JRUBY AND RAILS ON GOOGLE APP ENGINE

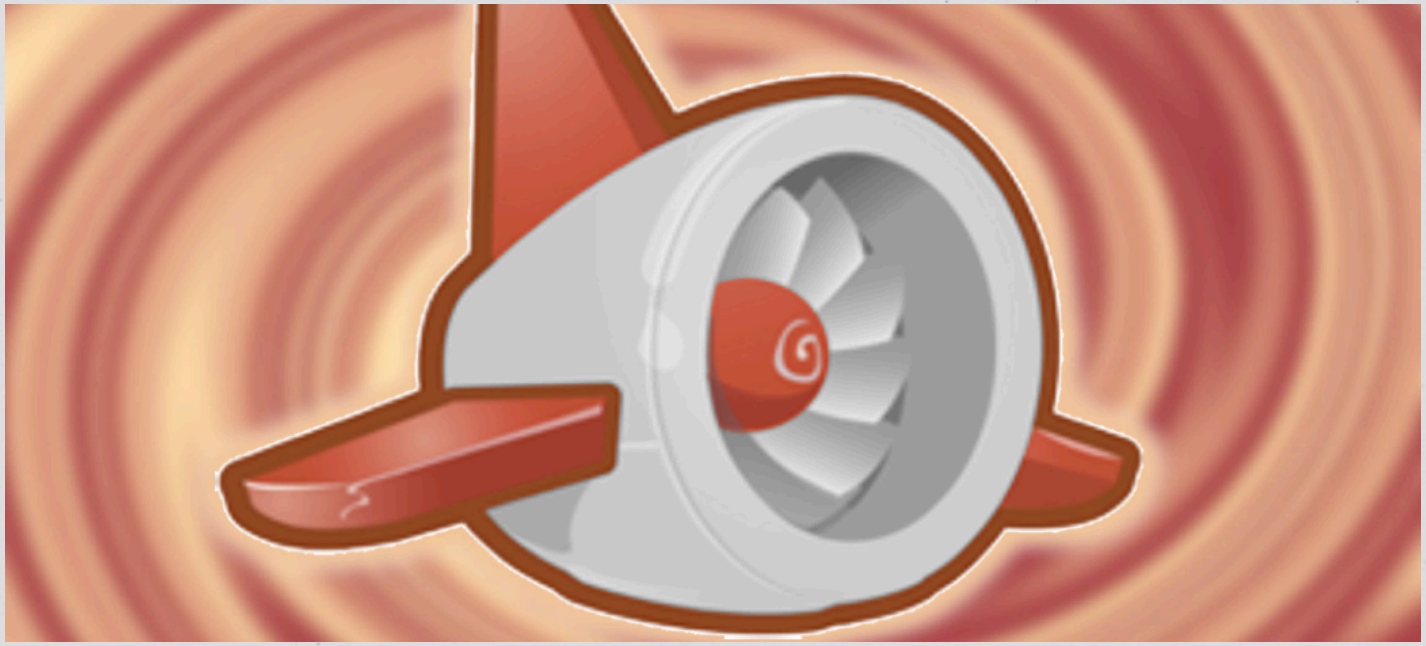
Amy Unruh

Overview of Talk

- a demo
- JRuby and why you might want to use it
- Running a Rails app on JRuby
  - using Java libs from your Ruby code
- Running a Rails app on Tomcat with JRuby
- Running Rails 2 on Google App Engine/Java
  - accessing the App Engine Datastore with the DataMapper adapter
  - using App Engine services

DEMO

http://aju-rails.appspot.com/



JRuby

- Java implementation of Ruby (`jruby.org`)
  - Ruby 1.8.7 compatible (with a few minor differences)
  - 1.9 compatible soon
- Core group of developers: Charles Oliver Nutter, Thomas Enebo, Nick Sieger, Ola Bini, Marcin Mielzynski, Bill Dortch, Wayne Meissner, MenTaLguY

