Building Applications with OpenInteract2

Chris Winters

Optiron Corporation

June 17, 2004

What is OI2?

OpenInteract2 is a database and presentation independent application server that makes it easy to build and distribute standalone applications.

Major rewrite

OpenInteract2 is a major rewrite of 1.x (currently 1.61)

- OpenInteract 1.05 first released to CPAN Feb 2001
- Some design decisions were... less than optimal
- Lots of direct hash references vs. method calls
- Tied closely to Apache/mod_perl
- Not a full rewrite, most core concepts have analog
- Poor overall documentation
- No tests

Features of OpenInteract2

- Database independent
- Built-in object-relational mapping
- Presentation independent
- Authentication
- Authorization (application and data)

Features of OpenInteract2 (ctd)

- Internationalization
- Simple framework for creating management tasks
- Runs under multiple platforms (Apache 1.x, Apache 2.x, LWP, CGI...)
- Also works with LDAP
- Extensible architecture: not just tied to web
- Lots of patterns!

Database independent?

Yes, really. DBI makes it all possible:

- Usual suspects: PostgreSQL, MySQL, SQLite
- Commercial folks: Microsoft SQL Server, Oracle, Sybase ASE, Sybase ASA
- Even DB2 on an AS400!

Presentation independent?

Yes, really. Currently we support three CPAN modules, but it's easy to write your own adapter.

- Template Toolkit (strongly preferred)
- HTML::Template
- Text::Template

vs other frameworks

- OI2 is fairly heavyweight and may be intimidating
- Especially versus something like Maypole
- Not as many users (or books) as Mason
- Has lots of dependencies – you want us to rewrite them?

Create an Application

You're here to learn how to create applications, so let's start building!

Hello World, but interesting

- We'll start by developing a Hello World application
- Well, maybe a **Hello Mongers** application
- It will have templates
- ...using different template technologies!
- It will have database access ...what do you expect for a half-hour?

Distributable Hello Mongers

An application in OI is a package. Packages are distributable, can install themselves to a supported database, and are customizable per site.

Using the Standard Tools

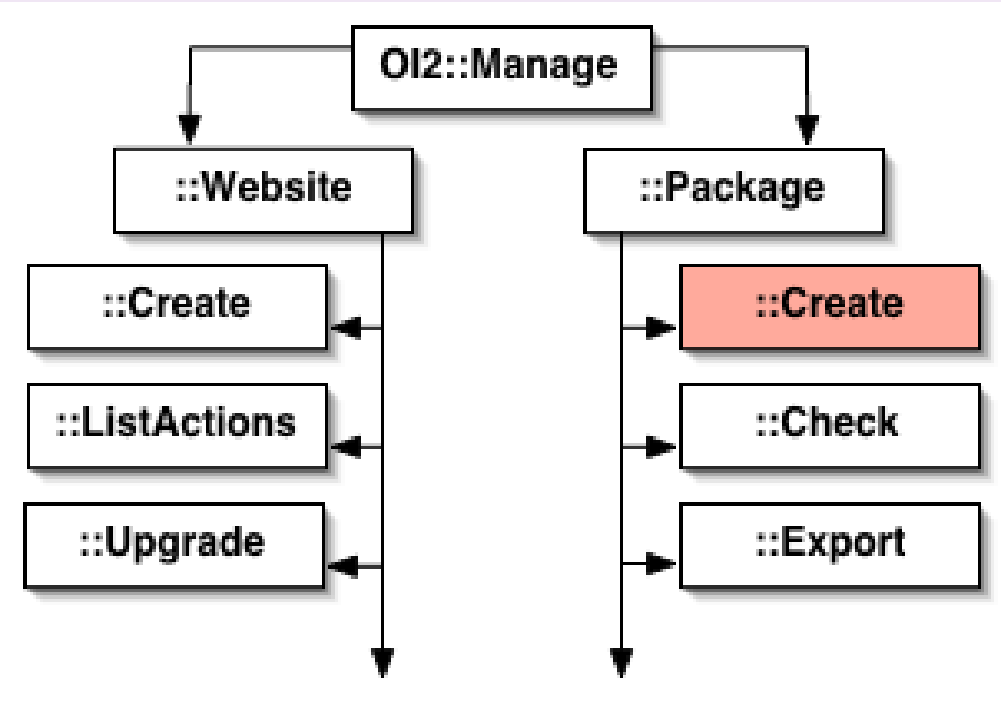
- Nothing up my sleeve
- Using standard unixy stuff (browser, terminal window, `bash`)
- OI2 tools you get with the distribution (`oi2_manage`)

Using a Nonstandard Tool

- ...and a funky keyboard

Create the Package

Using the standard `oi2_manage` tool we can hook into a framework of management tasks.



Running 'create_package'

Each task in the `OpenInteract2::Manage` hierarchy has a name, and you can use `oi2_manage` to run them.

- To create a package, run `oi2_manage create_package`
- Needs `--package` - name of package
- Also `--source_dir` - with files to create the package

An Act of Creation

First, we'll create the package skeleton

What does that include?

OI creates lots of seed files for us:

- Persistent object configuration
- Action configuration
- Action class
- SQL installer class
- Sample template
- Package-level documentation
- Package metadata (MANIFEST, changelog, description, dependencies, etc.)

