# Building Applications with OpenInteract2

Chris Winters

Optiron Corporation

June 17, 2004

# What you'll know when we're done

When we're done here you should:

- be able to create an application
- have a good familiarity with OI2 architecture
- ...at least enough to do what you want

# How we'll do it

Barring digressions and overflows, each part should match a session/break time:

- Part 1: Quick intro, first app (Hello Mongers)
- Part 2: Add flourishes to first app; quick architecture overview
- Part 3: Start our main application

# How we'll do it (parts 4-6)

- Part 4: Continue implementing main app
- Part 5: Advanced: add comments, object behavior, observers

# How we'll do it (parts 4-6)

- Part 4: Continue implementing main app
- Part 5: Advanced: add comments, object behavior, observers
- (if we get there)
- Part 6: Advanced: add boxes, datasources, multi-table searching

# Some quick background

Obligatory intro:

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

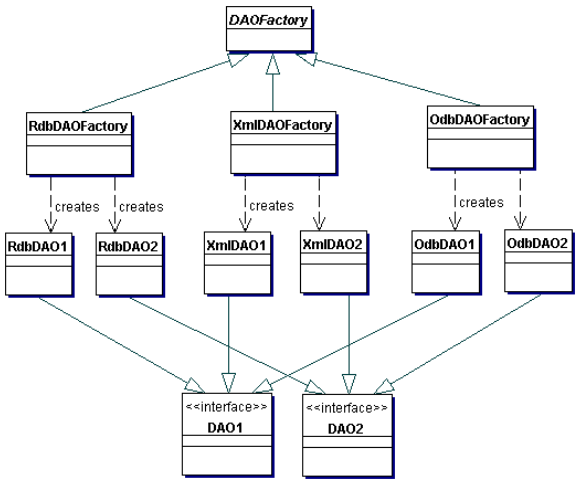
# Major rewrite

- OpenInteract2 is a major rewrite of 1.x (currently 1.61)
- OpenInteract 1.05 first released to CPAN Feb 2001
- Some suboptimal design decisions
- Tied closely to Apache/mod_perl
- Overall + specific docs lacking
- No tests

# Features of OpenInteract2

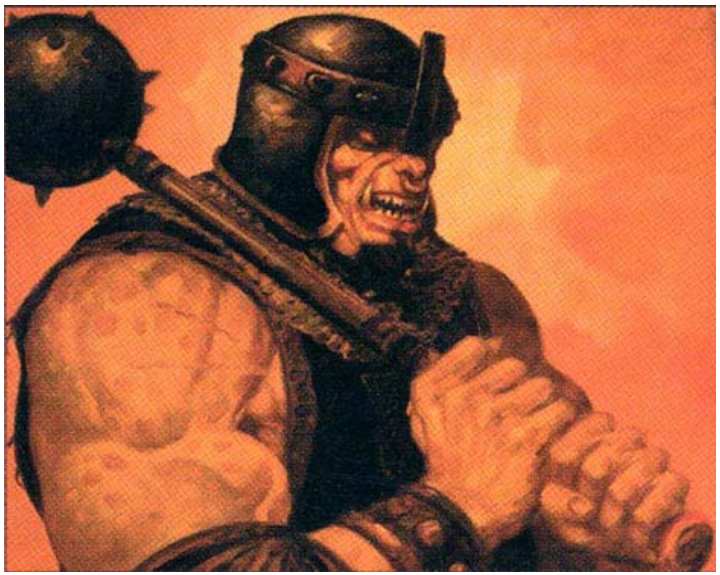
- Database independent
- Built-in object-relational mapping
- Presentation independent
- Authentication
- Authorization (application and data)
- Also works with LDAP

