Uploading Files

Tapestry provides a file upload component based on Apache Commons FileUpload to make it easier to handle files uploaded through web forms (via the standard <input type="file"> HTML element).

Downloading

tapestry-upload is not automatically included in Tapestry applications because of the additional dependencies it requires. To include it, just add the `tapestry-upload` dependency to the pom of your application, something like this:

```
<dependency>
    <groupId>org.apache.tapestry</groupId>
    <artifactId>tapestry-upload</artifactId>
    <version>${tapestry-release-version}</version>
</dependency>
```

If you aren't using Maven, you'll have to download the jar and its dependencies yourself.

Usage

The upload component supports default value binding (based on id) and validation.

JumpStart Demo:
File Upload

Component Template

```
    <t:form>
        <t:errors/>
        <input t:type="upload" t:id="file" t:value="file" validate="required"/>
        <br/>
        <input type="submit" value="Upload"/>
    </t:form>
```

Here, because the value parameter was not bound, the component used the file property of its container (because the component's id is 'file'). If you want to upload as a different property, either bind the value parameter or change the component's id.

Page class

```
    public class UploadExample
    {
        @Property
        private UploadedFile file;

        public void onSuccess()
        {
            File copied = new File("/my/file/location/" + file.getFileName());

            file.write(copied);
        }
    }
```

Upload Exceptions

In some cases, file uploads may fail. This can be because of a simple communication exception, or more likely, because the configured maximum upload size was exceeded.

When a file upload exception occurs, Tapestry will trigger a "uploadException" event on the page to notify it of the error. All other normal processing is skipped (no "activate" event, no form submission, etc.).

The event handler should return a non-null object, which will be handled as a navigational result. Example:

```
    @Persist(PersistenceConstants.FLASH)
     @Property
     private String message;


     Object onUploadException(FileUploadException ex)
     {
         message = "Upload exception: " + ex.getMessage();

         return this;
     }
```

Note the importance of `return this;`. A void event handler method, or one that returns null, will result in the FileUploadException being reported to the user as an uncaught runtime exception.

Configuration

Four values may be configured as symbols:

upload.repository-location	The directory to which files that are too large to keep in memory will be written to. The default is from the java.io.tmpdir system property.
upload.repository-threshold	Upload size, in bytes, at which point the uploaded file is written to disk rather than kept in memory. The default is 10 kilobytes.
upload.requestsize-max	Maximim size, in bytes, for the overall request. If exceeded, a FileUploadException will occur. The default is no maximum.
upload.filesize-max	Maximum size, in bytes, for any individual uploaded file. Again, a FileUploadException will occur if exceeded. The default is no maximum.

The class UploadSymbols defines constants for all four of these.

Potential Issues

The Commons FileUpload library uses the CommonsIO file cleaner service to remove temporary files when they are no longer needed. This service creates a thread to carry out its work. If the commons-io library is shared amongst multiple applications (e.g. added to server classpath) it is possible for an application to terminate this thread prematurely and cause errors for the other applications. (see the Resource Cleanup section in for more discussion)

Technically the file cleanup service is not needed by Tapestry Upload (which deletes temporary files at the end of request processing). However it is currently not possible to disable it (enhancement request has been filed as FILEUPLOAD-133).