Advantages of JRuby

- Can access Java APIs– both core libs and third-party packages– in your JRuby scripts
- Can run Rails apps on Java App Servers like Tomcat
- Can run Rails apps on platforms (like App Engine) that don't support Ruby
- Outperforms MRI in many cases
  - 2x to 10x
- Better threading support, access to JVM tools

JRuby Gems and Libraries

- Complete implementation of Ruby core/builtin classes, nearly complete implementation of standard libs
- Any libraries/gems that use native C must be reimplemented in Java
  - `jruby-openssl`, …
  - Some gems build their native library in a platform-independent manner
    - **mongrel** is one
  - Most 'critical' gems supported in JRuby now

Accessing JRuby

- Run `jruby -S` to access the JRuby versions of system-level executable commands

  -
    ```
    jruby -S gem install <gemname>
    jruby -S rails <appname>
    ```
- Use '`jruby`' to run Ruby programs

  -
    ```
    jruby script/server
    jruby a_ruby_script.rb
    ```
- invoke `jirb` (instead of `irb`)

JRuby on Rails

- Uses **JDBC** to talk to MySql
- `jruby -S gem install rails mongrel jdbc-mysql \ activerecord-jdbcmysql-adapter`
- Modify **database.yml**:

  ```
  adapter: jdbcmysql
  ```
- Then, just pay attention to which implementation of Ruby you are using:

  ```
  jruby -S rails <appname>
  jruby -S rake db:migrate
  jruby script/server
  ```

Accessing Java APIs from Ruby

```
require 'java'

import java.net.URL;
import com.sun.syndication.fetcher.impl.HttpURLFeedFetcher;
```

- Either add jars to the CLASSPATH, or require them.
  - `require 'path/to/mycode.jar'`

Accessing Java APIs

```
require 'java'

import java.net.URL
import com.sun.syndication.fetcher.impl.HttpURLFeedFetcher

class PostsController < ApplicationController
  def index
     // use java libs to fetch and parse ATOM feed
    fetcher = HttpURLFeedFetcher.new
    url = URL.new("http://twitter.com/statuses/user_timeline/newscientist.atom")
    sf = fetcher.retrieveFeed(url)
    @feedTitle = sf.getTitle()
    @feedDescription = sf.getDescription()
    @entries = sf.getEntries()
    @posts = Post.all
    ...
  end
```

(DEMO: JRUBY ON RAILS)

Running Rails on a Java App Server

- Thanks to Nick Sieger's Warbler gem, this is straightforward
  - creates a .war file from a Rails, Merb, or Rack-based application
  - bundles `jruby-core`, `jruby-stdlib` and `jruby-rack` .jar files
  - pulls in jar files from `<app-root>/lib`
  - installs gems into WEB-INF/gems
- `jruby -S gem install warbler`
- In <app-root>: % `warble`
  - edit `config/warble.rb` to specify the gems to include
    - to use jdbc-mysql, uncomment:
      `config.gems += ["activerecord-jdbcmysql-adapter", "jruby-openssl"]`
- Then copy the .war file to (e.g.) `<tomcat>/webapps` to deploy

DEMO: RAILS ON TOMCAT

GOOGLE APP ENGINE/JAVA

Features & Benefits

- Google App Engine lets you run your web applications on Google′s infrastructure.
- Highly **scalable**, including many scalable **services**
    - Distributed **Datastore** based on BigTable
- Don′t need to administer stack or data store
- **Admin console** for app management and statistics
- Pay as you go (with small quotas free)

Some Limitations & Considerations

- No access to file system, no threads, no sockets
  - So, not all third-party Java code will run on App Engine
- 30-second request timeout
  - and, no long-running http connections (no streaming)
- Can be long spin-up time– problematic for apps that aren't heavily used
- The Datastore is not relational; this can make existing app ports challenging
  - (but: App Engine for Business)
- Request/response size and Datastore *entity* size limits (but: Blobstore)
- Roadmap includes: background servers that can run longer requests, 'warm' or reserved server instances, raised limits on request/response size

