PubMedPrefill-JSPPubmedPrefillStep.java

```
package my.dspace.app.webui.submit.step;

import org.dspace.app.webui.submit.JSPStep;
import org.dspace.app.webui.submit.JSPStepManager;
import org.dspace.app.webui.util.UIUtil;
import org.dspace.app.util.SubmissionInfo;
import org.dspace.core.Context;
import org.dspace.core.ConfigurationManager;
import org.dspace.authorize.AuthorizeException;
import org.dspace.submit.step.SampleStep;
import org.dspace.submit.AbstractProcessingStep;
import org.apache.log4j.Logger;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;
import java.io.IOException;
import java.sql.SQLException;

/**
 * Sample PubMed Prefill JSP interface layer
 */
public class JSPPubmedPrefillStep extends JSPStep
{
        /** log4j logger */
        private static Logger log = Logger.getLogger(JSPPubmedPrefillStep.class);

        /** JSP which displays the step to the user * */
        private static final String DISPLAY_JSP = "/submit/pubmed-step.jsp";

        /** JSP which displays information to be reviewed during 'verify step' * */
        private static final String REVIEW_JSP = "/submit/review-pubmed.jsp";

        /**
         * Do any pre-processing to determine which JSP (if any) is used to generate
         * the UI for this step. This method should include the gathering and
         * validating of all data required by the JSP. In addition, if the JSP
         * requires any variable to passed to it on the Request, this method should
         * set those variables.
         * <P>
         * If this step requires user interaction, then this method must call the
         * JSP to display, using the "showJSP()" method of the JSPStepManager class.
         * <P>
         * If this step doesn't require user interaction OR you are solely using
         * Manakin for your user interface, then this method may be left EMPTY,
         * since all step processing should occur in the doProcessing() method.
         *
         * @param context
         *            current DSpace context
         * @param request
         *            current servlet request object
         * @param response
         *            current servlet response object
         * @param subInfo
         *            submission info object
         */
        public void doPreProcessing(Context context, HttpServletRequest request,
                        HttpServletResponse response, SubmissionInfo subInfo)
                        throws ServletException, IOException, SQLException,
                                        AuthorizeException
        {
                // Tell JSPStepManager class to load "sample-step.jsp"
                JSPStepManager.showJSP(request, response, subInfo, DISPLAY_JSP);
        }

        /**
```

```
     * Do any post-processing after the step's backend processing occurred (in
     * the doProcessing() method).
     * <P>
     * It is this method's job to determine whether processing completed
     * successfully, or display another JSP informing the users of any potential
     * problems/errors.
     * <P>
     * If this step doesn't require user interaction OR you are solely using
     * Manakin for your user interface, then this method may be left EMPTY,
     * since all step processing should occur in the doProcessing() method.
     *
     * @param context
     *            current DSpace context
     * @param request
     *            current servlet request object
     * @param response
     *            current servlet response object
     * @param subInfo
     *            submission info object
     * @param status
     *            any status/errors reported by doProcessing() method
     */
    public void doPostProcessing(Context context, HttpServletRequest request,
                   HttpServletResponse response, SubmissionInfo subInfo, int status)
                   throws ServletException, IOException, SQLException,
                   AuthorizeException
    {
            /*********************************************************************
             * IMPORTANT FUNCTIONS to be aware of :
             *********************************************************************/

            // This function retrieves the path of the JSP which just submitted its
            // form to this class (e.g. "/submit/sample-step.jsp", in this case)
            String lastJSPDisplayed = JSPStepManager.getLastJSPDisplayed(request);

            // This function retrieves the number of the current "page"
            // within this Step. This is useful if your step actually
            // has more than one "page" within the Progress Bar. It can
            // help you determine which Page the user just came from,
            // as well as determine which JSP to load in doPreProcessing()
            int currentPageNum = SampleStep.getCurrentPage(request);

            // This function returns the NAME of the button the user
            // just pressed in order to submit the form.
            // In this case, we are saying default to the "Next" button,
            // if it cannot be determined which button was pressed.
            // (requires you use the AbstractProcessingStep.PREVIOUS_BUTTON,
            // AbstractProcessingStep.NEXT_BUTTON, and AbstractProcessingStep.CANCEL_BUTTON
            // constants in your JSPs)
            String buttonPressed = UIUtil.getSubmitButton(request,
                           AbstractProcessingStep.NEXT_BUTTON);

            // We also have some Button Name constants to work with.
            // Assuming you used these constants to NAME your submit buttons,
            // we can do different processing based on which button was pressed
            if (buttonPressed.equals(AbstractProcessingStep.NEXT_BUTTON))
            {
                    // special processing for "Next" button
                    // YOU DON'T NEED TO ATTEMPT TO REDIRECT/FORWARD TO THE NEXT PAGE
                    // HERE,
                    // the SubmissionController will do that automatically!
