Architecting Drupal Modules

Report from the Frontlines

bluespark

Ronald Ashri

Directory of Technology

ISTOS

drupalrooms
for hotels, villas and b&bs

actSMART

t: @ronald_istos

bluespark

building modules

path not always clear

http://www.flickr.com/photos/hesketh/2249658529

bluespark

example 1

I want my module to use a javascript library

http://drupal.org/node/304255

Community Documentation

Community Docs Home | Installation Guide | Administration Guide

4. Adding JavaScript to your theme or modul

The PHP function drupal_add_js() lets you add a JavaScr setting or inline code to the page and it takes 5 parame the api reference).

http://drupal.org/node/304255

Community Documentation

Community Docs Home | Installation Guide | Administration Guide

Managing JavaScript in Drupal 7

Adding JavaScript in the module's .info file

You can now add Javascript in the module's .info file if it should be added on every page. This allows Javascript to be aggregated in an optimal way, and is the preferred method of adding Javascript that most visitors will need on a typical site visit:

```
scripts[] = somescript.js
```

External JavaScript

drupal_add_js() now allows you to add external scripts.
Example:

```
drupal_add_js('http://example.com/example.js', 'external');
```

example 1

I want my module to use a javascript library

hook_library

7 system.api.php hook_library()

Registers JavaScript/CSS libraries associated with a modul

Modules implementing this return an array of arrays. The
readable name of the library. Each library may contain the

- **'title'**: The human readable name of the library.
- **'website'**: The URL of the library's web site.
- **'version'**: A string specifying the version of the library
version like "1.2.3" is not a valid float. Use PHP's versi
versions.
- **'js'**: An array of JavaScript elements; each element's k
element's value is used as $options array for drupal_a
module-specific) JavaScript settings, the key may be s
=> 'setting', and the actual settings must be containe
- **'css'**: Like 'js', an array of CSS elements passed to dru
- **'dependencies'**: An array of libraries that are required
listing the module and name of another library. Note
dependent library will also be added when this library

Libraries API

View | Version control | Revisions | Automated Testing

Posted by **sun** on *May 18, 2009 at 12:51pm*

The common denominator for all Drupal modules/profiles/themes that integrate with external libraries.

This module introduces a common repository for libraries in `sites/all/libraries` resp. `sites/<domain>/libraries` for contributed modules.

External libraries

> Denotes libraries (*"plugins"*) that are neither shipped nor packaged with a project on drupal.org. We do not want to host third-party libraries on drupal.org for a multitude of reasons, starting with licensing, proceeding to different release cycles, and not necessarily ending with fatal errors due to conflicts of having the same library installed in multiple versions.

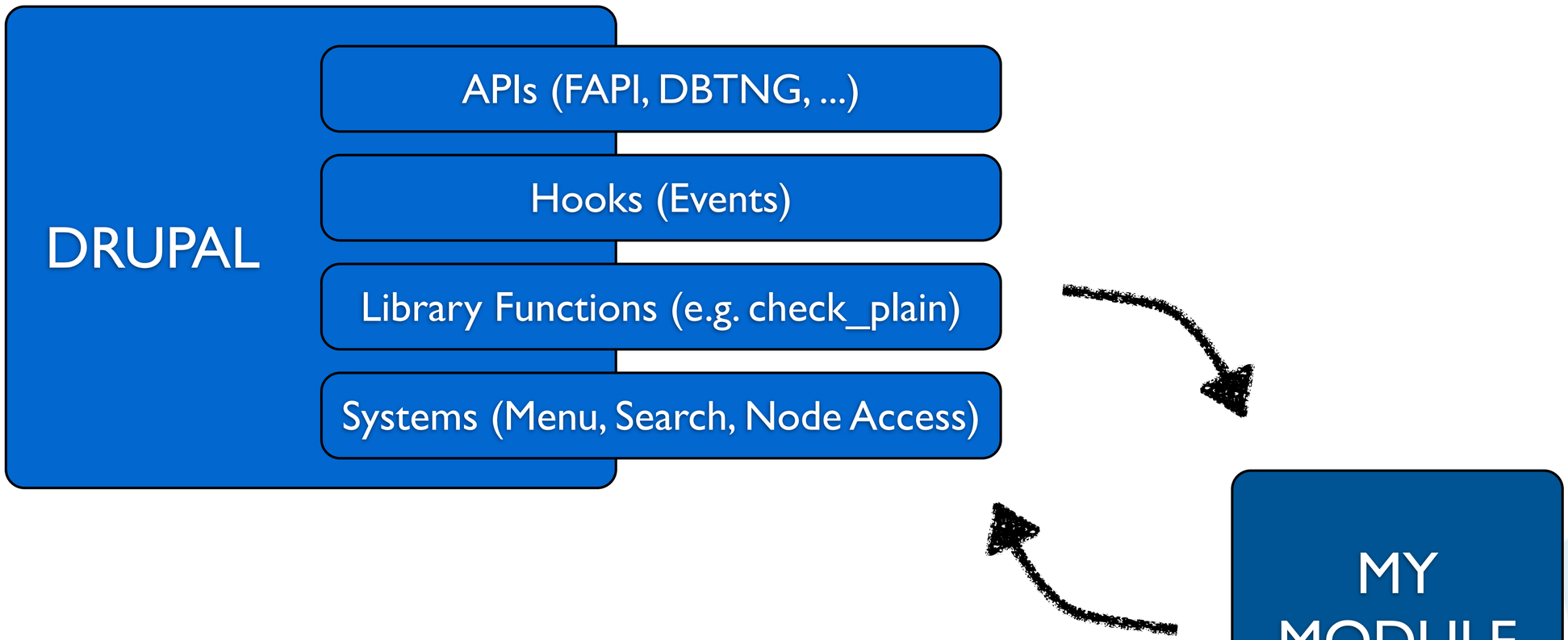
Drupal 7 only has built-in support for non-external libraries via hook_library(). But it is only suitable for drupal.org projects that bundle their own library; i.e., the module author is the creator and vendor of the library. Libraries API should be used for externally developed and distributed libraries. A simple example would be a third-party jQuery plugin.

