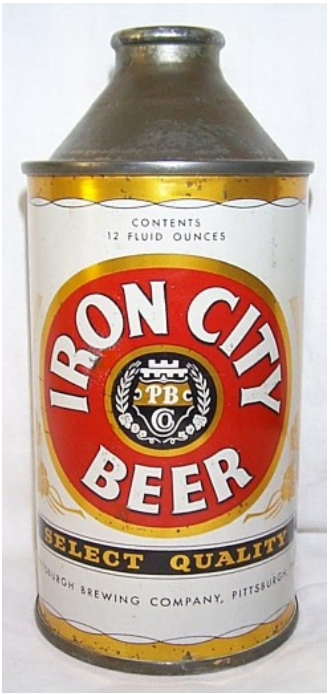
# Building Applications with OpenInteract2

Chris Winters

Optiron Corporation

June 17, 2004

# Part 3: Our main application




We'll borrow Simon's example of the Beer DB

- ...with a few modifications

# What the app will do

- Data entry: enter and categorize beers
- ...including all the related information
- Allow users to comment on beers and pubs
- Search for beers and pubs in our full-text index
- Hook into our sitewide "What's new?" page

# Future features

- Boxes: for an emeddable summary of recent activity
- Security: so only the chosen few can create beers
- Filters: we can become whores for advertising
- Searches: allow complex multi-table searches
- Observers: enable notifications on new beer additions
- ...covered in later parts of the tutorial

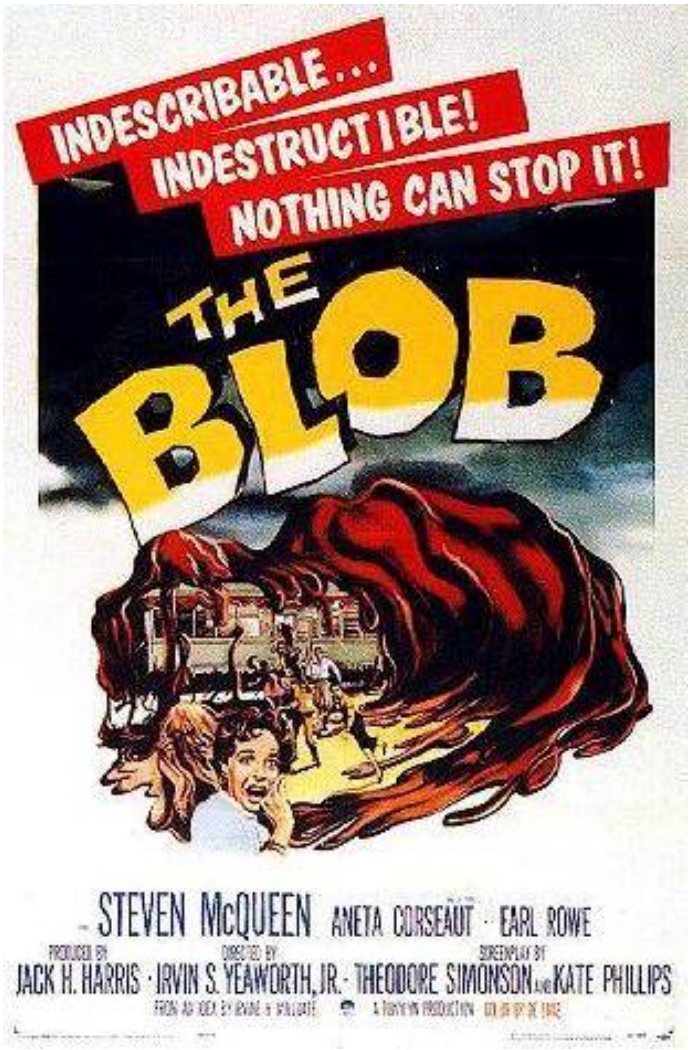
# First step: Data structures

- Start out with the tables
- beer: our beers
- pub: where the beer is served
- style: type of beer
- brewery: who makes the beer
- + one linking table for pub to beer

# Creating portable applications

- OI2 allows you to create portable structures
- ...install them on any supported database
- Using SPOPS: map objects to records in the structures
- Nice: database handle always available
- ...since object-relational tools are never perfect

# Keys to database portability




- Not many differences for common use
- Retrieving IDs
- Datetime declaration
- BLOB/CLOB handling
- ...but we don't deal with BLOBs (scary)

