholders for values with {0}, {1}, etc.
  - E.g.: `value.required={0} is required.`
- **ActionForm**
  - Add extra arguments to ActionMessage constructor
    - One argument for each placeholder
    - Up to four separate arguments allowed
      - If more arguments needed, supply an array
  - Perform more complex validation (types of arguments, relationship among values, etc.)

Example: ActionForm (FormatFormBean)

```
...
  public ActionErrors validate(ActionMapping mapping,
                               HttpServletRequest request) {
    ActionErrors errors = new ActionErrors();
    if (isMissing(getTitleSize())) {
      errors.add("titleSizeMissing",
                 new ActionMessage("value.required",
                                   "Title size"));
    } else if (!isInt(getTitleSize())) {
      errors.add("titleNotInt",
                 new ActionMessage("value.int",
                                   "title size",
                                   getTitleSize()));
    }
    ...
    if (isMissing(getFgColor())) {
      errors.add("fgColorMissing",
                 new ActionMessage("value.required",
                                   "Foreground color"));
    } else if (getBgColor().equals(getFgColor())) {
      errors.add("colorsIdentical",
                 new ActionMessage("colors.match",
                                   getBgColor()));
    }
    return(errors);
  }
```


Example: ActionForm (Continued)

```
private boolean isMissing(String value) {
    return((value == null) || (value.trim().equals("")));
}

private boolean isInt(String potentialInt) {
  boolean isInt = true;
  try {
    int x = Integer.parseInt(potentialInt);
  } catch(NumberFormatException nfe) {
    isInt = false;
  }
  return(isInt);
}
```

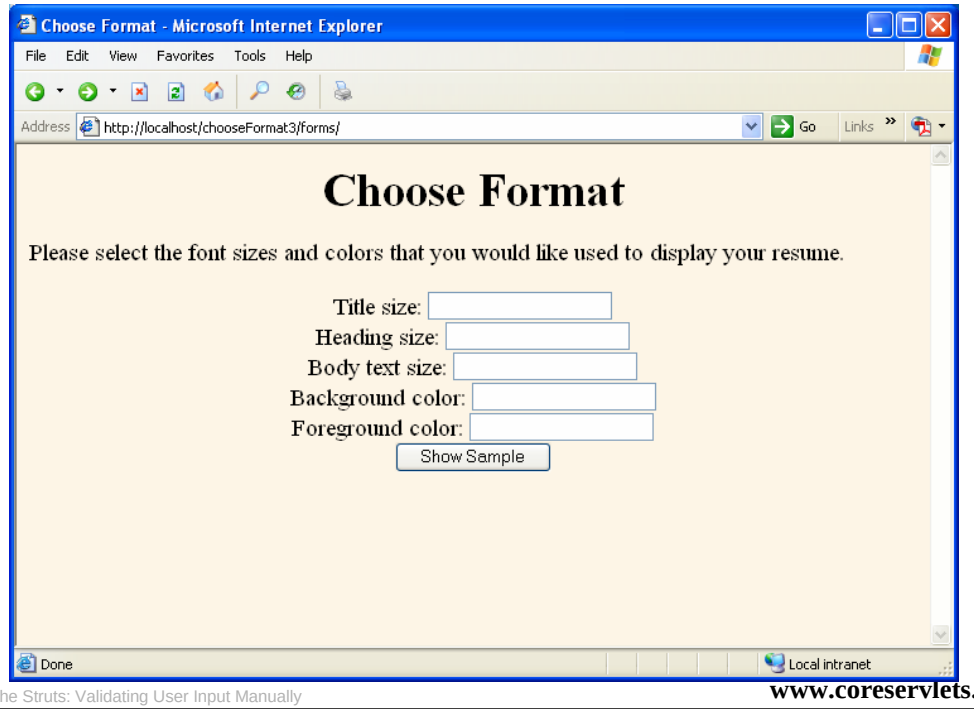
Example: Properties File
(WEB-INF/classes/MessageResources.properties)

