Building Applications with OpenInteract2

Chris Winters

Optiron Corporation

June 17, 2004

Continue with the other objects

- Repeat the `SPOPS/::CommonAction` dance
- Add pubs and breweries
- ...with forms, etc.

Add pubs and breweries

Add object mappings and actions

OI2 shortcuts with SPOPS rules

- The server startup process is fairly extensive
- ...which is why the tests take so long to run
- ...partly because we're slinging lots of files around
- But also: lots of places to step in and do magic

SPOPS rules: quick intro

- SPOPS allows you to step in at lots of places
- ...before/after fetch, save, remove
- ...kind of like AOP
- These are called **rules**

Some sample rules

- Rules can do whatever you want:
- ...translate a date field to/from a `DateTime` object
- ...filter certain words before users ever see
- ...spellchecking
- ...translate to/from unicode

OI2 comes with some rules

- One of the rules: index objects
- ...this is a 'post-save' rule
- Every time you save an object it's added to the index
- ...easy as pie

Indexing by configuration

- How do we enable it?
- Configuration, of course:
- ...add a ruleset to `isa` (if you need other behavior)
- ...add a ruleset to `rules_from` (if you don't)

```
[news]
class      = OpenInteract2::News
isa        =
rules_from = SPOPS::Tool::DBI::MaintainLinkedList
....
```

...and that's how OI 1.x worked

- If you wanted an object indexed: add the right rules

```
$spops = {
 'news' => {
   class => 'OpenInteract::News',
   isa   => [ qw/ OpenInteract::FullText
                  OpenInteract::SPOPS::DBI
                  SPOPS::Secure
                  SPOPS::DBI::MySQL
                  SPOPS::DBI / ],
...
```

The problem with that...

- You need to remember the right class
- ...and **that can't change**
- It's messy too, tough to see nesting unless you format right
- ...yet another benefit of INI files

Compare and contrast

What could be easier than this?

```
[news]
class          = OpenInteract2::News
isa            =
rules_from     = SPOPS::Tool::DBI::MaintainLinkedList
is_searchable = yes
...
```

How do we do it?

- The class `OpenInteract2::Config::Initializer` is observable
- ...one of the observations fired says:

I'm done reading the SPOPS configuration!

The observer code

- We have a bunch of observers that catch the observations
- Each observer is very simple; here is the heart of one:

```
sub _spops_fulltext {
    my ( $init, $type, $config ) = @_;
    return unless ( $type eq 'spops' );
    if ( $config->{is_searchable} eq 'yes' ) {
        unshift @{ $config->{isa} },
                'OpenInteract2::FullTextRules';
    }
....
```

You can add your own!

- Just add classes to the `config_watcher` section in server configuration
- ...they can dynamically add filters to actions
- Like our 'Kevin' filter
- ...they can modify SPOPS class behavior before it's generated

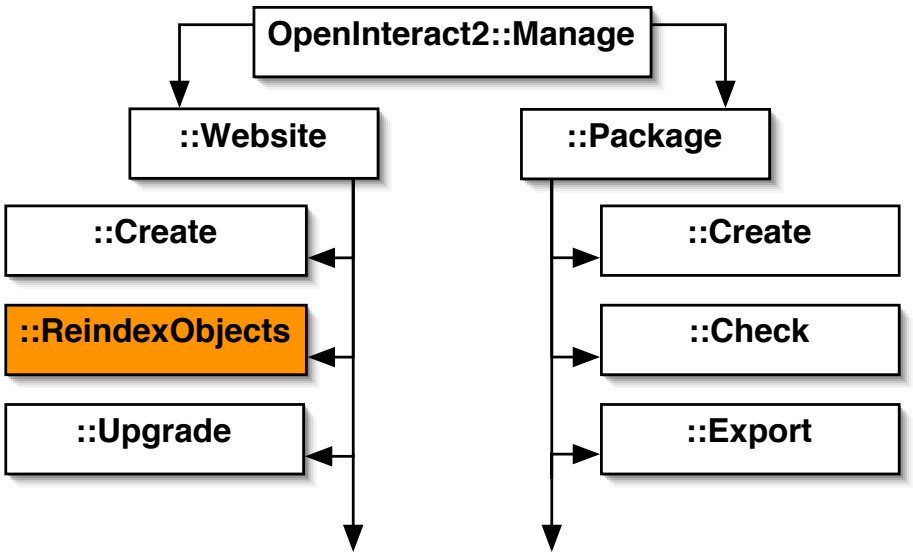
Back to indexing

So adding indexing to beers, pubs and brewery is a snap

How to add existing objects to index

- Our old friend, `oi2_manage`!
- He's got a task for us: `reindex_objects`

Reindexing task in relation



Reindex existing objects

Go ahead and run the task

Search the index

- That 'search' box and button are active you know...
- ...go ahead and search for something we've added
- And we'll meet another part of the SPOPS object: 'display'

SPOPS objects can display themselves

- Well, sort of – they can generate a URL
- They know what type of thingys they are

SPOPS objects can display themselves

- Well, sort of – they can generate a URL
- They know what type of thingys they are
- ...for `humans` that is

SPOPS objects can display themselves

- Well, sort of – they can generate a URL
- They know what type of thingys they are
- ...for `humans` that is
- They also know how to name themselves
- Type: Beer
- Name: Sam Adams

How do they know?