App Engine services

- The Datastore
- Task Queues and cron jobs
- Memcache
- XMPP
- Google Accounts User API
- Mail, Images, Blobstore
- Upcoming/beta: Channel API (push to client), OAuth support, MapReduce service, OpenId

The Datastore

- Based on BigTable, distributed and highly scalable, *not relational*
  - **Read time is linear in size of result, not stored data**
  - All *versions* of an app share the same Datastore
- Constrains the query types allowed in order to maintain linear access time and allow partitioning. E.g.:
  - no joins (yet)
  - can't perform multiple inequality tests in a single query
- Has bulk loading capabilities

Datastore entities

- The Datastore holds **entities**
  - Entities have a *kind*, and contain *properties*, or named values, from a set of *core value types* (http://goo.gl/DOPf)
  - Properties can be multivalued (*property list*)

Kind	Friend	Friend
(Entity ID portion of) Key	12345	44
firstName	Bob	Joe
lastName	Smith	▨
tags	colleague, sports	coffee-drinker
emailAddress	▨	joe@joe.com

- Entities are **schema-less**
  - Two entities of the same kind do not need to have the same properties, or use the same value types for the same properties.
  - so: no schema-definition migrations necessary

A Lot More to the Datastore…

- How its indexes are structured and maintained
  - Default and custom indexes
  - Constraints on query construction
- Entity *groups*
  - Can define key-based parent/child relationship between entities
  - Such entities form a group based on ancestor root
  - *Transactions* may only occur across entity groups

