Drag & Drop Artifact Creation

Background:
The goal of this project is to create a component that will allow a user to drag a file from their filesystem into a region of an HTML page, and then have the component assist them creating an Artifact record.

An Artifact record is a component of our product. It consists of a database record, along with a file. The files are currently stored on the application server's filesystem, but may someday be stored in the database. The fields in the record provide information about the file, and allow association with other types of records.

Implementation:
When the user drags a file over the HTML page, it should change the CSS settings for the target DOM element to show that it is a drop target. If the user drops the file within this DOM element, the component should capture the file information, and present to the user a Bootstrap Modal dialog containing an HTML form. When the user clicks the OK/Submit button for the form, the component should submit the form, including the file that was dropped. The server will respond, and the component should fire an event to indicate completion.
The component should replace the form in the modal dialog, with a progress meter, as soon as the form has been submitted. Once the server has returned, the progress meter shows complete, if the server returned an error, the error message should be displayed with the meter. It is the user's task to close the modal dialog when he/she is ready.

The form fields:
NAME — text input field
SUBJECT — text input field
DESCRIPTION — textarea s field

The dropped file should be added to the form for submission with the name value of SOURCE_FILE.

Once the user clicks OK, the form (with file) should be POST to the server with Content-Type of `multipart/form-data.`
The form should use Bootstrap basic form markup.

Only a single file may be associated with one form record. If the user attempts to drag more than one file, the component should display a notification that only one artifact record at a time may be created.

Following is an example of the form markup which includes a file input. The form should include this input, because the dialog should also be capable of being launched without the drag and drop initiation. It may be invoked programmatically from a button click event handler, for example. The file input need not be displayed if the invocation of the dialog is the result of a drag and drop operation, and the file is already captured by the component. The component should be capable of dynamically creating the required DOM structures for the

Bootstrap dialog if not provided (see below).

```
<form method="POST" accept-charset="utf-8" enctype="multipart/form-data"
      action="../request/artifact" data-formtype="artifact" >
  <input type="hidden" name="_charset_" id="_charset_" value="utf-8" />

  <div class="form-group">
    <label for="NAME">Name</label>
    <input type="text" class="form-control" id="NAME" name="NAME"
           placeholder="Name" autofocus="1″ required="‶required"/>
  </div>

  <div class="form-group">
    <label for="SUBJECT">Subject</label>
    <input type="text" class="form-control" id="SUBJECT" name="SUBJECT"
           placeholder="Subject" />
  </div>

  <div class="form-group">
    <label for="DESCRIPTION">Subject</label>
    <textarea class="form-control" id="DESCRIPTION" name="DESCRIPTION" >
</textarea>
  </div>

  <div class="form-group">
    <label for="SOURCE_FILE">File</label>
    <input type="file" id="SOURCE_FILE" name="SOURCE_FILE">
  </div>

  <button type="submit" class="btn btn-default">Submit</button>
</form>
```

The component may be implemented as a jQuery plugin. A DOM selector for the drop target element will be supplied. In addition, and configuration object may be supplied. The component should be capable of creating the Bootstrap Modal, and the form specified above without external requirements.

Example:

```
<script type="‶text/javascript″>

$('div.drop-target').artifactCreator({
     formURL : '',                 // optional
     formTemplate : '',            // optional
     modalTemplate : '',           // optional
     onComplete : function(){     // callback for server completion

    }
});


// A second example allowing the invocation without utilizing drag
and drop. This is simply a // programmatic invocation. The show
method should initiate the same sequence as the drop
// event, except that the form will have to contain an HTML input of
type "file" named
// SOURCE_FILE.

$('div.drop-target').artifactCreator({
     formURL : '',                // optional
     formTemplate : '',           // optional
     modalTemplate : '',          // optional
     onComplete : function(){     // callback for server completion

    }
}).show();



</script>
```

The formURL, formTemplate, modalTemplate, are all optional. If they are supplied the will be used to override the default form shown above. The form URL will be a relative URL to the origin server that will supply the markup. The formTemplate, and modalTemplate will be DOM selectors, or DOM elements within the existing page that contain the markup required.

Future Expansion:
The configuration object may hold JavaScript data for additional form fields. Or, it may contain a server side URL that will supply the entire HTML markup for the form. In any case the component will remain responsible for associating a file supplied through drag and drop, with the form for submission to the server.