Drupal

The Inner-Workings

Ari Summer
CSCI 5448
Graduate Presentation
11/16/2012

What is Drupal?

- Drupal is an open source content management platform
- Allows you to easily build a website with many tools for customization
- It is similar to a framework in that:
  - Model-View-Controller architecture
  - User can create custom data types without creating database tables manually
  - User can query database any way they like using the Views Module
- Drupal is different from a framework in that it provides a user interface that saves the user from writing tons of code
- Trading Flexibility for Usability

Before we go any further...

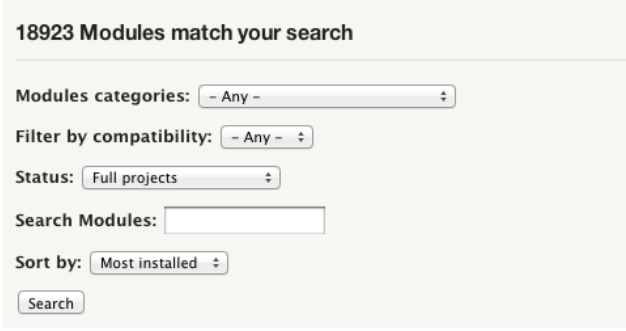
- Although Drupal does not extensively use PHP's class system, it utilizes many Object-Oriented fundamentals and techniques
  - At the time of implementation, PHP's class system was not mature enough for Drupal
    - This is changing with the introduction of PHP5. Drupal is beginning to use PHP5's class system more and more.
  - With the PHP class system the dynamic inclusion of files would have been much slower and/or included massive amounts of logic (Not Cool!)

Modular Design

- Drupal functionality comes from the installation of modules and themes for customization
  - Even much of the Drupal "core" is made from installed modules
- Drupal's core is isolated from third-party contributed modules and themes
- Extends Drupal's functionality without changing Drupal's core files
- Allows for clean upgrades and less problems when changing your web design

http://drupal.org/project/omega

http://drupal.org/project/Modules



Module Examples

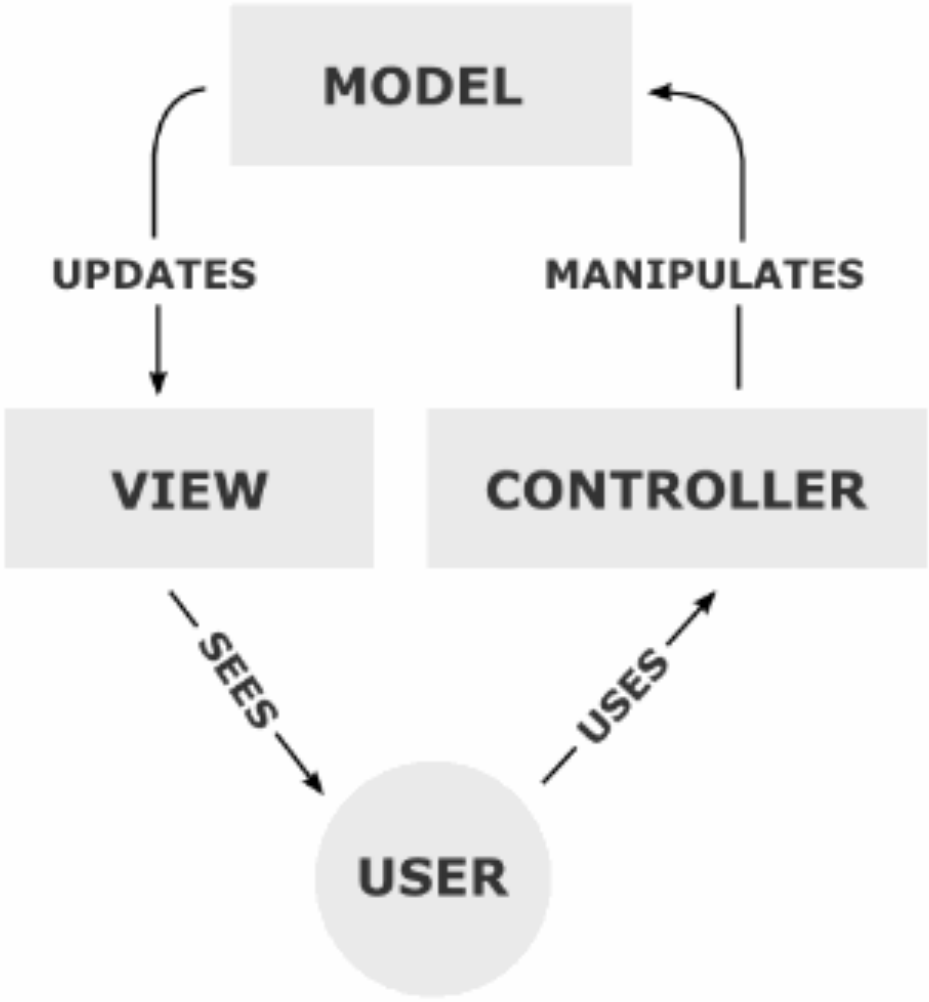
- Drupal for Facebook: "This set of modules turns Drupal into a platform for developing Facebook Applications. This allows you to embed your content and features within facebook, or allow facebook users onto your site via Facebook Connect."
- Quick Tabs: "The Quick Tabs module allows you to create blocks of tabbed content, specifically views, blocks, nodes* and other quicktabs*. You can create a block on your site containing multiple tabs with corresponding content. "
- Backup & Migrate: "This allows you to easily dump the sites database minus cache tables which is great for migrating the site across environments. It is also great for scheduled backups that run on cron runs."
- Flag: "Using this module, the site administrator can provide any number of flags for nodes, comments, users, and any other type of entity. Some possibilities include bookmarks, marking important, friends, or flag as offensive..."