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

# -- Custom validation messages --
value.required={0} is required.
value.int=Whole number required for {0}; "{1}"
          is not an integer.
colors.match=The foreground and background color
             are both "{0}".
```

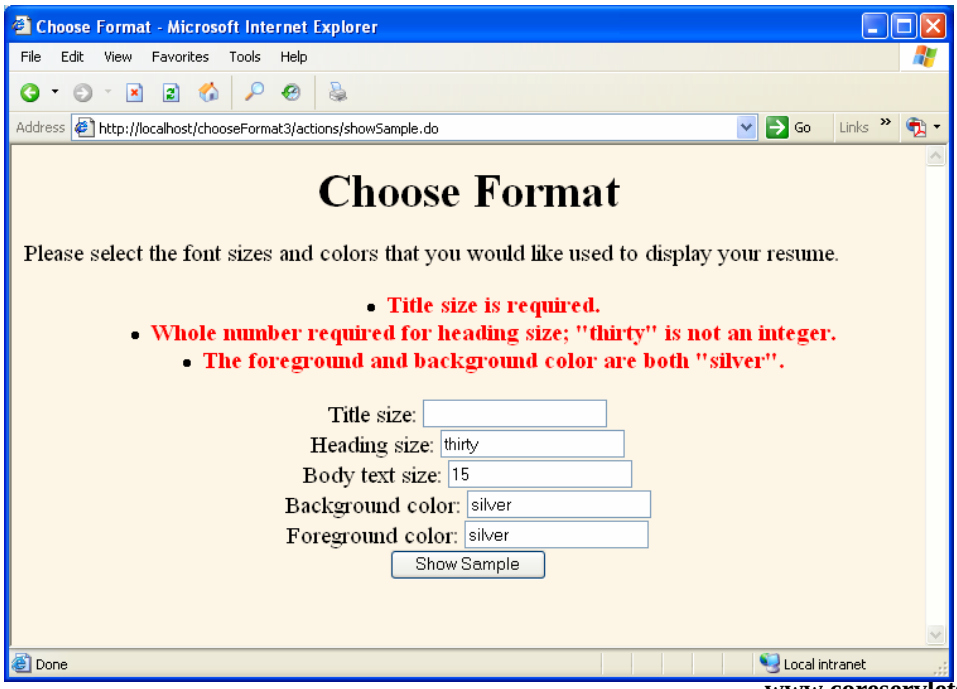

Unchanged Elements

- **Identical to previous example:**
  - Input form
    - Still simply uses <html:errors/>
  - Action
    - Still does not need validation logic or a special return value to indicate redisplaying the input form
  - struts-config.xml
    - Still lists the original form as the value of the input attribute of action

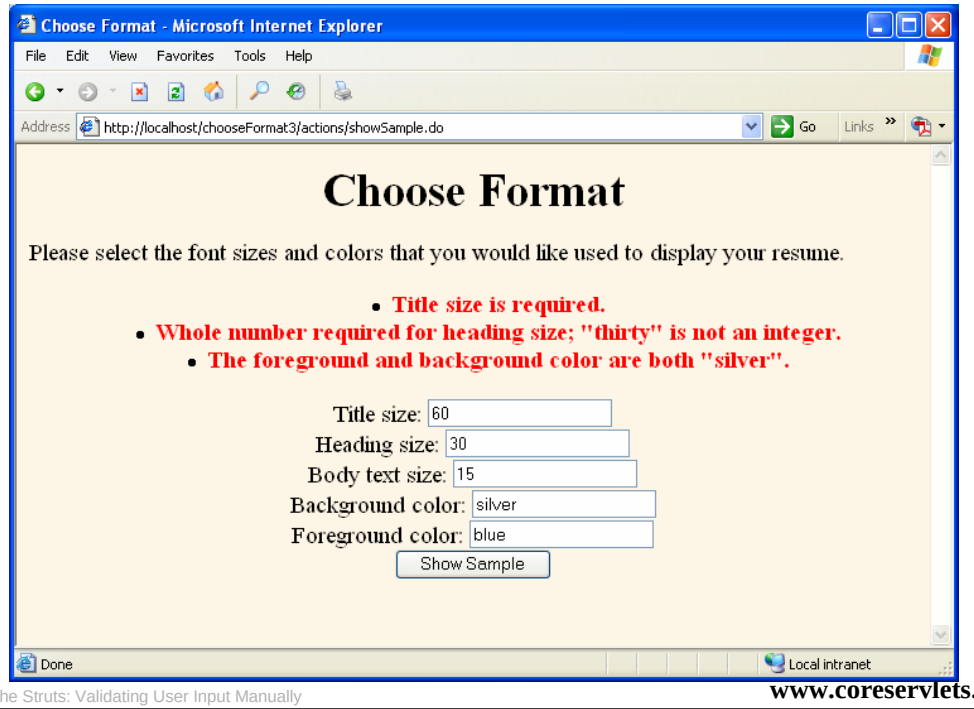
Result: Initial Form



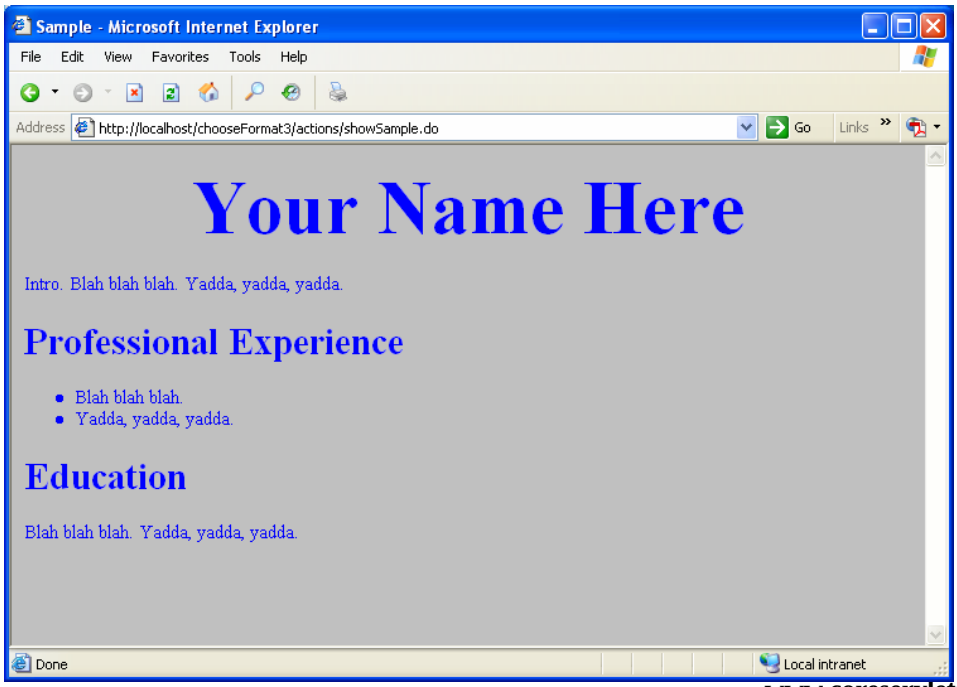
Result: Incomplete Data



Result: Rentering Data



Result: Complete Data



Displaying Separate Error Messages

- **Instead of a list of errors, you can display separate error messages for each field**
- **Use <html:errors property="name"/>**
- **The name should match the name given when ActionMessage was created**

- **Deficiency**
  - Not all warnings can be easily done with optional text
    - Turning the background color of table cells to red
    - Changing the text of a prompt

Example: Properties File (Shorter Messages)

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

# -- Custom validation messages --
value.required={0} required.
value.int=Need int for {0}; "{1}" invalid.
colors.match=FG and BG are both "{0}".
```