Modify a Template

Let's start with the easy stuff:, modifying the system-generated template

Meet the templates

Our template engine of choice: Template Toolkit

- Easily handles objects and subroutines
- Simple to create plugins
- Flexible framework, step in where you want
- ...oh yeah, it does simple datastructures too

Change the template

So let's change the template

Some package sanity checks

OI comes with tools to do some basic package checks:

- Do your classes compile?
- Does your MANIFEST match?
- Are your configurations valid?
- Are your data files syntactically correct?
- Do your templates pass a syntax check?

Running 'check_package'

Checking a package is also part of the management framework.

- To check a package, run `oi2_manage check_package`
- Run it in the package directory, you don't need anything else

Check the package

Let's run the checks

Running 'export_package'

To be distributable we need a package to be in a single file.

- OI2 can produce and consume packages in ZIP files
- To export a package, run `oi2_manage export_package`
- Run it in the package directory, you don't need anything else

Export the package

Run the export

Deploy the package

OpenInteract's deployment paradigm is the site. Every site is independent of every other site.

- A site includes configuration, docs, logs, uploads...
- ...and packages
- To install a package guess what we use?

Running 'install_package'

Installing a package is also part of the management framework.

- To install a package, run `oi2_manage install_package`
- Pass it a `--package_file` with the exported zip file
- Pass it a `--website_dir` with the site directory
- ...or use `$ENV{OPENINTERACT2}` (preferred, less typing)

Install the package

Install the package to the site

How do we know it works?

Once we restart the server we should have a new action 'mongers' with two tasks, 'hello' and 'list'

Invoke the package

See if it's working by hitting its actions and tasks:

- (more later on what 'action' and 'task' are)
- `http://localhost/mongers/`
- `http://localhost/mongers/list/`
- `http://localhost/mongers/hello/`

Package fingerprints on the server

- Ask the server about the packages it contains using `oi2_manage list_packages`
- Look into the package repository for the current version
- Scan the filesystem for directories under `pkg/`

Code an Action

An action gets invoked by a URL or programmatically by looking up an action by name.

Action Overview

An action comprises two parts:

- A declaration in a configuration file
- The code specified by the declaration
- Both are shipped with package ...but it's possible to override configuration per site

Configuring an action

Most of OI2 uses a modified INI format. Each action is a section and declares a properties to the framework. Example:

```
[mongers]
class = OpenInteract2::Action::Mongers
task_default = list
is_secure = no
```

Mapping URL to class and method

Actions subclass `OpenInteract2::Action` and by default each task maps to a method. So with our config:

`http://localhost/mongers/list/`

maps to:

```
Class:  mongers => OpenInteract2::Action::Mongers
Method:  list => list()
```

Coding a new task

We'll add a new task, 'display'

Redeploy the quick way...

Instead of going through the check/export/install routine from earlier, we can use a combination management task `package_all`...

Modify Content Generator

It's easy – honest! – to use different templating systems.

Multiple Configurations, One Action

You can point multiple actions at a single class.
Their parameters and properties will be different as reflected in the created action object.
Action property examples:

- content generators (TT, HTML::Template, ...)
- url
- default task
- security requirements

Another configuration, another engine

Here we'll specify a different content generator but point to the same source:

```
[mongers]
class = OpenInteract2::Action::Mongers
task_default = list
is_secure = no

[pgh_mongers]
class = OpenInteract2::Action::Mongers
task_default = list
is_secure = no
content_generator = HTMLTemplate
```

Specifying templates in configuration

Since we have two content generators using the same task we cannot tie the task and its template together.
So specify in configuration instead:

```
...
[mongers template_source]
display = mongers::display
...
[pgh_mongers template_source]
display = mongers::display_htmlt
```

Deploy and view

Deploy the new package and view

Use Objects from Database

We'll create a table and read data from it, modifying our existing 'display' action.

Steps to use database object

1. Create the structures (tables/sequences)
2. Create initial seed data (optional)
3. Tell OI2 about them for installation (SQLInstall class)
4. Configure SPOPS object
5. Modify action to use database

Create your structures

- OI2 has plumbing to install tables and sequences
- Create the table in `struct/monger.sql`

```
CREATE TABLE monger (
  monger_id     %%INCREMENT%%,
  first         varchar(20) not null,
  last          varchar(20) not null,
  primary key( monger_id )
)
```

Create seed data

- OI2 takes advantage of SPOPS import capabilities
- We can declare objects to import at install time

Create `data/initial_data.dat`

Modify SQLInstall class

Modify the OI2-created `OpenInteract2::SQLInstall::Mongers`

Configure SPOPS object

Modify the OI2-created `conf/spops.ini`

Modify the action code

Change 'display' task to use database

Modify the templates

We don't need to modify the templates!

Deploy with a change

- We'll deploy the new package as before
- But also run a new `oi2_manage` task: `install_sql`
- This will install all tables, initial data and security for a package

Install structures and data

Install the SQL structures and data

See how it works

- We should now see the data from the database
- ...in both the TT and HTML::Template pages

For more information

OpenInteract Home Page
http://www.openinteract.org/

Current docs
http://www.openinteract.org/docs/oi2-snapshot/

Chris Winters
Optiron Corporation
chris@cwinters.com
http://www.cwinters.com/