How to build an XMLForm Wizard, Step Four

by Heidi Brannan

1. Step Four: HowtoWizardAction.java

This is based on the WizardAction.java. The prepare, perform and reset methods need to be altered. Your XML pages also need to be defined.

1.1. XML Pages defined

```
// different form views
// participating in the wizard
final String VIEW_START = "start";
final String VIEW_REGISTRATION = "registration";
final String VIEW_INTEREST = "interest";
final String VIEW_GARDENING = "organicGardening";
final String VIEW_COOKING = "cooking";
final String VIEW_SMALLHOLDING = "smallholdingManagement";
final String VIEW_CONFIRM = "confirm";
final String VIEW_END = "end";
```

1.2. Prepare Method

This method prepares the form and the pages to be returned before the actual form population starts.

The first time the URL http://localhost:8080/cocoon/mount/xmlform/howto-wizard.html is called there is no command passes so the if statement test is met and the start page is returned.

Copyright © 2002-2003 The Apache Software Foundation. All rights reserved.

After the start page has been viewed and the user clicks on "Start" the command start is passed so the else test is met and the registration page is returned. Any old forms are removed and a form listener is added to the form.

If neither of these tests are met then nothing is returned.

```
/**
 * The first callback method which is called
 * when an action is invoked.
 *
 * It is called before population.
 *
 *
 * @return null if the Action is prepared to continue.
 * an objectModel map which will be immediately returned by the action.
 *
 * This method is a good place to handle buttons with Cancel
 * kind of semantics. For example
 * <pre>if getCommand().equals("Cancel") return page("input");</pre>
 *
 */
protected Map prepare()
{

  if ( getCommand() == null )
    {
      return page( VIEW_START );
    }
  else   if ( getCommand().equals( CMD_START ) )
  {
    // reset state by removing old form
    // if one exists
    Form.remove( getObjectModel(), getFormId() );
    getForm().addFormListener( this );

    return page( VIEW_REGISTRATION );
  }

  // get ready for action
  // if not ready return page("whereNext");
  return null;
}
```

1.3. Perform Method

The perform method controls the logic of the forms pages to be displayed.

First the model is saved to the JavaBean

Copyright © 2002-2003 The Apache Software Foundation. All rights reserved.

```
/**
  * Invoked after form population
  *
  * Semanticly similar to Struts Action.perform()
  *
  * Take appropriate action based on the command
  *
  */
 public Map perform ()
 {

   // get the actual model which this Form encapsulates
   // and apply additional buziness logic to the model
   HowToBean  jBean = (HowToBean) getForm().getModel();

   // set the page control flow parameter
   // according to the validation result
   if ( getCommand().equals( CMD_NEXT ) &&
     getForm().getViolations () != null )
   {
     // errors, back to the same page
     return page( getFormView() );
   }
   else
   {
     // validation passed
     // continue with control flow

     // clear validation left overs in case the user
     // did not press the Next button
     getForm().clearViolations();

     // get the user submitted command (through a submit button)
     String command = getCommand();
     // get the form view which was submitted
     String formView = getFormView();

     // apply control flow rules
     if ( formView.equals ( VIEW_REGISTRATION ) )
     {
       if ( command.equals( CMD_NEXT ) )
       {
         return page(  VIEW_INTEREST );
       }
     }
     else if ( formView.equals ( VIEW_INTEREST ) )
     {
       if ( command.equals( CMD_NEXT ) )
       {
          if ( jBean.getOrganicGardening() == true )
          {
            return page( VIEW_GARDENING );
```