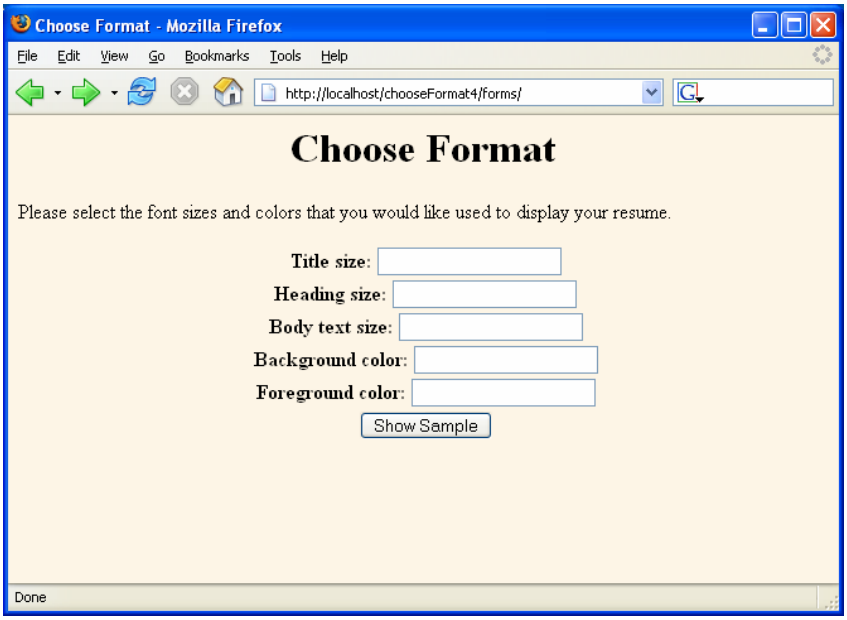
Example: ActionForm (FormatFormBean Unchanged)

```
...
  public ActionErrors validate(ActionMapping mapping,
                               HttpServletRequest request) {
    ActionErrors errors = new ActionErrors();
    if (isMissing(getTitleSize())) {
      errors.add("titleSizeMissing",
                 new ActionMessage("value.required",
                                   "Title size"));
    } else if (!isInt(getTitleSize())) {
      errors.add("titleNotInt",
                 new ActionMessage("value.int",
                                   "title size",
                                   getTitleSize()));
    }
    ...
    } else if (getBgColor().equals(getFgColor())) {
      errors.add("colorsIdentical",
                 new ActionMessage("colors.match",
                                   getBgColor()));
    }
    return(errors);
  }
```