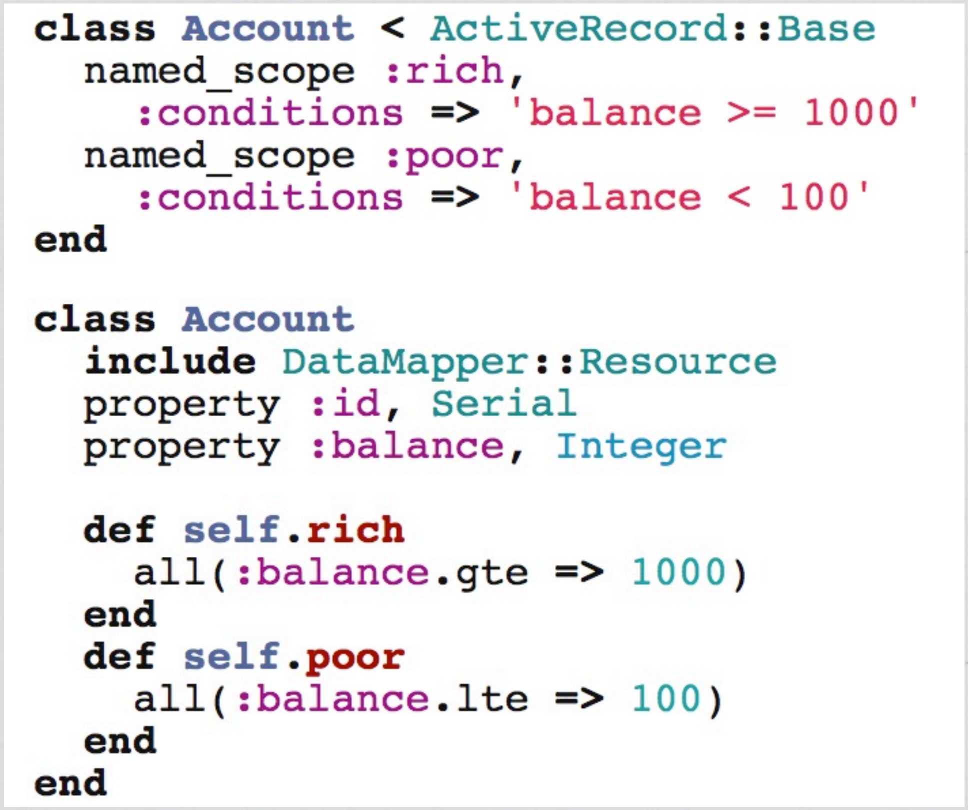
DataMapper and the Datastore

- ActiveRecord does not work on App Engine
- DataMapper is a Ruby ORM mapper
  - `http://datamapper.org/`
- A DataMapper/Datastore adapter has been created
  - Josh Moore, David Masover, ...
- Not all DataMapper features are supported by the Datastore integration
  - Queries unsupported in the Datastore will throw a runtime error
- If you're adapting an existing app, Datastore query constraints may require changing your data model

Defining Models using DataMapper

- Specify model properties **in the model files**
- Use Datastore core value types
  - Of note: `List` (Datastore property lists), `Text` (strings > 500 chars)
  - Use `Serial` for id

DataMapper vs ActiveRecord



```
class Account < ActiveRecord::Base
  named_scope :rich,
    :conditions => 'balance >= 1000'
  named_scope :poor,
    :conditions => 'balance < 100'
end

class Account
  include DataMapper::Resource
  property :id, Serial
  property :balance, Integer

  def self.rich
    all(:balance.gte => 1000)
  end
  def self.poor
    all(:balance.lte => 100)
  end
end
```

(example from David Masover)

Queries in DataMapper

- Use hashes and builtin comparison methods
  - `books = Book.all(:price.gt => 20, :title => ‘My Story’)`
  - sort order: `@books = Book.all(:order => [ :title.asc ])`
  - Illegal for Datastore (multiple inequality tests): `books = Book.all(:price.gt => 20, :title.lt => ‘A’)`

DataMapper Associations

ActiveRecord Terminology	DataMapper Terminology
has_many	has n
has_one	has 1
belongs_to	belongs_to

- Many-to-many associations currently require special treatment with the Datastore adapter
  - DataMapper docs do not apply

```
class Category
  include DataMapper::Resource

  property :id,   Serial
  property :name, String, :required => true, :length => 500
  timestamps :at
  has n, :books
 end
```

Running a Rails App on GAE/J

- Unofficial, but works pretty well already!
- Under active development
- http://code.google.com/p/appengine-jruby/wiki/RunningRails , http://groups.google.com/group/appengine-jruby ...
  - (John Woodell, David Masover, Ryan Brown, ...)

Configuration/Setup for GAE/Rails

- http://gist.github.com/486250
  - Use MRI, not JRuby (JRuby jars will be bundled)
  -
    ```
    sudo gem install google-appengine
    sudo gem install rails -v "2.3.8"
    sudo gem install rails_dm_datastore
    sudo gem install activerecord-nulldb-adapter
    ```
- ...then run a setup script

Configuration for GAE/Rails (cont'd)

- Files of note generated by the setup script:
  - `config.ru`
  - `app.yaml`
    - is used to auto-generate the `web.xml` and `appengine-web.xml` files
    - specifies App Engine app id and version
    - map request handlers, specify authentication requirements
  - `Gemfile`
    - uses Gem Bundler; result is `WEB-INF/lib/gems.jar`
  - `./script .sh` files
  - `WEB-INF` directory
    - config file changes (`database.yml`, `environment.rb`, ...)

Example: app.yaml File

```
application: aju-rails
version: plist3
runtime: jruby

inbound_services:
- xmpp_message
- mail

handlers:
- url: /remote_api/*
  login: admin
  servlet: com.google.apphosting.utils.remoteapi.RemoteApiServlet
  name: remoteapi

- url: /mailnotif
  login: admin
  servlet: com.example.MailNotifServlet
  name: mailnotif

- url: /staff/*
  login: required
```

Example: generating a scaffold and model

- Generate a **dd_model**
  - can access Datastore core value types
  - don't need a migration for the 'schema'

-
  ```
  ./script/generate scaffold post title:string content:text \
  posted:date tags:list -f --skip-migration
  ```