Model-View-Controller (MVC)

- Drupal uses the MVC architecture when using the Views Module in order to display data and react to changes.
  - Similar to that used in Ruby on Rails
- Some people say Drupal is PAC and there seems to a be a lot of confusion about the difference between MVC and PAC.
- Larry Garfield has a detailed explanation about why he thinks Drupal is a PAC architecture, but this was written in 2006.
- With the newer Views Module, it is arguable that Drupal's architecture is closer to MVC according to the classic definition.
- Some people seem to think that in MVC, the view can speak directly with the model and others do not.
- The terms MVC and PAC are often times loosely used, but the main point is that the **model, view, and controller are separate** from each other creating a modular design

MVC Architecture

- Either way...
  - Model: Represents data and rules to manipulate that data
  - View: Queries database and displays data
  - Controller: Notifies View and Model of changes
- In Drupal:
  - Model: Database and Nodes
  - View: Theme and Views Module
  - Controller: Web Browser and Menu System



http://en.wikipedia.org/wiki/File:MVC-Process.png

MVC vs PAC

- MVC and PAC are similar but there is a distinct difference
- In MVC, the View is allowed to talk to the Model in order to retrieve information
- In PAC, the Presentation (View in MVC) cannot talk directly to the Abstraction (Model in MVC).
  - The Presentation only speaks directly to the Controller as does the Abstraction
  - Therefore, the Controller acts as the middleman

MVC in Drupal

- The menu system acts as the **Controller**
- Data is stored in "**nodes**" which is the **Model**
- The View consists of the **Theme** system and its associated **Views** and **Blocks**
- Views and Blocks can easily be created using the interface without writing any code

