FileUploadsWithCocoon2.1

There were some important changes to file uploading in Cocoon in April 2003, just before the release of 2.1M1. If you have used a version of Cocoon prior to that, pay careful attention to the differences.

File uploads are quite easy to deal with in Cocoon and greater things are planned for later versions of Cocoon.

Configuration

Unlike most configuration in Cocoon, all the options controlling file uploads in Cocoon 2.1 are set in web.xml:

parameter name	default value	Description (see web.xml comments for more)	
enable-uploads	false	"Turn on" upload handling in Cocoon. As a convenience can also be set from a local build property config.enable-uploads=true	
upload-directory	upload-dir under the "work directory"	where Cocoon should put temporary uploaded files on disk if configured to do so	
autosave-uploads	true	Causes all files to be saved temporarily to upload-dir during the request	
overwrite-uploads	rename	what to do with name conflicts with existing file. Acceptable values are **deny**, **allow**, **rename** (default, but could possibly lead to endless loop condition with concurrent requests using the same upload file name)	
upload-max-size	102 400 bytes	maximum allowed size of uploads; normal default of 10Mb is overwritten in web.xml "as shipped" to allow Cocoon samples to run	

Examples

There are several approaches to handling uploads in Cocoon:

- FileUploadsWithFlow
- FileUploadWithAction (careful, written for 2.0 and needs to be rewritten) for an action which retrieves a FilePartFile (see!) object by name from the request (requires autosave-uploads=true) for further processing (i.e., move to different directory, save to Blob in database, etc.) If you want to create an action in 2.1 and need help, there is a tip at the end of the flow example.
- FileUploadsInCustomActions
- RecipeUploadUsingAction provides working code which can easily be modified to support uploading

Technical Overview

..for those who are interested...

If uploads are "turned on", when the Cocoon servlet receives a request, org.apache.cocoon.servlet.multipart.RequestFactory parses out the multipart data which comprises the file data, and places an org.apache.cocoon.servlet.multipart.Part object in the request using the name of the multipart file parameter (the name of the <input type="file" ...> from the html). This object provides access methods for later Cocoon steps to deal with the file. It handles multiple files by placing multiple objects in the request.

Part is abstract, but has two concrete subclasses: PartOnDisk and PartInMemory which are the actual objects placed in the request as described above, depending on configuration options to be described below. PartOnDisk is a file already saved to disk and provides an additional method getFile(), not defined in Part, which returns the java.io.File object for the file on disk. PartInMemory holds the contents of the uploaded file in a byte array in memory. Both PartOnDisk and PartInMemory are only temporary and will be cleaned up (deleted/dereferenced) at the end of the request. They will not persist after the request is serviced unless otherwise acted upon.

What this means is that a form constructed to post a file to any Cocoon page will result in that file being parsed and either kept in memory or saved automatically (though temporarily) to disk. Without this knowledge, the file upload sample shipping with Cocoon (here in Tomcat here in Jetty) seems confusingly simple to some - the code in the xsp does not upload the file (nor does any action as some may suspect) but merely lists information about it... In fact, the sample has nothing to do with xsp and should probably be moved.

It would be real nice if it could have an option to save the file permanently!!