# Abstracting out the differences

- Most of these are field-based
- ...so we can do simple runtime substitutions to deal
- ...better yet: let someone else do it:
  `SPOPS::Import::DBI::TableTransform`

# Table substitutions at work

```
CREATE TABLE mytable (
  id         %%INCREMENT%%,
  other_id   %%INCREMENT_TYPE%% NOT NULL,
  some_date  %%DATETIME%% NULL,
```

MySQL

```
CREATE TABLE mytable (
  id         INT NOT NULL AUTO_INCREMENT,
  other_id   INT NOT NULL,
  some_date  DATETIME NULL,
```

Microsoft SQL Server

```
CREATE TABLE mytable (
  id         NUMERIC(10,0) IDENTITY,
  other_id   NUMERIC(10,0) NOT NULL,
  some_date  DATETIME NULL,
```

PostgreSQL

```
CREATE TABLE mytable (
  id         INT NOT NULL,
  other_id   INT NOT NULL,
  some_date  TIMESTAMP NULL,
```

# Write up the tables

Create the package and code the tables

# Setting up table installation

- Every package can have a 'SQL installer'
- ...class with hooks for installing structures and data
- ...declared in `package.conf`

```
name            mongers
version         0.01
author          Chris Winters <chris@cwinters.com>
url             http://www.openinteract.org/
sql_installer   OpenInteract2::SQLInstall::Mongers
```

# More about the SQL installer

- Most work done in `OpenInteract2::SQLInstall`
- ...just declare your table, data and security filenames
- But you can do more...
- ...seed the database with current data (weather, `/etc/password`, Active Directory)
- ...load different datasets depending on user input

# Adding our tables and sequences

Just add our table and sequence filenames to `OpenInteract2::SQLInstall::Beer`

# Map SPOPS objects to tables

- SPOPS relies on external configuration
- At runtime it generates a class into existence
- ...this class can fetch, store, remove and query for objects
- ...and relate to other objects as well (we'll get to that)

# SPOPS isn't required your code

- OpenInteract2 has excellent support for `SPOPS`
- ...but you can use `Class::DBI` if you like

# SPOPS isn't required your code

- OpenInteract2 has excellent support for `SPOPS`
- ...but you can use `Class::DBI` if you like
- or `Alzabo`

# SPOPS isn't required your code

- OpenInteract2 has excellent support for `SPOPS`
- ...but you can use `Class::DBI` if you like
- or `Alzabo`
- `Tangram`

# SPOPS isn't required your code

- OpenInteract2 has excellent support for `SPOPS`
- ...but you can use `Class::DBI` if you like
- or `Alzabo`
- `Tangram`
- `Rosetta`

# SPOPS isn't required your code

- OpenInteract2 has excellent support for `SPOPS`
- ...but you can use `Class::DBI` if you like
- or `Alzabo`
- `Tangram`
- `Rosetta`
- `DBIx::Recordset`

# SPOPS isn't required your code

- OpenInteract2 has excellent support for `SPOPS`
- ...but you can use `Class::DBI` if you like
- or `Alzabo`
- `Tangram`
- `Rosetta`
- `DBIx::Recordset`
- `DBIx::DBO2`...

# SPOPS isn't required your code

- OpenInteract2 has excellent support for `SPOPS`
- ...but you can use `Class::DBI` if you like
- or `Alzabo`
- `Tangram`
- `Rosetta`
- `DBIx::Recordset`
- `DBIx::DBO2`...
- ...even your own home-grown, half-assed system

# Using SPOPS in OI2

- Don't need to remember/type package names
- ...name vs. class provides insulation for authors
- Just lookup the class from the context:

```
my $beer_class = CTX->lookup_object( 'beer' );
my $beer = $beer_class->fetch( 13 );
print "Beer with ID 13 is $beer->{name}";
```

# Our first mapping

- Each SPOPS object is one INI file
- ...easy to manage and modify
- ...by default they're in `conf/spops_*`
- We'll start out with standalone objects
- ...later link them together when we're running

# Map the objects

For now we'll do the 'beer' and 'style' objects

# Second: Create actions

- Now we want to do something with our objects
- This is where actions enter the picture

# Actions are the heart

- An OI2 action is your application
- ...it's the code that actually does the work
- ...as we saw before actions are looked up by name
- And OI2 provides glue to match URL names to Action names