Views

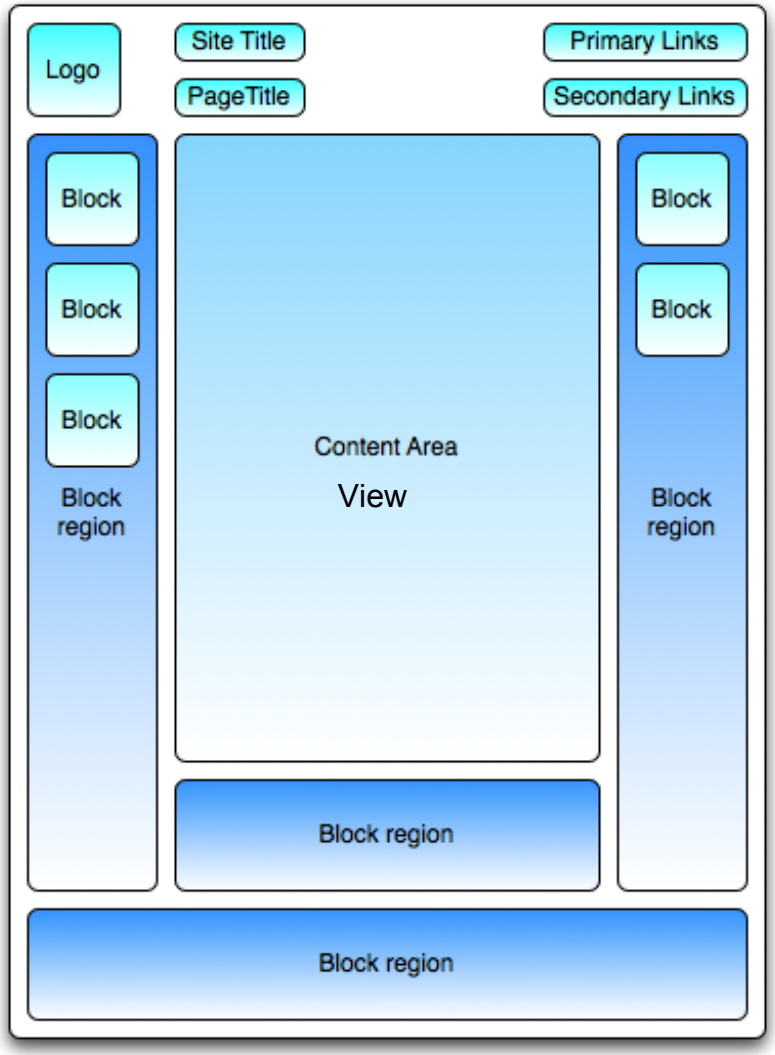
- View - Some presentation of content
  - This is usually used to display the main content on a page such as Blog Posts
- The Views Module is a very popular module and allows you to create Views with a user interface
  - This module's output is SQL-queries which pull info from the database to display the desired content to the user
- Views contain Fields, Filters, Sort Criteria, Relationships, etc. to sort and display data.
- Views can be displayed in Page or Block form
- **Page Views** are assigned a URL paths and are usually the primary content of a page
- Blocks will be discussed on the next slide

Blocks

- Blocks can be generated by using the Views Module, writing your own HTML, or installing other modules that come with pre-made blocks
- These usually consist of secondary content and are often on many pages on the website
- Blocks system provides a user interface that allows you to place blocks in specific areas based on your theme
- Examples: Facebook Like Box, Menus, Login

Image: http://www.bugtreat.com/blog/how-to-add-a-facebook-like-box-to-your-blog/

Page Layout



http://drupalwatchdog.com/1/2/stacking-up-drupal

Nodes

- **Nodes** are essentially objects that contain data
  - Methods for Nodes are contained in node.module
  - Store most content on a Drupal site
- **Nodes** can be of different **Content Types** such as a Page, Blog Post, story, article, etc.
- It contains information or "fields" such as the Author, Date, Title, and Description.
- Node module allows you to list, sort, manage, and configure Content Types.
- The Content Construction Kit (CCK) Module allows for customization of Content Types

User "Objects"

- Users are also objects in Drupal
- These objects contain data such as profile information, session tracking, and privileges.

```
stdClass Object

(
        [uid] =>
        [name] =>
        [pass] =>
        [mail] =>
        [mode] => 0
        [sort] => 0
        [threshold] => 0
        [theme] =>
        [signature] =>
        [signature_format] => 0
        [created] => 1268923269
        [access] => 1279119654
        [login] => 1278690259
        ...}
```