# More features of OpenInteract2



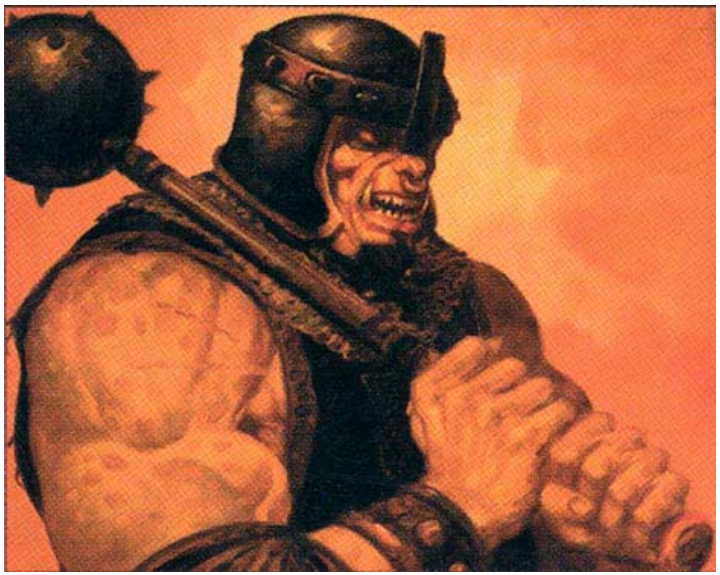

- Internationalization
- Simple framework for creating management tasks
- Runs on multiple platforms (mod_perl, LWP, CGI...)
- Extensible architecture: not just tied to web
- Lots of patterns!

# vs other frameworks



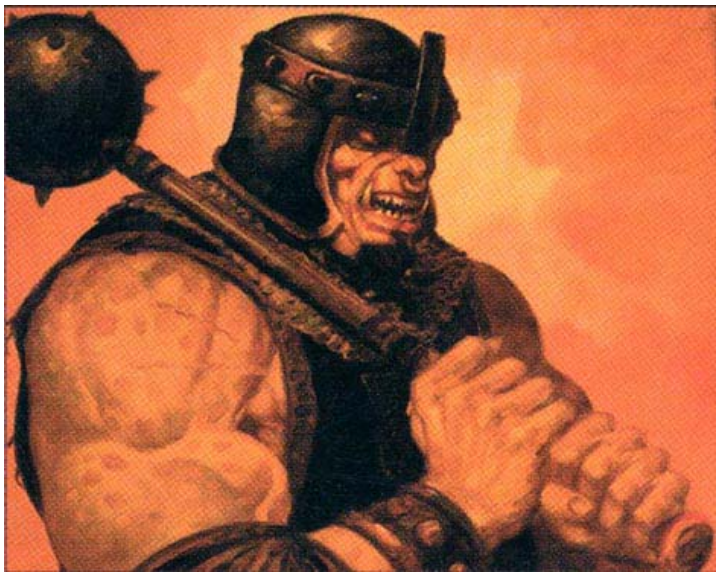
- OI2 is fairly heavyweight and may be intimidating

# vs other frameworks



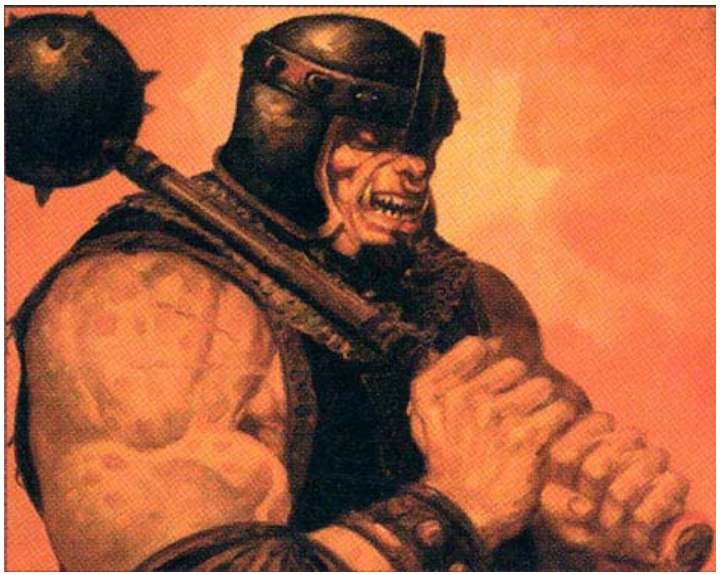
- OI2 is fairly heavyweight and may be intimidating
- (...but not after this tutorial!)