Copyright © 2002-2003 The Apache Software Foundation. All rights reserved.

```
           }
           else if ( jBean.getCooking() == true )
           {
             return page( VIEW_COOKING );
           }
           else if ( jBean.getSmallholdingManagement() == true )
           {
             return page( VIEW_SMALLHOLDING );
           }
           //else if ( getForm().get
          return page(  VIEW_CONFIRM );
        }
        if ( command.equals( CMD_PREV ) )
        {
           return page( VIEW_REGISTRATION );
        }
      }
      else if ( formView.equals ( VIEW_GARDENING ) )
      {
        if ( command.equals ( CMD_NEXT ) )
        {
           if ( jBean.getCooking() == true )
           {
             return page( VIEW_COOKING );
           }
           else if ( jBean.getSmallholdingManagement() == true )
           {
             return page( VIEW_SMALLHOLDING );
           }
          return page( VIEW_CONFIRM );
        }
        else if( command.equals( CMD_PREV ) )
        {
          return page( VIEW_INTEREST );
        }
      }
      else if ( formView.equals ( VIEW_COOKING ) )
      {
        if ( command.equals ( CMD_NEXT ) )
        {
           if ( jBean.getSmallholdingManagement() == true )
           {
             return page( VIEW_SMALLHOLDING );
           }
          return page( VIEW_CONFIRM );
        }
        else if ( command.equals( CMD_PREV ) )
        {
          if ( jBean.getOrganicGardening() == true )
          {
            return page( VIEW_GARDENING );
          }
          return page( VIEW_INTEREST );
        }
```

Copyright © 2002-2003 The Apache Software Foundation. All rights reserved.

```
      }
      else if ( formView.equals ( VIEW_SMALLHOLDING ) )
      {
        if ( command.equals( CMD_NEXT ) )
        {
          return page( VIEW_CONFIRM );
        }
        else if ( command.equals( CMD_PREV ) )
        {
          if ( jBean.getCooking() == true )
          {
            return page( VIEW_COOKING );
          }
          else if ( jBean.getOrganicGardening() == true )
          {
            return page( VIEW_GARDENING );
          }
          return page( VIEW_INTEREST );
        }
      }
      else if ( formView.equals ( VIEW_CONFIRM ) )
      {
        if ( command.equals( CMD_NEXT ) )
        {
           return page( VIEW_END );
        }
        else if( command.equals( CMD_PREV ) )
        {
           if ( jBean.getOrganicGardening() == true )
           {
             return page( VIEW_GARDENING );
           }
          return page( VIEW_INTEREST );
        }
      }
    }

    // should never reach this statement
    return page( VIEW_START );

  }
```

1.4. Reset Method

The reset method is used to tidy up any checkboxes and can be used to reset other fields in the form.

```
  /**
```

Copyright © 2002-2003 The Apache Software Foundation. All rights reserved.

```
   *
   * FormListener callback
   * called in the beginning Form.populate()
   * before population starts.
   *
   * This is the place to handle unchecked checkboxes.
   *
   */
  public void reset( Form form )
  {
    // based on the current form view
    // make some decisions regarding checkboxes, etc.
    String formView = getFormView();
    if ( formView.equals ( VIEW_INTEREST ) )
    {
      // deal with the organicGardening checkbox
      form.setValue( "/organicGardening", Boolean.FALSE );
      // deal with the cooking checkbox
      form.setValue( "/cooking", Boolean.FALSE );
      // deal with the smallholdingManagement checkbox
      form.setValue( "/smallholdingManagement", Boolean.FALSE );
    }
    else if ( formView.equals ( VIEW_GARDENING ) )
    {
      // deal with the flowers checkbox
      form.setValue( "/flowers", Boolean.FALSE );
      // deal with the vegetables checkbox
      form.setValue( "/vegetables", Boolean.FALSE );
      // deal with the fruitTrees checkbox
      form.setValue( "/fruitTrees", Boolean.FALSE );
    }
    else if ( formView.equals ( VIEW_COOKING ) )
    {
     // deal with the traditionalReciepes checkbox
      form.setValue( "/traditionalReciepes", Boolean.FALSE );
      // deal with the soups checkbox
      form.setValue( "/soups", Boolean.FALSE );
      // deal with the veganCookery checkbox
      form.setValue( "/veganCookery", Boolean.FALSE );
    }
    else if ( formView.equals ( VIEW_SMALLHOLDING ) )
    {
    // deal with the pigKeeping checkbox
      form.setValue( "/pigKeeping", Boolean.FALSE );
      // deal with the pygmyGoats checkbox
      form.setValue( "/pygmyGoats", Boolean.FALSE );
      // deal with the henKeeping checkbox
      form.setValue( "/henKeeping", Boolean.FALSE );
    }

  }
```

Copyright © 2002-2003 The Apache Software Foundation. All rights reserved.

