Technical Design Document

IRS Static File Upload

Jan 17, 2017

Contents

Revision Record

Revision #	Date	Author(s)	Revision Notes
1	01/13/2017	Frank Granger	Initial template for release of Technical Design Document.
2	01/17/2017	Vaihbav Patel	Additional details specific to Static File Module.

1 Introduction

As Drupal developers, it is important to use contributed modules when available and develop custom modules when necessary. This source serves as the primary documentation for the latter, though significant enhancements to contributed modules will be documented in a similar manner.

Developers should be coding to the specifications outlined by Drupal here. In addition to this document, the backend developers will include some level of commenting within the code itself. In the event that upgrades, enhancements, other modules affect the compatibility of the module, this should be a detailed resource for production support.

This document is intended to provide programmers with an understanding of the technical details of the custom module and some reasoning behind why the developer chose to develop it in the way they did.

This document assumes that readers have the following pre-requisite skills:

- Experience with PHP application development
- Familiarity with Drupal API

This document will only briefly touch upon use cases, and the more functional aspects of the module as it pertains to special coding considerations. If looking for more detailed information to understand why a particular module was developed, please see the Functional Design Document.

1 Objective

Create a module which will allow the users to upload file to specific directory and replace any file which has the same name and location

2 Environment Description

Drupal 8.x and the supporting LAMP stack.

2.1 Drupal and PHP Version Requirements

This module was developed and tested on the Lightning Distribution of Drupal offered and supported by Acquia version 8.0.2. It is running PHP 5.5.33. For more information and details regarding these distributions please visit http://lightning.acquia.com/ and http://www.php.net/.

2.2 Dependent Modules

- Entity

This module extends the entity API of Drupal core in order to provide a unified way to deal with entities and their properties. Additionally, it provides an entity CRUD controller, which helps simplifying the creation of new entity types.

- Media Entity

Media entity provide a 'base' entity for media. This is a very basic entity which can reference to all kind of media objects(local files, YouTube, video, Tweets, Instagram photos, ...). Media entity provides a relation between Drupal and the media resource. You can referenced to/use this entity within any other Drupal entity.

- File

The File module enables you to upload and attach files to content and to manage these uploads if you have the appropriate permission. This module is responsible for validating file content and managing uploaded files. It also provides options for displaying file content.

- Media entity document

Local document integration for media entity module.

- PUP Custom Modules Base

This is the default container for all IRS PUP modules.

2.3 Configuration Setup

2.3.1 Fields

- field_pup_file_public_path

Maching name for static file path. It creates the directory based on select repository list.

- Field_document

Machine name for media bundle document.

Above both field required if you want to use this module in your form.

2.3.2 Taxonomy

- Add taxonomy vocabulary for hierarchical terms and referred it to field_pup_file_public_path

2.4 Automatic Processes

2.4.1 Cron Jobs

Automated Cron module is in core. so using this we can set our cron time for the remove temporary files.

3 Module Description

3.1 API

- File_prepare_directory

Check that the directory exists and is writable. Directories need to have execute permission to be considered a directory by FTP servers, etc.

- File_move

Moves a file to a new location and update the file’s database entry.

- loadAllParents($tid)

Finds all ancestors of a given term ID.

3.2 Functions

This module extends media functionality which is overrides/placed the file in the location selected by a user. Below are the description of the functionalities implemented in this module

3.2.1 Retrive a static file path

- When the user select a folder location, based on it folder path will appear.
- Function irsFileOperationsGetTax($termid) returns folder path based on the $termid selected.
- In this function, we have used Entity Manager to get storage of taxonomy term.
- We have loaded all parents based on $termid using loadAllParents function. We have used foreach for $parents and concatenated the string to generate the folder location path.

3.2.2 Upload

- When we upload a file, we needed the “URI” for the folder location path.
- Function irsFileOperationsUri($fid) returns URI based on fid passed to it.
- We have used the db select query to feth the $uri for a particular $fid.

3.2.3 Public/Private

- By default media is stored in the private folder when the media is unpublished. If the user published the media, It will move from private to public directory.
- Function moveFileToDestination($fid,$destination) takes care of this by checking the $fid and $destination parameters.
- In this function, we have used Entity Manager to get storage of file. We have extracted the $file_uri & $file_name based on the above data. $destination is used in file_prepare_directory which created the directory if it is does not exists. We have used the db updated query in the file_manages table to update the $url based on the destination. We have truncated the cache_entity table to reflect the $uri changes.

3.3 Error Handling

We have handled error below.

- File does not exist in the directory: if the file is not present in the location and we try to submit the media, it will throw the error.
- Directory does not exist: if the directory is not created or exists it will print the error with the return value for file_prepare_directory function.
- File not moved: If the file is not moved to destination directory it will throw the error with the return value of file_move function.