GOM

THE GOBJECT MAPPER

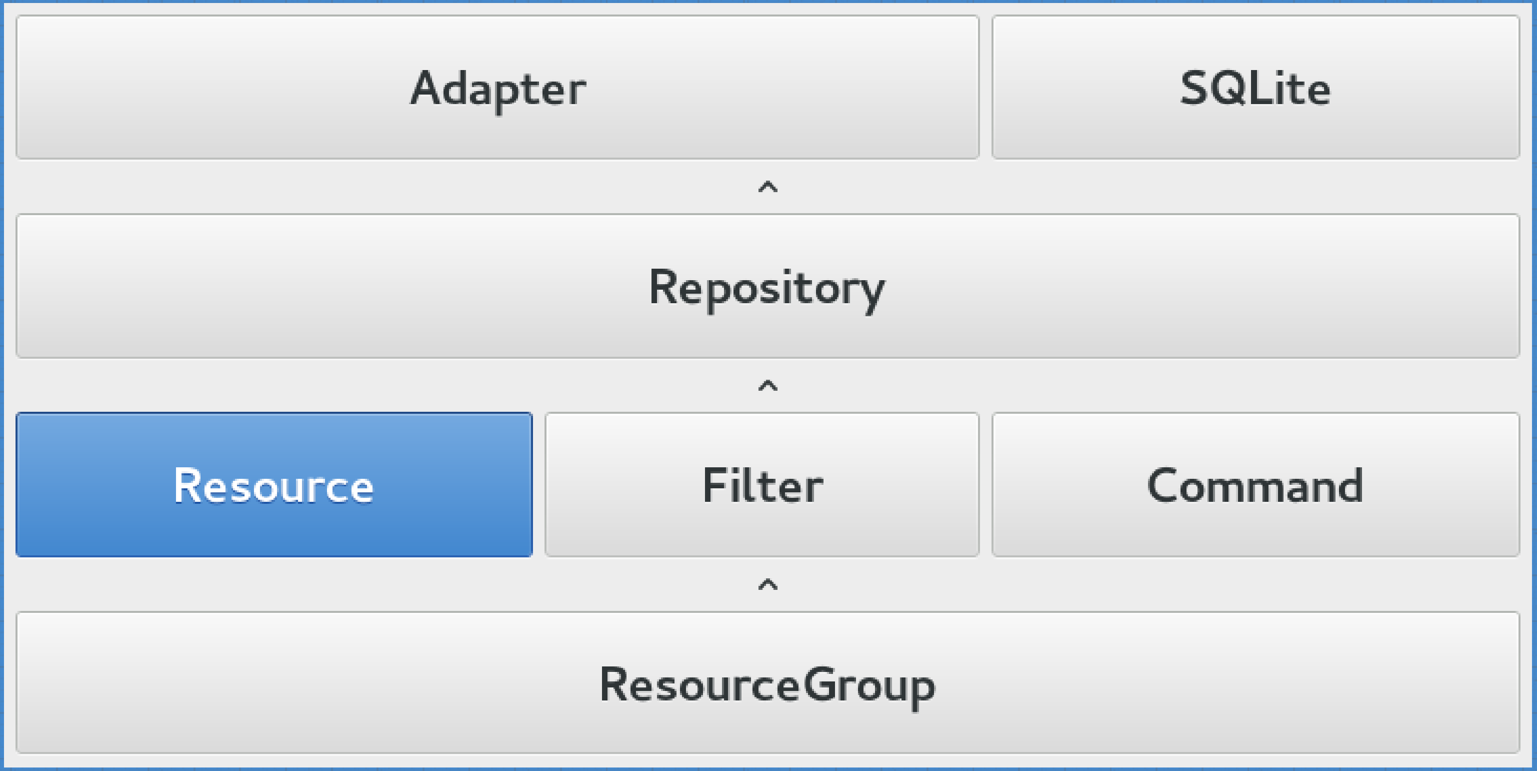
Christian Hergert
<christian@hergert.me>

#GUADEC2014

WHAT IS AN OBJECT MAPPER?

- Bridges an Object's properties to/ from storage.
- It also provides an interface to query existing items.
- Deleted items are truncated from storage.

GOM API DESIGN



RESOURCE

GomResource is your data object's Parent GObject.

It provides lots of helper functions.

- Set Primary Key for Class
- Set Database Table
- Save (a)synchronously
- Delete (a)synchronously
- Fetch relation via many-to-many table
- Configure custom GValue<->SQLite translators

CLASS INHERITANCE

Resources can inherit from other resources.
Resource -> Shape -> {Rectangle, Circle}

GOM will automatically join inheritance tables for you in CRUD operations.

REPOSITORY

Resources save-into and are retrieved-from a Repository.

¡This should generally be performed asynchronously!

Repositories contain all of your data objects.

You typically only need one per-process.
Consider it your data Singleton.

ADAPTER

Adapter is responsible for translating repository operations into storage-layer operations.

Currently, only SQLite is supported.

A GVariant based adapter would be lovely.

RELATIONSHIPS

You are responsible for inflating relating structures.
Resource provides helpers to simplify this.

Use an accessor to fetch the relation asynchronously so it need not be inflated upfront.

Clearly there is room for improvement here.

RESOURCE GROUP

A lazy collection of Resources.
Can enforce type of contained objects (Generics).

You can inflate ranges of the result asynchronously.
Great for "infinite scroll" scenarios.

FILTER

Filters are used to query a repository.
A Filter attached to a ResourceGroup would fetch all objects of a given type matching said Filter.

Think of Filter as the WHERE clause of a SQL statement.

COMMAND

Command allows for direct access to storage layer.

- Run a SQLite statement.
- Used by Resource for CRUD operations.
- Can return a Cursor for iterating result set.

CURSOR

Abstracts iteration of the result set.
Also handles marshaling out of SQLite.
Helpers to access known types, or just get a GValue.

SCHEMA GENERATION

Support for auto-generating schema from property information.

Support for automatically performing migrations. (Will never delete columns).

Alternatively, you can handle this all manually with the migration callback.

TRANSACTIONS?

Not today.

You can emulate them via a write-callback in the SQLite thread.
(Just no write concurrency).

GOBJECT INTROSPECTION

Not quite there yet.
Need to figure out how to integrate class-level functions.

Excited for cross-language potential.

Potential for a LibSoup based web server with handlers in various languages (sharing data model and connections).

DEMO TIME!

CODE COMPLETION

Let's keep it simple, we just have symbols.

Symbols have a type and a name.

CODE COMPLETION

<u>Symbol</u>

id: Integer

name: String

type: Enum

CODE COMPLETION

CodeSymbol inherits GomResource
enum CodeSymbolType

CODE COMPLETION

CodeSymbolCompletionProvider implements GtkSourceCompletionProvider

Queries GomRepository for CodeSymbol with :name LIKE 'foo%'.

Terribly naive in terms of code completion, but quick and easy.

LET'S RUN IT

QUESTIONS?

https://git.gnome.org/browse/gom
https://github.com/chergert/gom-talk-demo

And a special thanks to all our contributors!

Bastien Nocera, Christian Kirbach, Cosimo Cecchi, Daniel Mustieles, Enrico Nicoletto, Jonathon Jongsma, Marek ernocký, Matej Urban i , Mathieu Bridon, Rafael Ferreira, Tristan Brindle