File Uploads with RSF

Adding File Uploads to your HTML Forms with RSF is relatively straightforward. There are, however, a few caveats which can trip one up.

Step One: Check your web.xml

Be sure your Sakai Request Filter has the upload.enabled parameter sent to false. Otherwise, the uploads won't get through and it will appear as if nothing is being uploaded.

Web.xml for Application with File Uploading

```
<filter>
  <filter-name>sakai.request</filter-name>
  <filter-class>org.sakaiproject.util.RequestFilter</filter-class>
  <init-param>
    <param-name>upload.enabled</param-name>
    <param-value>false</param-value>
  </init-param>
</filter>
```

Step Two: Add the form to your HTML Template

There are some important things to be aware of in your HTML form

1. Your form method must be post
2. You must include the enctype and set it to "multipart/form-data"
3. Your input tag for each file must have a name parameter. Also, the input file tags should not have rsf:id's. In your request scope bean, you will get a java.util.Map of all the file uploads. This means you can have as many as you want in the form, but they should each have a unique name.
4. All the rest of your input tags are normal and should have rsf:id's if you want to use them.

HTML Form with File Uploads

```
<form rsf:id="uploadform" method="post" enctype="multipart/form-data">
  <input name="fileupload1" type="file" /><br/>
  <input name="fileupload2" type="file" /><br/>
  <input name="fileupload3" type="file" /><br/>
  <input rsf:id="submit" type="submit" value="Upload"/>
</form>
```

Step Three: Hook up the form in your View Producer

Since the input tags for file uploads must not have rsf:id's, you do not need to fill them in your producer. The rest of the tags can be filled as usual. Because your form must be of method POST, you will most likely want to implement NavigationCaseReporter, to redirect to the desired page.

In this example we are binding to a bean called wizarddata. We'll put the logic for dealing with the uploaded files there.

View Producer with File Upload form

```
public class Step1Producer implements ViewComponentProducer, NavigationCaseReporter {
  public static final String VIEWID = "ImportWizardPage1";

  public String getViewID() {
    return VIEWID;
  }

  public void fillComponents(UIContainer tofill, ViewParameters viewparams, ComponentChecker checker) {
    UIForm uploadform = UIForm.make(tofill, "uploadform" );
    UICommand.make(uploadform, "submit", "wizarddata.uploadExportFile");
  }

  public List reportNavigationCases() {
    List i = new ArrayList();
    i.add(new NavigationCase("success", new SimpleViewParameters("ImportWizardPage2")));
    return i;
  }
```

Step Four: Add some logic to process the uploaded files

The example below is the wizarddata bean we used above.

When you iterate through the keys of multipart map, each stored object will be of type org.springframework.web.multipart. MultipartFile. Using each MultipartFile you can get the name, size, and data for each uploaded file. Generally, multipartMap will have an entry for each input file upload tag in the form. If a user does not select a file for one of the uploaded entries, that entry will still show up in the Map, and MultipartFile.getSize() will return 0 for that entry.

Processing Uploaded Files

```
public class WizardData {
  public Map multipartMap;

  public String uploadExportFile() {
    System.out.println("Number of uploaded files: " + multipartMap.size());
    return "success";
  }
}
```

When you declare this request scope bean, you will need to inject it with multipart map. You can name your property anything, but the ref must be multipartMap. You can make this property a pea, or use the traditional JavaBean style getter and setter. The mutlipartMap bean is declared in RSF's configuration, and will automatically be populated whenever you submit a form with the proper enctype containing file upload input tags.

Request Scope declaration for the logic bean

```
<bean id="wizarddata" class="nuchem.clickers.importwizard.WizardData">
  <property name="multipartMap" ref="multipartMap" />
</bean>
```