The whole file HowtoWizardAction.java is below for you to copy to the folder C:\projects\apache\xml-cocoon2\src\scratchpad\src\org\apache\cocoon\samples:

```
/*
 * $Header: /home/cvspublic/xml-cocoon2/src/scratchpad/src/org/apache
 *     /cocoon/samples/xmlform/HowtoWizardAction.java,
 v 1.2 2002/05/09 07:26:07 ivelin Exp $
 * $Revision: 1.5 $
 * $Date: 2003/08/13 10:34:56 $
 *
 * ====================================================================
 * The Apache Software License, Version 1.1
 *
 *
 *
 * Copyright (c) 1999-2001 The Apache Software Foundation.  All rights
 * reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. The end-user documentation included with the redistribution, if
 *    any, must include the following acknowlegement:
 *       "This product includes software developed by the
 *        Apache Software Foundation (http://www.apache.org/)."
 *    Alternately, this acknowlegement may appear in the software itself,
 *    if and wherever such third-party acknowlegements normally appear.
 *
 * 4. The names "The Jakarta Project", "Commons", and "Apache Software
 *    Foundation" must not be used to endorse or promote products derived
 *    from this software without prior written permission. For written
 *    permission, please contact apache@apache.org.
 *
 * 5. Products derived from this software may not be called "Apache"
 *    nor may "Apache" appear in their names without prior written
 *    permission of the Apache Group.
 *
 * THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED
 * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
 * OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED.  IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
```

Copyright © 2002-2003 The Apache Software Foundation. All rights reserved.

```
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
 * USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
 * OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 * ====================================================================
 *
 * This software consists of voluntary contributions made by many
 * individuals on behalf of the Apache Software Foundation and was
 * originally based on software copyright (c) 2001, Plotnix, Inc,
 * <http://www.plotnix.com/>.
 * For more information on the Apache Software Foundation, please see
 * <http://www.apache.org/>.
 */
package org.apache.cocoon.samples.xmlform;


// Java classes
import java.util.Map;
import java.util.HashMap;
import java.util.SortedSet;
import java.util.Iterator;
import java.util.Properties;
import java.io.InputStream;
import java.io.FileInputStream;
import java.io.File;

// XML classes
import javax.xml.transform.stream.StreamSource;
import javax.xml.transform.TransformerException;
import org.xml.sax.InputSource;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

// Framework classes
import org.apache.avalon.framework.parameters.Parameters;
import org.apache.avalon.excalibur.pool.Poolable;
import org.apache.avalon.framework.configuration.Configuration;
import org.apache.avalon.framework.configuration.ConfigurationException;

// Cocoon classes
import org.apache.cocoon.environment.Redirector;
import org.apache.cocoon.environment.SourceResolver;
import org.apache.cocoon.acting.*;
import org.apache.cocoon.environment.Request;
import org.apache.cocoon.environment.ObjectModelHelper;
import org.apache.cocoon.environment.Session;
import org.apache.cocoon.environment.Context;

// Schematron classes
import org.apache.cocoon.validation.SchemaFactory;
import org.apache.cocoon.validation.Schema;
import org.apache.cocoon.validation.Validator;
```

Copyright © 2002-2003 The Apache Software Foundation. All rights reserved.

```
import org.apache.cocoon.validation.Violation;

// Cocoon Form
import org.apache.cocoon.acting.AbstractXMLFormAction;
import org.apache.cocoon.xmlform.Form;
import org.apache.cocoon.xmlform.FormListener;


/**
 * This action demonstrates
 * a relatively complex form handling scenario.
 *
 * @author Ivelin Ivanov <ivelin@apache.org>
 */
public class HowtoWizardAction
  extends AbstractXMLFormAction
  implements FormListener

{


  // different form views
  // participating in the wizard
  final String VIEW_START = "start";
  final String VIEW_REGISTRATION = "registration";
  final String VIEW_INTEREST = "interest";
  final String VIEW_GARDENING = "organicGardening";
  final String VIEW_COOKING = "cooking";
  final String VIEW_SMALLHOLDING = "smallholdingManagement";
  final String VIEW_CONFIRM = "confirm";
  final String VIEW_END = "end";

  // action commands used in the wizard
  final String CMD_START = "start";
  final String CMD_NEXT = "next";
  final String CMD_PREV = "prev";


  /**
   * The first callback method which is called
   * when an action is invoked.
   *
   * It is called before population.
   *
   *
   * @return null if the Action is prepared to continue.
   * an objectModel map which will be immediately returned by the action.
   *
   * This method is a good place to handle buttons with Cancel
   * kind of semantics. For example
   * <pre>if getCommand().equals("Cancel") return page("input");</pre>
   *
   */
  protected Map prepare()
```