http://www.flickr.com/photos/42302655@N02/4064741545

bluespark

let's take a step back

what are we trying to do here



bluespark

Building Software

Methodology, Framework, Patterns, Architecture

Methodology

A set of guidelines / processes that accompany you from problem definition to solution

Framework

Provides reusable elements and underlying structure you plug into

Patterns

Proven reusable solutions

Your Architecture

Elements and relationships between them

Building Drupal Modules

Methodology, Framework, Patterns, Architecture

Methodology?

A set of guidelines / processes that accompany you from problem definition to solution

Drupal

Framework

Provides reusable elements and underlying structure you plug into

Patterns?

Proven reusable solutions

Your Drupal Module Architecture

Elements and relationships

example 2

how should I structure my files?

- ☐ everything in the root of your module directory
- ☐ .module, .info in root, images in images, js in js, inc in includes, views in views
- ☐ some in root, some in directories based on history, module evolution, style changes, etc

example 3

how should I name my files

- [ ] .module, .info, .inc, .install, .test
- [ ] occasionaly throw in: .theme, .install.inc
- [ ] sometimes: .<modulename>.<thing>.inc, ClassName.inc
- [ ] myconventionisbetter.me
- [ ] no! MyConventionIsBetter.class.inc

example 4
I need to store data

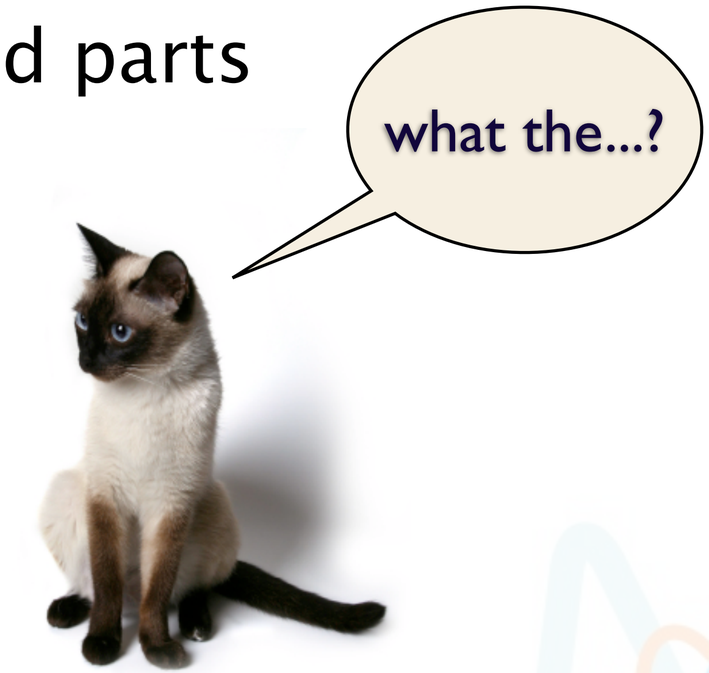
- ☐ my table via hook_schema + DBTNG
- ☐ entities are the way to go – always! (maybe?)
- ☐ fields, then a fieldgroup entity (or node) to add what you are missing
- ☐ is your data content or configuration?

Guidelines

what is important in general

Drupal is a complex system of many interlinked parts

There are always many ways to skin a cat...



It’s not about the recipes
– it’s about the principles

Guidelines + Patterns

Guidelines

what is important in general

- [ ] **Separation of concerns** – (e.g. logic in modules, presentation that can fully by managed by themes, flexible admin)

- [ ] **Decoupled** – more smaller modules that incrementally add functionality, OO where possible

- [ ] **Consistent** – similar things always happen and are described in the same way

Define Problem and Design

not in a Drupal specific way

- You module solves a problem – you should be able to describe that in generic terms.
  - A hotel owner needs to be able to display a list of available rooms with their associated descriptions given an arrival and departure date
  - Vs: I need to get all bookable unit entities and attach a field entity reference to them pointing to Room Description nodes that I can then render in Rooms view mode

Define Problem and Design

not in a Drupal specific way

- Describe your architecture in generic terms first and then in specific Drupal terms
  - Allows you to focus on what's important and not get distracted by how Drupal does things
  - Enables you to better choose what Drupal way to use subsequently

Patterns - OO Patterns

Entity API, Subdomain

- Entity API offers interfaces and implementation of those interfaces as well as helper “procedural” style functions
- Subdomain offers all of its core functionality via a class - you can extend/replace via your own module
  - Allows us to plug into “known” generic ways of doing things - reduces the burden of Drupal to define new styles

Patterns – Drupal Patterns

Views, Commerce

- Separate UI from core module functionality
  - Allows us to focus on each and replace
  - Form submit handlers, etc should use you “core engine” functions – avoid stuffing a lot of logic there
  - Can switch UI off for performace gains

Patterns – Service Patterns

Rooms

- Where possible decouple interaction points within same module
  - Rooms produces all availability data in JSON following a callback
  - Allows us to use any number of display techinques such as the FullCalendar JS library
  - Can easily abstract and connect to non-Drupal site

in conclusion

- ☐ need to work on methodology
- ☐ start collecting and documenting patterns
- ☐ discuss conventions
- ☐ talk to me if interested – @ronald_istos