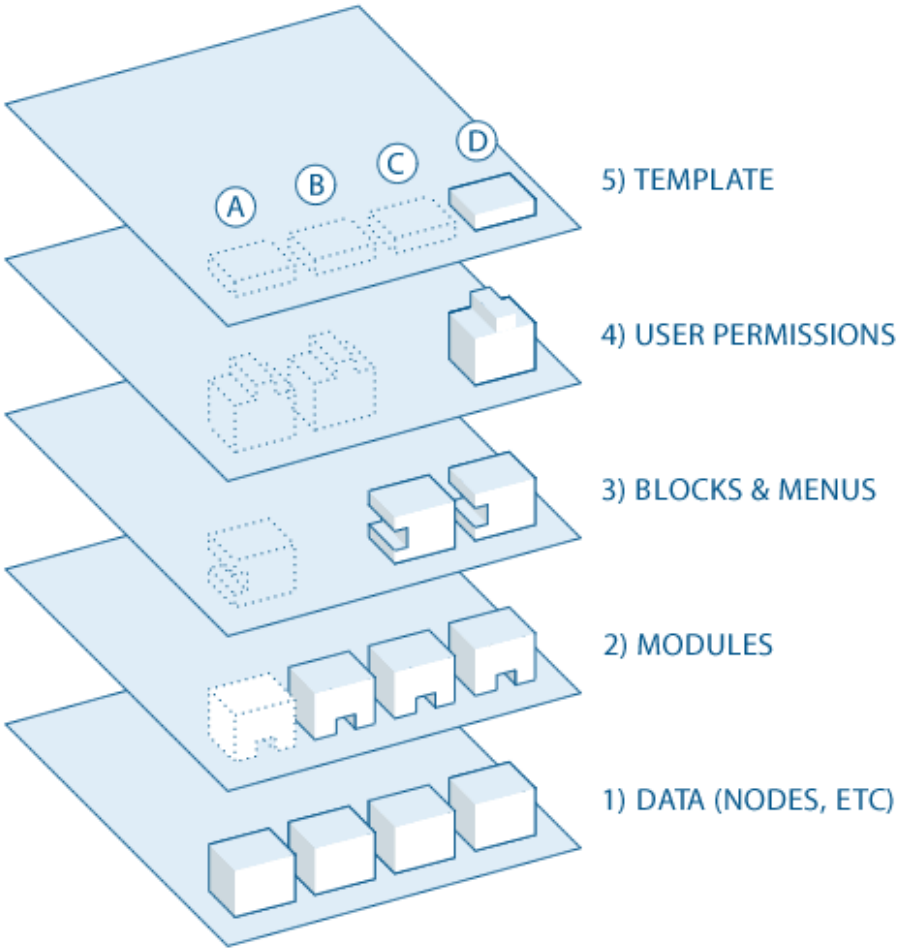
```
if ($user->uid == 0) {
  //user is not logged in
}
```

- uid - User ID
- name - User name
- pass - Encrypted password
- mail - User current email address
- theme - Name of the theme shown for this user (no longer changeable in core but can be in contrib)
- signature - Signature as set in the user account settings
- signature_format - Text format to apply to signature
- created - Unix timstamp for when the account was created
- access - Unix timstamp of the last time the user accessed the site
- login - Unix timstamp of the last successful login by this user
- status - 1 if the user is active, 0 if blocked
- timezone - User's timezone as a PHP compatible timezone string (date_default_timezone_set())
- language - User's language code
- picture - User picture / avatar
- init - Contains the email address the user provided during initial registration
- data - Data stored in the users table by contrib modules (second argument of user_save())

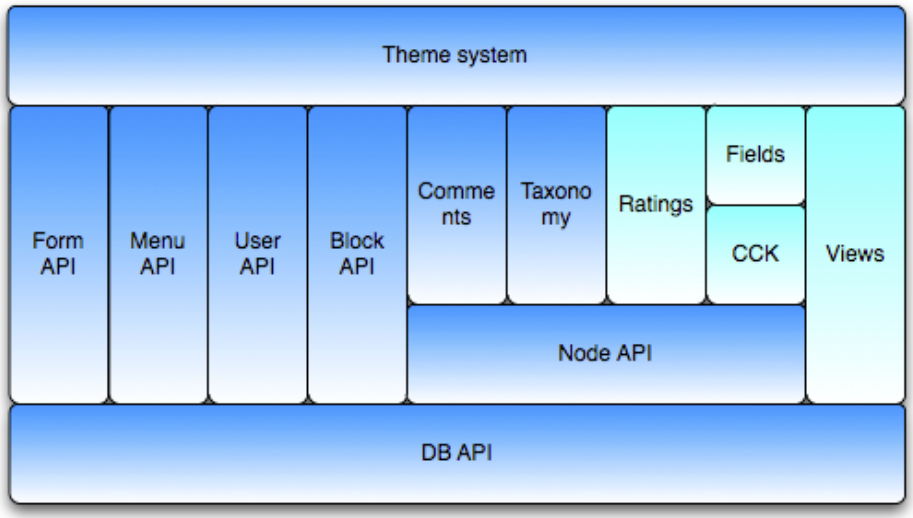
Other Objects

- From a Users perspective, modules and themes can also be thought of as Objects
- Modules contain functions or methods and utilize the hook system to perform tasks and pass messages