Copyright © 2002-2003 The Apache Software Foundation. All rights reserved.

```
{

  if ( getCommand() == null )
    {
      return page( VIEW_START );
    }
  else   if ( getCommand().equals( CMD_START ) )
  {
    // reset state by removing old form
    // if one exists
    Form.remove( getObjectModel(), getFormId() );
    getForm().addFormListener( this );

    return page( VIEW_REGISTRATION );
  }


  // get ready for action
  // if not ready return page("whereNext");
  return null;
}


/**
 * Invoked after form population
 *
 * Semanticly similar to Struts Action.perform()
 *
 * Take appropriate action based on the command
 *
 */
public Map perform ()
{

  // get the actual model which this Form encapsulates
  // and apply additional buziness logic to the model
  HowToBean  jBean = (HowToBean) getForm().getModel();
  //jBean.incrementCount();

  // set the page control flow parameter
  // according to the validation result
  if ( getCommand().equals( CMD_NEXT ) &&
    getForm().getViolations () != null )
  {
    // errors, back to the same page
    return page( getFormView() );
  }
  else
  {
    // validation passed
    // continue with control flow

    // clear validation left overs in case the user
    // did not press the Next button
```

Copyright © 2002-2003 The Apache Software Foundation. All rights reserved.

```
      getForm().clearViolations();

      // get the user submitted command (through a submit button)
      String command = getCommand();
      // get the form view which was submitted
      String formView = getFormView();

      // apply control flow rules
      if ( formView.equals ( VIEW_REGISTRATION ) )
      {
        if ( command.equals( CMD_NEXT ) )
        {
          return page(  VIEW_INTEREST );
        }
      }
      else if ( formView.equals ( VIEW_INTEREST ) )
      {
        if ( command.equals( CMD_NEXT ) )
        {
           if ( jBean.getOrganicGardening() == true )
           {
             return page( VIEW_GARDENING );
           }
           else if ( jBean.getCooking() == true )
           {
             return page( VIEW_COOKING );
           }
           else if ( jBean.getSmallholdingManagement() == true )
           {
             return page( VIEW_SMALLHOLDING );
           }
           //else if ( getForm().get
          return page(  VIEW_CONFIRM );
        }
        if ( command.equals( CMD_PREV ) )
        {
           return page( VIEW_REGISTRATION );
        }
      }
      else if ( formView.equals ( VIEW_GARDENING ) )
      {
        if ( command.equals ( CMD_NEXT ) )
        {
           if ( jBean.getCooking() == true )
           {
             return page( VIEW_COOKING );
           }
           else if ( jBean.getSmallholdingManagement() == true )
           {
             return page( VIEW_SMALLHOLDING );
           }
          return page( VIEW_CONFIRM );
        }
        else if( command.equals( CMD_PREV ) )
```

Copyright © 2002-2003 The Apache Software Foundation. All rights reserved.

```
        {
          return page( VIEW_INTEREST );
        }
      }
      else if ( formView.equals ( VIEW_COOKING ) )
      {
        if ( command.equals ( CMD_NEXT ) )
        {
           if ( jBean.getSmallholdingManagement() == true )
           {
             return page( VIEW_SMALLHOLDING );
           }
          return page( VIEW_CONFIRM );
        }
        else if ( command.equals( CMD_PREV ) )
        {
          if ( jBean.getOrganicGardening() == true )
          {
            return page( VIEW_GARDENING );
          }
          return page( VIEW_INTEREST );
        }
      }
      else if ( formView.equals ( VIEW_SMALLHOLDING ) )
      {
        if ( command.equals( CMD_NEXT ) )
        {
          return page( VIEW_CONFIRM );
        }
        else if ( command.equals( CMD_PREV ) )
        {
          if ( jBean.getCooking() == true )
          {
            return page( VIEW_COOKING );
          }
          else if ( jBean.getOrganicGardening() == true )
          {
            return page( VIEW_GARDENING );
          }
          return page( VIEW_INTEREST );
        }
      }
      else if ( formView.equals ( VIEW_CONFIRM ) )
      {
        if ( command.equals( CMD_NEXT ) )
        {
           return page( VIEW_END );
        }
        else if( command.equals( CMD_PREV ) )
        {
           if ( jBean.getOrganicGardening() == true )
           {
             return page( VIEW_GARDENING );
           }
```