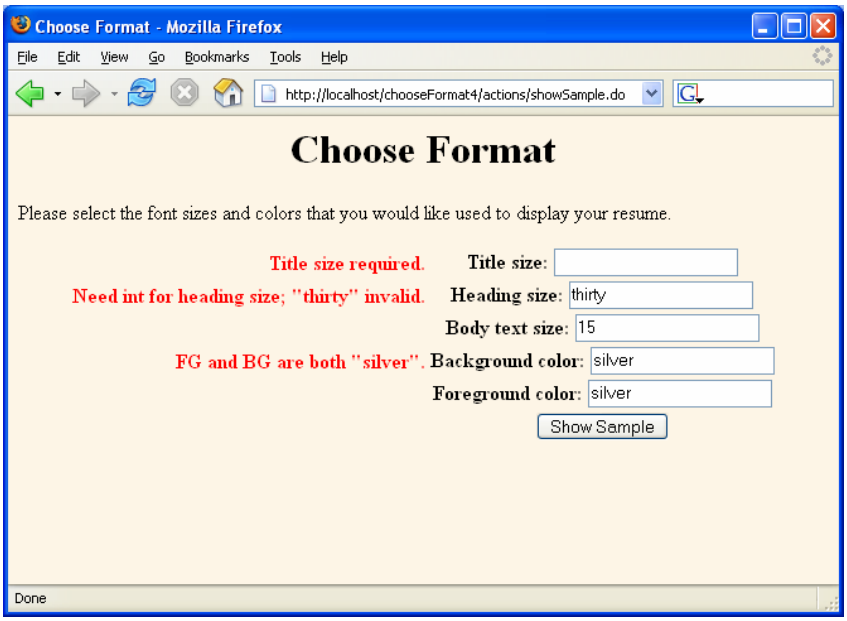

Example: Input Form

```
<html:form action="/actions/showSample">
<TABLE ALIGN="CENTER">
<TR><TH ALIGN="RIGHT">
      <html:errors property="titleSizeMissing"/>
      <html:errors property="titleNotInt"/>
    <TH>Title size: <html:text property="titleSize"/>
<TR><TH ALIGN="RIGHT">
      <html:errors property="headingSizeMissing"/>
      <html:errors property="headingNotInt"/>
    <TH>Heading size: <html:text property="headingSize"/>
<TR><TH ALIGN="RIGHT">
      <html:errors property="bodySizeMissing"/>
      <html:errors property="bodyNotInt"/>
    <TH>Body text size: <html:text property="bodySize"/>
<TR><TH ALIGN="RIGHT">
      <html:errors property="bgColorMissing"/>
      <html:errors property="colorsIdentical"/>
    <TH>Background color: <html:text property="bgColor"/>
<TR><TH ALIGN="RIGHT">
      <html:errors property="fgColorMissing"/>
    <TH>Foreground color: <html:text property="fgColor"/>
<TR><TH><TH><html:submit value="Show Sample"/>
</TABLE>
</html:form>
```