Visual Representations



http://drupal.org/getting-started/before/overview



http://www.palantir.net/sites/default/files/general/images/matw-1_0.png

Common OO Fundamentals in Drupal

- Drupal uses many OO fundamentals in its design
  - Abstraction
  - Encapsulation
  - Polymorphism
  - Inheritance
- It also uses many Design Patterns defined by the Gang of Four
  - Decorator
  - Observer
  - Bridge
  - Chain of Responsibility
  - Command

Abstraction

- The hook system in Drupal provides the abstraction
- Contract: when a module implements a certain hook, it will perform a specific type of task
- Caller does not need to know anything about callee or the way in which the hook is implemented during invocation

Hooks

- Module system is built on "hooks" which are PHP functions
- In order for Drupal to utilize modules and for modules to call other modules, they implement hooks.
- When an action occurs where a hook is used, all modules implementing this hook get called.

```
function nice_try(){
  drupal_set_message(t('Nice try, sucker'),
$type = ‘error’); // display this message
  drupal_goto();  // reload the current page
}
```

Encapsulation

- In order to provide protection, Drupal relies on convention.
- Private methods inside modules are prefixed by an underscore. Public methods have no underscore.
- There is no strict implementation in the Drupal system that limits access to data within some object. *Yikes!*
- In my opinion, this is a weakness of Drupal, but that is up for debate

Polymorphism

- Nodes in Drupal are very much polymorphic
- When methods are called on Nodes, we can treat them the same way, but get the behavior appropriate to each specific node.
- For example, node_build() follow by drupal_render() will give an HTML representation of a Node
  - The implementation of the rendering may vary with the type of Node.

Polymorphism Continued

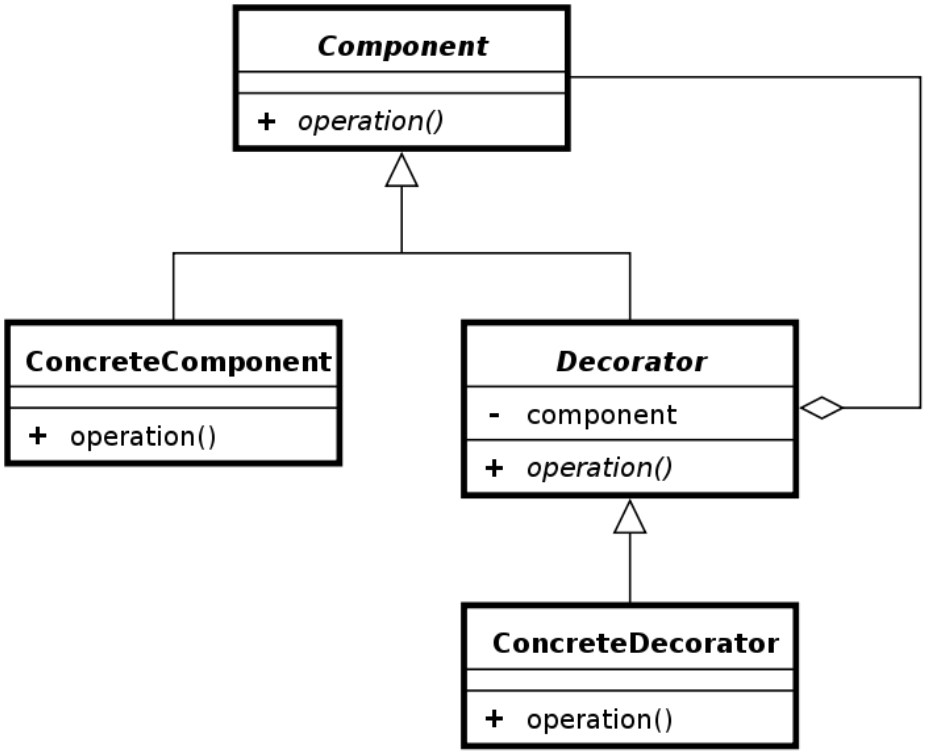
- Themes are also polymorphic.
- Each Theme can be told to render and display a node using the same interface and methods
- However, a specific theme will render a node differently depending on the Theme implementation.