# Actions and URLs, refreshed

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

**conf/action.ini**

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

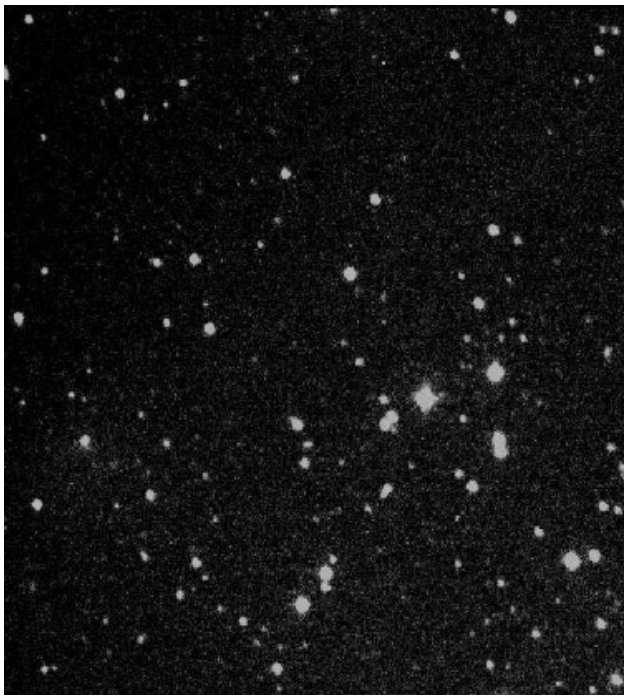
**Class**

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

**Method (also: Task)**

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

# Most applications are similar

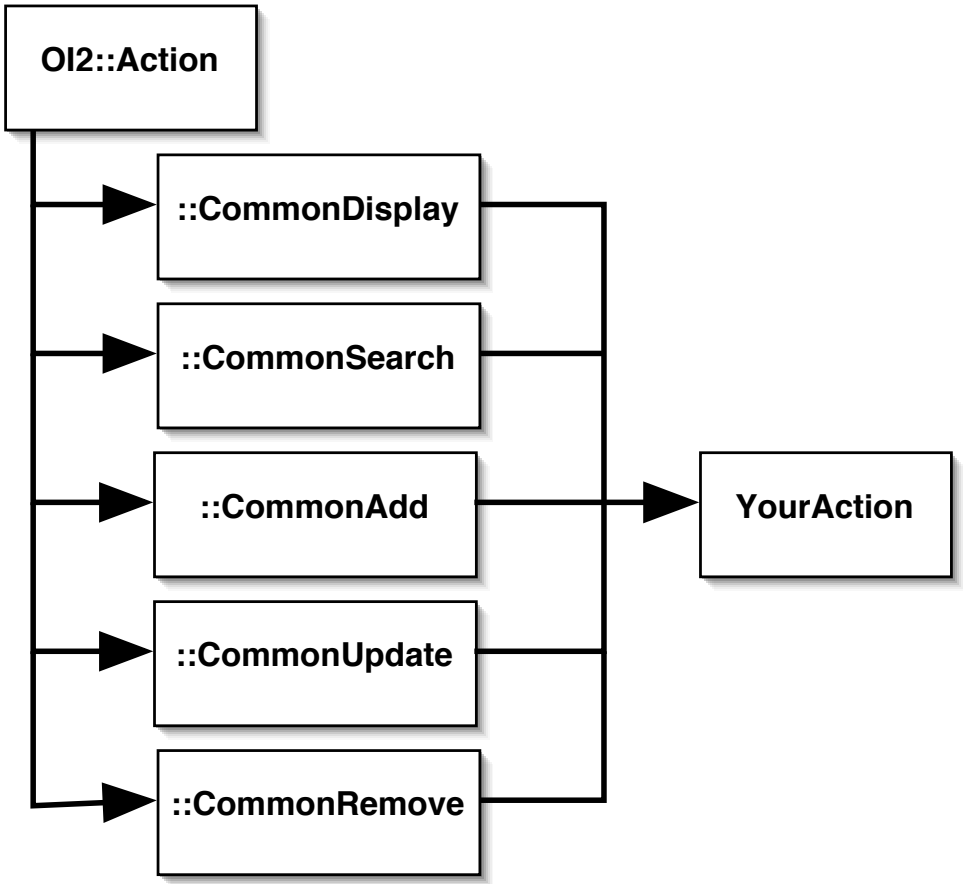


- Find yourself doing repetitious work?
- Create, update, delete, search...
- How many applications just use this?
- ...billions and billions