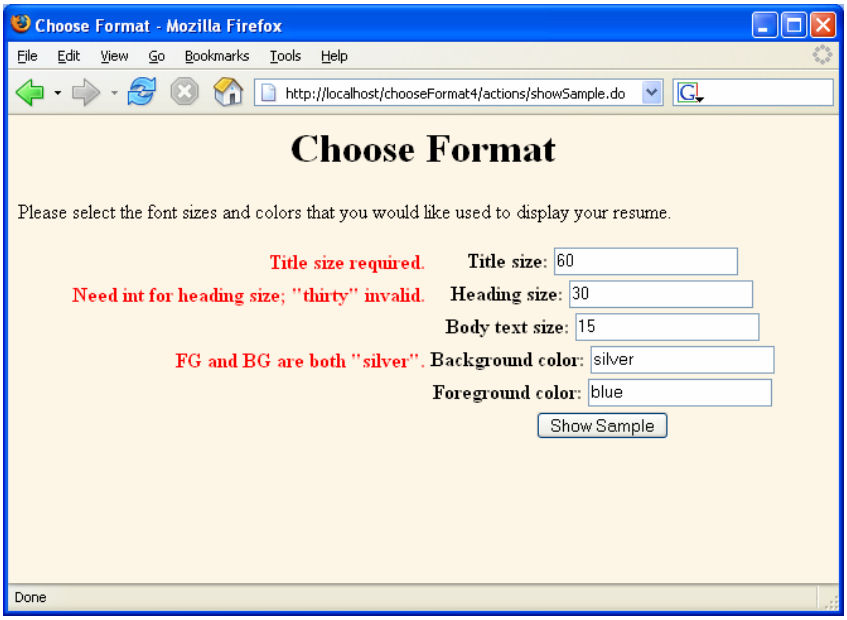
Result: Initial Form



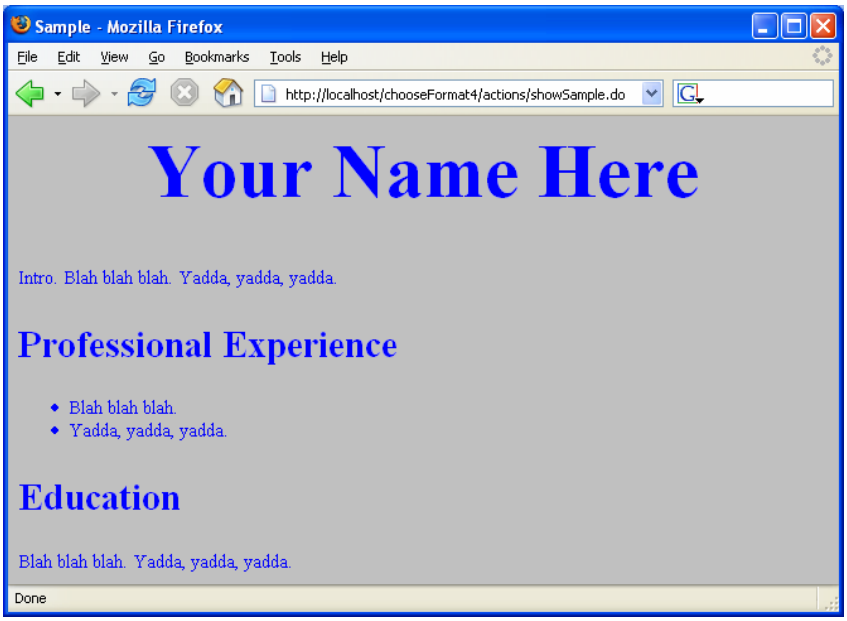
Result: Incomplete Data



Result: Rentering Data



Result: Complete Data



Summary

- **Perform app-specific validation in the Action**
- **Put more reusable validation in ActionForm**
  - Return non-empty ActionErrors from validate to trigger redisplay of input form
- **Error messages are given as keys**
- **Keys match entries in properties file**
  - Possibly with parameterized substitution
- **<html:errors/> prints out error messages**
  - Usually in a list; sometimes separately
  - Empty string output if no error messages
- **Preview: automatic validation framework**
  - Writing code to check for standard cases is tedious
  - What about JavaScript?

Questions?

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

Slides © Marty Hall, http://www.coreservlets.com, books © Sun Microsystems Press