Inheritance

- Modules and Themes are essentially classes that inherit behavior from their abstract base classes
- They can override the default behavior of the base class and add additional behavior
- Themes: can override default renderings
- Modules: can override hooks and decide which ones to implement

Design Patterns: Decorator

- This pattern allows for an object's functionality to be extended at run-time
- Using the hook system, a module can extend another modules functionality
- No subclassing necessary
- Example: Upload module could add an image to a node module



http://en.wikipedia.org/wiki/File:Decorator_UML_class_diagram.svg

Design Patterns: Observer

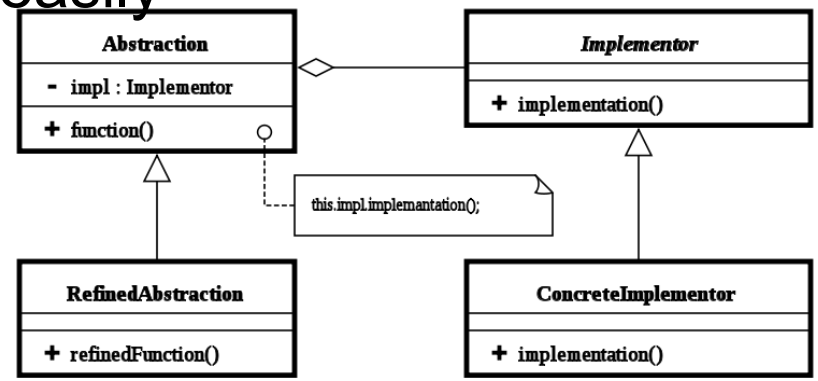
- The Observer pattern allows Objects to become Observers of Subjects on which they depend
  - Allows for Objects to update their state when necessary
- The Observer Design Pattern exists within the hook system.
- By implementing a hook, a module will become an observer of another object.
- When a change occurs in the subject object that is related to a hook, the observer implementing the hook will be notified.
- Examples: hook_node_load(), hooke_node_login()

Design Patterns: Visitor

- Similar idea as Observer but provides operations to perform on subject
- "The Visitor pattern represents an operation to be performed on the elements of an object structure without changing the classes on which it operates"
- For example, hook_node_view() allows modules to perform operations on a node to view it, but does not change the node class on which it operates.

Design Patterns: Bridge

- Drupal uses a Database Abstraction Layer that utilizes a design similar the the Bridge Pattern.
- When getting information, Modules use this abstraction layer.
- They do not query directly on the database.
- This allows Modules to be independent of the database system implementation
- New databases can be added easily



http://en.wikipedia.org/wiki/Bridge_pattern

Design Patterns: Factory

- In addition to providing a common Database Abstraction by using the Bridge pattern, the Factory Pattern is used to return the correct Object for a given Database without the user knowing
- For example, when calling an Insert Query command using db_insert, caller does not decide whether InsertQuery_mysql or InsertyQuery_pgsql
  - This is Factory's job

Design Patterns: Command

- The Command Pattern encapsulates all data needed to execute operations at any time within an Object
- The InsertQuery object is an example of the Command Pattern
  - Contains the operations executed on the Database within InsertQuery
- Increases usability and flexibility
  - Can be saved for later execution
  - Hooks can edit the query easily before executing
- Drupal could use this for undo operations but does not currently