Copyright © 2002-2003 The Apache Software Foundation. All rights reserved.

```
          return page( VIEW_INTEREST );
        }
      }
    }

    // should never reach this statement
    return page( VIEW_START );

  }





  /**
   *
   * FormListener callback
   * called in the beginning Form.populate()
   * before population starts.
   *
   * This is the place to handle unchecked checkboxes.
   *
   */
  public void reset( Form form )
  {
    // based on the current form view
    // make some decisions regarding checkboxes, etc.
    String formView = getFormView();
    if ( formView.equals ( VIEW_INTEREST ) )
    {
      // deal with the organicGardening checkbox
      form.setValue( "/organicGardening", Boolean.FALSE );
      // deal with the cooking checkbox
      form.setValue( "/cooking", Boolean.FALSE );
      // deal with the smallholdingManagement checkbox
      form.setValue( "/smallholdingManagement", Boolean.FALSE );
    }
    else if ( formView.equals ( VIEW_GARDENING ) )
    {
      // deal with the flowers checkbox
      form.setValue( "/flowers", Boolean.FALSE );
      // deal with the vegetables checkbox
      form.setValue( "/vegetables", Boolean.FALSE );
      // deal with the fruitTrees checkbox
      form.setValue( "/fruitTrees", Boolean.FALSE );
    }
    else if ( formView.equals ( VIEW_COOKING ) )
    {
     // deal with the traditionalReciepes checkbox
      form.setValue( "/traditionalReciepes", Boolean.FALSE );
      // deal with the soups checkbox
      form.setValue( "/soups", Boolean.FALSE );
      // deal with the veganCookery checkbox
      form.setValue( "/veganCookery", Boolean.FALSE );
```

Copyright © 2002-2003 The Apache Software Foundation. All rights reserved.

```
    }
    else if ( formView.equals ( VIEW_SMALLHOLDING ) )
    {
    // deal with the pigKeeping checkbox
      form.setValue( "/pigKeeping", Boolean.FALSE );
      // deal with the pygmyGoats checkbox
      form.setValue( "/pygmyGoats", Boolean.FALSE );
      // deal with the henKeeping checkbox
      form.setValue( "/henKeeping", Boolean.FALSE );
    }

  }


  /**
   * FormListener callback
   *
   * Invoked during Form.populate();
   *
   * It is invoked before a request parameter is mapped to
   * an attribute of the form model.
   *
   * It is appropriate to use this method for filtering
   * custom request parameters which do not reference
   * the model.
   *
   * Another appropriate use of this method is for graceful filtering of invalid
   * values, in case that knowledge of the system state or
   * other circumstainces make the standard validation
   * insufficient. For example if a registering user choses a username which
   * is already taken - the check requires database transaction, which is
   * beyond the scope of document validating schemas.
   * Of course customized Validators can be implemented to do
   * this kind of domain specific validation
   * instead of using this method.
   *
   *
   * @return false if the request parameter should not be filtered.
   * true otherwise.
   */
  public boolean filterRequestParameter (Form form, String parameterName)
  {
    // TBD
    return false;
  }


  public  String getFile( String FileName ) {
    try
    {
      final String  FILE_PREFIX = "file:";
      String path = getSourceResolver().resolve(FileName).getSystemId();
      if(path.startsWith(FILE_PREFIX))
         path = path.substring(FILE_PREFIX.length());
```

Copyright © 2002-2003 The Apache Software Foundation. All rights reserved.

```
      return path;
    }
    catch(Exception e)
    {
       getLogger().error("could not read mapping file",e);
      return null;
    }
  }

  private Validator validator_ = null;
  private boolean initialized_ = false;

}
```

Finally Step 5: the Sitemap

2. Revisions

Find a problem with this document? Consider contacting the mailing lists or submitting your own revision. For instructions, read the How To Submit a Revision.

Copyright © 2002-2003 The Apache Software Foundation. All rights reserved.