- We told it so!
- `object_name` is the type of thingy
- `name` is a field or method in the SPOPS class

```
[beer]
class       = OpenInteract2::Beer
...
object_name = Beer
name        = name
```

Get identifying data from SPOPS

- Every `SPOPS` object has a method `object_description()`
- Returns a hashref with all sorts of good information
- ...see `perldoc SPOPS` and search for it

Quick: Add the items to "What's New?"

- The "What's New?" listing is maintained through rules
- Just add the class `OpenInteract2::WhatsNewTrack` to `rules_from`
- ...no, we haven't create a shortcut yet

Add the items

We want the beers, breweries and pubs to be added

Adding relationships between objects

- Objects by themselves are kind of interestin
- ...but relationships are much more fun!

Declarative relationships

- Not quite as concise as `Class::DBI`
- ...but we're working on it
- Two types of relationships:
- `has_a` means an object contains the ID of another
- `links_to` means an object links to multiple other objects

How relationships work

- Remember: SPOPS uses a code generation system
- At runtime we read in configuration data
- ...based on that configuration we generate different methods
- ...you can customize this too, but it's a bit hairy

First: has-a relationships in beer

```
CREATE TABLE beer (
    beer_id     %%INCREMENT%%,
    brewery_id int,
    style_id    int,
    name        varchar(30),
    primary key( beer_id )
);
```

- First: beer has-a brewery

First: has-a relationships in beer

```
CREATE TABLE beer (
    beer_id     %%INCREMENT%%,
    brewery_id int,
    style_id    int,
    name        varchar(30),
    primary key( beer_id )
);
```

- First: beer has-a brewery
- ...well, really: beer is-contained-by brewery

First: has-a relationships in beer

```
CREATE TABLE beer (
    beer_id     %%INCREMENT%%,
    brewery_id int,
    style_id    int,
    name        varchar(30),
    primary key( beer_id )
);
```

- First: beer has-a brewery
- ...well, really: beer is-contained-by brewery
- And: beer has-a style

Declare in SPOPS

- The 'has_a' section can hold multiple relationships
- Object class is key, field is value

```
[beer has_a]
OpenInteract2::Brewery = brewery_id
OpenInteract2::BeerStyle = style_id
```

Methods created at runtime

- The 'has-a' creates a method with some rules
- ...if the ID name is the same as the object referenced: object name
- So after this we can just run `beerstyle()` or `brewery()`
- ... and get the referenced object back

Modify the beer display

Fix the beer display to name our style and brewery

Defining links-to relationships: one-to-many

- There are at least two types of links
- First: the ID of a record is in multiple other records
- ...for example, a single brewery is in many beers
- a.k.a one-to-many

Getting tables in the action

- The key is the class, the value the table linked to with our ID
- So we're saying:
- ...I'm linking to `OpenInteract2::Beer` objects
- ...I can find them in the `beer` table
- ...I can find my beer objects using my ID (`brewery_id`)

```
[brewery links_to]
OpenInteract2::Beer = beer
```

Methods created

- One method is generated at runtime:
- ...the linked object (**beer()**) returns the related objects
- ...always an arrayref

Modify brewery display

Fix the brewery display to list beers

Defining links-to relationships: many-to-many

- Second type of links-to: linking table points to both records
- ...a pub can have many beers, a beer can be at many pubs
- Currently the linking table isn't an official object
- ...unless you make it so and create the relationships yourself

Two declarations to do the job

- Add declarations to both 'pub' and 'beer' SPOPS files
- Both point to same table

```
[beer links_to]
OpenInteract2::Pub = pub_serves_beer
```

```
[pub links_to]
OpenInteract2::Beer = pub_serves_beer
```

Methods created

- Three methods generated at runtime, use pub as example:
- ...`beer()` still returns arrayref of related items
- ...`beer_add( $beer )` adds reference to beer
- ...`beer_remove( $beer )` removes reference to beer

Modify pub and beer displays

Fix the pub display to list beers, beer to list pubs

Onto part five!

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/