PubMedPrefill-PubmedPrefillStep.java

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

import org.dspace.submit.AbstractProcessingStep;
import org.dspace.submit.step.SampleStep;
import org.dspace.core.Context;
import org.dspace.core.PluginManager;
import org.dspace.app.util.SubmissionInfo;
import org.dspace.app.webui.util.UIUtil;
import org.dspace.authorize.AuthorizeException;
import org.dspace.content.Item;
import org.dspace.content.crosswalk.IngestionCrosswalk;
import org.apache.commons.lang.StringUtils;
import org.apache.log4j.Logger;
import org.jdom.Element;
import org.jdom.input.DOMBuilder;
import org.w3c.dom.Document;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import java.io.IOException;
import java.sql.SQLException;

/**
 * Main processing code for PubMed lookups
 */
public class PubmedPrefillStep extends AbstractProcessingStep
{
    /** log4j logger */
    private static Logger log = Logger.getLogger(PubmedPrefillStep.class);

    public int doProcessing(Context context, HttpServletRequest httpServletRequest, HttpServletResponse
httpServletResponse, SubmissionInfo submissionInfo) throws ServletException, IOException, SQLException,
AuthorizeException
    {
        String identifier = httpServletRequest.getParameter("identifier_pmid");
        String originalId = httpServletRequest.getParameter("original_pmid");
        boolean forceLookup = UIUtil.getBoolParameter(httpServletRequest, "force_refresh");

        // If we have an identifier, and it:
        // - is not the same as when the form was build (ie. it is a first time entry, or has been changed)
        // - the lookup has been forced
        // then fill the metadata with the new values
        if (!StringUtils.isEmpty(identifier))
        {
            if (!identifier.equals(originalId) || forceLookup)
            {
                if (!isValidPubmedID(identifier))
                {
                    httpServletRequest.setAttribute("identifier.error", true);
                    return SampleStep.STATUS_USER_INPUT_ERROR;
                }

                // Clear data only if the search param is valid
                clearMetaDataValues(submissionInfo);

                if (!fetchPubmedData(context, submissionInfo, identifier))
                {
                    // the Fetch failed..display the inability to get details..and allow user to enter
                    // details manually
                    httpServletRequest.setAttribute("identifier.info", Boolean.valueOf(true));

                    return SampleStep.STATUS_USER_INPUT_ERROR;
                }
```

```
            }
        }

        return SampleStep.STATUS_COMPLETE;
    }

    public int getNumberOfPages(HttpServletRequest httpServletRequest, SubmissionInfo submissionInfo) throws
ServletException
    {
        return 1;
    }

    protected boolean isValidPubmedID(String pmid)
    {
        try
        {
            // A valid PMID will be parseable as an integer
            return (Integer.parseInt(pmid, 10) > 0);
        }
        catch (NumberFormatException nfe)
        {
            return false;
        }
    }

    protected Element retrievePubmedXML(String pmid) throws Exception
    {
        try
        {
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder parser = factory.newDocumentBuilder();
            Document doc = parser.parse("http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?
db=pubmed&retmode=xml&id=" + pmid);

            // Check to see if document is a valid article...all article has the DTD specified
            if(doc.getDoctype() != null)
            {
                // There seem to be some illegal PI in the xml doc, hence getting
                // the root element and then convert to JDOM Element
                org.w3c.dom.Element element = doc.getDocumentElement();
                DOMBuilder builder = new DOMBuilder();
                return builder.build(element);
            }

        }
        catch (Exception ex)
        {
            throw new Exception("Extracting PubMedID: "+ex.getMessage(),ex.getCause());
        }

        return null;
    }

    protected void clearMetaDataValues(SubmissionInfo subInfo)
    {
        Item item = subInfo.getSubmissionItem().getItem();
        item.clearMetadata(Item.ANY, Item.ANY, Item.ANY, Item.ANY);
    }

    protected boolean fetchPubmedData(Context context, SubmissionInfo subInfo, String pmid)
    {
        try
        {
            Element jElement = retrievePubmedXML(pmid);
            if(jElement != null)
            {
                // Use the ingest process to parse the XML document, transformation is done
                // using XSLT
                IngestionCrosswalk xwalk = (IngestionCrosswalk) PluginManager.getNamedPlugin(
                                                IngestionCrosswalk.class, "PMID");
                xwalk.ingest(context, subInfo.getSubmissionItem().getItem(), jElement);
```

```
                return true;
            }
        }
        catch (Exception ex)
        {
            // As this is not a critical error...will not terminate the process...allow user to
            // manually enter details
            log.error("Unable to fetch & load external data: "+ ex.getMessage(), ex.getCause());
        }
        return false;
    }
}
```