# vs other frameworks



- OI2 is fairly heavyweight and may be intimidating
- (...but not after this tutorial!)
- Especially versus something like Maypole

# vs other frameworks



- OI2 is fairly heavyweight and may be intimidating
- (...but not after this tutorial!)
- Especially versus something like Maypole
- Not as many users (or books) as Mason

# Create an Application

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

# "Hello Mongers"

- We'll start by developing a **Hello Mongers** application
- Simple: map URL to action, show templates, get data
- Get used to some core OI2 concepts

# Using the Standard Tools

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

# Using a Nonstandard Tool

...and a funky keyboard

# But first...

A query for the audience

# Create the Package

An OI2 package is an application. Packages:

- are distributable (ZIP files),
- can install themselves to a supported datasource,
- can be customized per site

# Running 'create_package'

Each task in the `OpenInteract2::Manage` hierarchy has a name; use `oi2_manage` to run them

# Running 'create_package'

Each task in the `OpenInteract2::Manage` hierarchy has a name; use `oi2_manage` to run them

- (we'll get to the hierarchy later)

# Running 'create_package'

Each task in the `OpenInteract2::Manage` hierarchy has a name; use `oi2_manage` to run them

- (we'll get to the hierarchy later)
- 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?

`oi2_manage` creates lots of seed files for us:

- Persistent object configuration
- Action configuration
- Action class
- Class to install database structures and data
- Sample template
- Package-level documentation (use it!)
- Package metadata (MANIFEST, config, changelog)

# Deploy the empty package

The created package actually works as-is: let's deploy it!

# How do we know it works?

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

- `http://myhost/mongers/`
- `http://myhost/mongers/list/`
- `http://myhost/mongers/hello/`
- We can also just ask the server with `oi2_manage`

# Modify a Template

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

# Meet the templates


Our templating engine of choice: Template Toolkit

# Meet the templates


Our templating engine of choice: Template Toolkit

- You have a problem with that?
- Don't worry, you can use your favorite too...

# Change the template

So let's change the sample template, accessing the plugin for user info

# Some package sanity checks

OI2 comes with tools to do some basic package checks:

- Does your MANIFEST match?
- Do your classes compile?
- 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 format
- 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

Packages don't make much sense outside of a website:

- 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

# 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:

- Declaration: `conf/action.ini`
- Code: `OpenInteract2/Action/SomeAction.pm`
- ...package name convention: actions subclass `OpenInteract2::Action`

# 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

By default each task maps to a method. So:

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

**conf/action.ini**

```
[mongers]
class = OpenInteract2::Action::Mongers
```

**Class**

```
OpenInteract2::Action::Mongers
```

**Method (also: Task)**

```
sub list { ...}
```

# Coding a new task

We'll add a new task 'display' to show static data

# Redeploy the quick way...

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

# Use objects from database

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

# 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
6. Modify templates to view database data

# Create your structures

- Create the table in `struct/monger.sql`
- Create sequence in `struct/monger_sequence.sql`

# Create seed data

- Create `data/initial_data.dat`: data to import

# Modify SQLInstall class

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

# Configure SPOPS object

- Change 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 no stinking templates!

# Deploy with a change

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

# Install structures and data

Install the SQL structures and data

# See how it works

- View: `http://myhost/mongers/display/`
- We should now see the data from the database

# Onto part two!

# For more information

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

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

This presentation
http://www.openinteract.org/yapc_2004/

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