            }
            else if (buttonPressed.equals(AbstractProcessingStep.PREVIOUS_BUTTON))
            {
                    // special processing for "Previous" button
                    // YOU DON'T NEED TO ATTEMPT TO REDIRECT/FORWARD TO THE PREVIOUS
                    // PAGE HERE,
                    // the SubmissionController will do that automatically!
            }
            else if (buttonPressed.equals(AbstractProcessingStep.CANCEL_BUTTON))
            {
```

```
                // special processing for "Cancel/Save" button
                // YOU DON'T NEED TO ATTEMPT TO REDIRECT/FORWARD TO THE CANCEL/SAVE
                // PAGE HERE,
                // the SubmissionController will do that automatically!
        }

        // Here's some sample error message processing!
        if (status == SampleStep.STATUS_USER_INPUT_ERROR)
        {
                // special processing for this error message
                JSPStepManager.showJSP(request, response, subInfo, DISPLAY_JSP);
        }

        /*********************************************************************
         * SAMPLE CODE (all of which is commented out)
         *
         * (For additional sample code, see any of the existing JSPStep classes)
         ********************************************************************/

        /*
         * HOW-TO RELOAD PAGE BECAUSE OF INVALID INPUT!
         *
         * If you have already validated the form inputs, and determined that
         * one or more is invalid, you can RELOAD the JSP by calling
         * JSPStepManger.showJSP() like:
         *
         * JSPStepManger.showJSP(request, response, subInfo, "/submit/sample-step.jsp");
         *
         * You should make sure to pass a flag to your JSP to let it know which
         * fields were invalid, so that it can display an error message next to
         * them:
         *
         * request.setAttribute("invalid-fields", listOfInvalidFields);
         */

        /*
         * HOW-TO GO TO THE NEXT "PAGE" IN THIS STEP
         *
         * If this step has multiple "pages" that appear in the Progress Bar,
         * you can step to the next page AUTOMATICALLY by just NOT calling
         * "JSPStepManger.showJSP()" in your doPostProcessing() method.
         *
         */

        /*
         * HOW-TO COMPLETE/END THIS STEP
         *
         * In order to complete this step, just do NOT call JSPStepManger.showJSP()! Once all
         * pages are finished, the JSPStepManager class will report to the
         * SubmissionController that this step is now finished!
         */
}

/**
 * Retrieves the number of pages that this "step" extends over. This method
 * is used by the SubmissionController to build the progress bar.
 * <P>
 * This method may just return 1 for most steps (since most steps consist of
 * a single page). But, it should return a number greater than 1 for any
 * "step" which spans across a number of HTML pages. For example, the
 * configurable "Describe" step (configured using input-forms.xml) overrides
 * this method to return the number of pages that are defined by its
 * configuration file.
 * <P>
 * Steps which are non-interactive (i.e. they do not display an interface to
 * the user) should return a value of 1, so that they are only processed
 * once!
 *
 *
 * @param request
 *            The HTTP Request
```

```
         * @param subInfo
         *            The current submission information object
         *
         * @return the number of pages in this step
         */
        public int getNumberOfPages(HttpServletRequest request,
                        SubmissionInfo subInfo) throws ServletException
        {
                /*
                 * This method tells the SubmissionController how many "pages" to put in
                 * the Progress Bar for this Step.
                 *
                 * Most steps should just return 1 (which means the Step only appears
                 * once in the Progress Bar).
                 *
                 * If this Step should be shown as multiple "Pages" in the Progress Bar,
                 * then return a value higher than 1. For example, return 2 in order to
                 * have this Step appear twice in a row within the Progress Bar.
                 *
                 * If you return 0, this Step will not appear in the Progress Bar at
                 * ALL! Therefore it is important for non-interactive steps to return 0.
                 */

                // in most cases, you'll want to just return 1
                return 1;
        }

        /**
         * Return the URL path (e.g. /submit/review-metadata.jsp) of the JSP
         * which will review the information that was gathered in this Step.
         * <P>
         * This Review JSP is loaded by the 'Verify' Step, in order to dynamically
         * generate a submission verification page consisting of the information
         * gathered in all the enabled submission steps.
         *
         * @param context
         *            current DSpace context
         * @param request
         *            current servlet request object
         * @param response
         *            current servlet response object
         * @param subInfo
         *            submission info object
         */
        public String getReviewJSP(Context context, HttpServletRequest request,
                        HttpServletResponse response, SubmissionInfo subInfo)
        {
                return REVIEW_JSP;
        }
}
```