# Common actions

- OI2 comes with common actions to:
- display: show a record
- search: find multiple records
- add: create a new record
- update: modify an existing record
- remove: permanently delete a record

# How they're setup



# How to use in your code

Just put the relevant classes in your `@ISA`

```
package OpenInteract2::Action::MyAction;

use base qw(
    OpenInteract2::Action::CommonDisplay
    OpenInteract2::Action::CommonSearch
);
```

# Mix-and-match

- You don't need to use all of them, just what you want
- Using just 'search' and 'display' gives you a read-only app
- ...which you can retrofit onto existing data quite nicely

# Declare what you want to do

All common actions rely on action properties like:

- **What type of object should I work on?**
- **What template should I use to display it?**
- **What fields should I search on?**
- **What fields should I use for creating a record?**
- **These are declared in action configuration**

# Action = class + configuration

- Actions instantiated with configuration:

**pkg/beer/action.ini**

```
[beer]
class = OpenInteract2::Action::Beer
c_object_type = beer
c_display_template = beer::beer_display
```

**code anywhere in OI2**

```
my $action = CTX->lookup_action( 'beer' );
print "Object type for common action: ",
      $action->param( 'c_object_type' );

>> Object type for common action: beer
```

# Override what you want

- Every action is an object
- ...so you can override whenever you want

```
sub param {
    my ( $self, $name, $value ) = @_;
    unless ( $name eq 'c_display_template' ) {
        return $self->SUPER::param( $name, $value );
    }
    return ( $self->_is_during_oktoberfest )
             ?  'beer::beer_display_oktober'
             :  'beer::beer_display';
}
```

# Adding value to common things

- All `::Action::Common*` classes allow customization
- Display: `_display_customize()`
- ...add to template's parameters
- ...such as: sourcing a drop-down list of beer styles

# Create the initial beer action

Use the 'add' common functions and implement 'search'

# Some initial templates

- We'll use Template Toolkit throughout
- ...adapt to your favorite system mentally

# Some initial templates

- We'll use Template Toolkit throughout
- ...adapt to your favorite system mentally
- ...and silently

# Coding with template widgets

- OI2 has a set of TT widgets for common uses
- ...especially for forms
- ...allows for centralization

# Widget example: before

- Instead of the typical two-column view...

```
<tr>
  <td><b>Screws Loose</b></td>
  <td><input type="text"
             name="screws_loose"
             size="8" value="14" />
  </td>
</tr>
```

# Widget example: after

- You get a single TT call
- ...which you can decorate in one place later
- ...not farfetched: using CSS vs embedded tables for borders...

```
[% INCLUDE label_form_text_row(
        label = 'Screws Loose',
        name  = 'screws_loose',
        size  = 8,
        value = 14 ) %]
```

# Widget example: after

- ...and easily localize

```
[% INCLUDE label_form_text_row(
        label_key = 'label.screws',
        name  = 'screws_loose',
        size  = 8,
        value = 14 ) %]
```

# Lots of widgets

- Form-based widgets (select, radio, text/area, submit...)
- Tables, column headers
- Date selector, search page count
- Error/status message display
- More advanced: transferring items using javascript

# Code the beer templates

Create the search form, search results and beer form

# Deploy and create beers

- Create a few records, search for them...

# Different action types

- OI2 also has 'action types'
- ...sort of like common actions, but no class needed

# Action type declarations

- Action types are declared in your server configuration
- (...can add dynamically too)

**conf/server.ini**

```
[action_types]
template_only = OpenInteract2::Action::TemplateOnly
lookup        = OpenInteract2::Action::LookupEdit
```

# The lookup action

- The 'lookup' package has a 'lookup' action type
- ...supports declarative grid-based editing
- ...generally useful for lookup table values
- ...kind of like beer styles

# Declaring our beer styles for editing

- Create a new action 'beer_style'
- ...has action type of 'lookup'
- ...plus list of our fields, labels, and sizes

# Create beer style action

Create new action for editing beer styles

# Add some beer styles

- Add some styles using the lookup tables
- ...and add some new beers with styles

# Onto part four!

# 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/