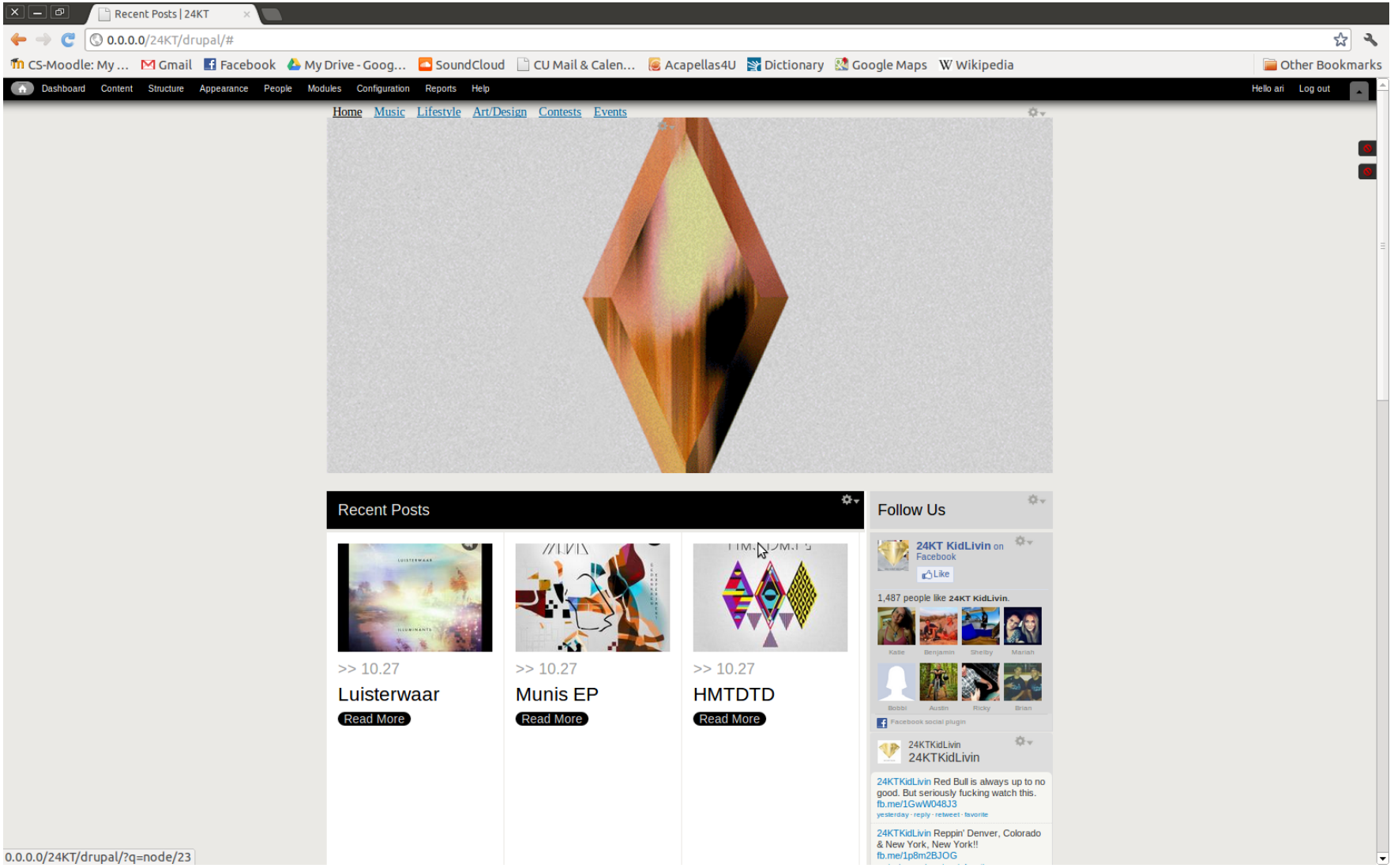
Design Patterns: Chain of Responsibility

- Objects can send a command up a "chain" without knowing what object(s) will handle it
  - Objects handle the request, pass it on, or both
- When the menu system receives a page request, it passes this request up a chain to determine how to handle this request and what modules will be called.

Conclusion

- Drupal is a great system for getting websites up and running fast
- It is still being developed and improvements are being made
- Drupal is a large complicated system and much is abstracted away from the user
  - It is often difficult to know what exactly is going on under the hood
  - Trades flexibility for usability
- Drupal is its own animal and it is often difficult to place a name on some of the design principles used
  - Many of them are different than the classical sense

My first website using Drupal...



Works Cited

- http://www.garfieldtech.com/blog/mvc-vs-pac
- http://views-help.doc.logrus.com/help/views/about
- http://inspiredm.com/drupal-terminology-nodes-blocks-and-views-oh-my%E2%80%A6/
- http://drupal.org/documentation/modules/node
- http://api.drupal.org/api/drupal/developer%21globals.php/global/user/7
- http://api.drupal.org/api/drupal/includes%21module.inc/group/hooks/7
- http://www.mccormickandwinter.com/blog/drupal-hooks-explained
- http://en.wikipedia.org/wiki/Decorator_pattern
- http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
- http://www.oodesign.com/decorator-pattern.html
- http://www.oodesign.com/command-pattern.html
- http://en.wikipedia.org/wiki/Command_pattern
- http://www.oodesign.com/chain-of-responsibility-pattern.html
- http://sourcemaking.com/design_patterns/visitor
- http://drupalwatchdog.com/1/1/design-patterns-of-drupal