-
  ```
  ./script/generate dd_model post title:string content:text \
  posted:date tags:list -f
  ```

The result: the `Post` model

```ruby
class Post
  include DataMapper::Resource

  property :id,      Serial
  property :title,   String,        :required => true, :length => 500
  property :content, Text,          :required => true, :lazy => false
  property :posted,  Date,          :required => true
  property :tags,    List,          :required => true
  timestamps :at
end
```

Demo: running on App Engine's local Dev Server

- `./script/server.sh` : generates java app configuration files from `app.yaml`, then starts up App Engine development server
- `http://localhost:8080/_ah/admin/` : local admin console, access to local 'Datastore'
- `WEB-INF/appengine-generated/local_db.bin` : holds local Datastore

Deployment to App Engine

- Create App Engine app id after signing in at appengine.com
- Define app id and app *version* in app.yaml file
- `./script/publish.sh`

Using App Engine Services via their Ruby APIs

- Require the service(s), e.g.:

  -
    ```
    require 'appengine-apis/memcache'
    require 'appengine-apis/labs/taskqueue'
    require 'appengine-apis/users'
    ```

Using the App Engine Memcache

```
def index
  cache = AppEngine::Memcache.new
  @books = cache.get(:books)
  if @books.nil?
    @books = Book.all(:order => [ :title.asc ])
  end
  cache.set(:books, @books)
end

def update
  cache = AppEngine::Memcache.new
  cache.delete(:books)
  ....
```

Adding a Task to the App Engine Task Queue

```
prms = {"booktitle" => @book.title}
AppEngine::Labs::TaskQueue.add(:url => "/mailnotif", :params => prms)
```

- Tasks called via *webhooks*, run asynchronously with admin privs
  - multiple tasks may be run in parallel
  - tasks are retried if they fail (return an exception), so should be *idempotent*
- Multiple task queues may be defined, with different throughputs and ‘bucket sizes’
  - defined in WEB-INF/`queue.xml`
  - This example uses the default task queue

Authentication with the App Engine Users API

* app.yaml:

```
- url: /staff/*
  login: required
```

```
<% if AppEngine::Users.logged_in? %>
 <%= link_to "Go to staff site", :controller => "staff", :action => "index" %>
<% else %>
 <%= link_to "Log in to staff site",
  AppEngine::Users.create_login_url(url_for(:controller => "staff", :action => "index")) %><br>
<% end -%>


<%= link_to "Logout", AppEngine::Users.create_logout_url(url_for(:controller => "bookstore",
  :action => "index")) %>
```

Integrating Servlet Request Handlers with Rails

- `require 'appengine-rack/java'` in `config.ru`
  - (However, this did not work for me; I needed to copy the `java.rb` file to my app)
- Allows direct access to App Engine Services via the Java API
- Define servlet/URL mapping in `app.yaml`
- Build jars with App Engine & servlet jars in `CLASSPATH`; move result to `WEB-INF/lib`

```
#!/bin/sh
export CLASSPATH=./build/lib/servlet.jar:./WEB-INF/lib/appengine-api-1.0-sdk-1.3.5.jar

javac com/example/Plain.java
javac com/example/MailNotifServlet.java
mkdir -p WEB-INF/lib
jar -cvf WEB-INF/lib/examples.jar com/example/*.class
rm com/example/*.class
```

Example: A Task to send email & XMPP Notifications

```
public class MailNotifServlet extends HttpServlet {

   public void doPost(HttpServletRequest req, HttpServletResponse resp)
      throws IOException {
    Properties props = new Properties();
    Session session = Session.getDefaultInstance(props, null);
    String booktitle = req.getParameter("booktitle");
    String msgBody = "a book, '" + booktitle + "', has been added...";
    try {
        // first, send XMPP msg
        JID jid = new JID("amygdala@jabber.org");
        logger.info("sending XMPP msg to: " + jid);
        com.google.appengine.api.xmpp.Message xMessage = new MessageBuilder()
          .withRecipientJids(jid)
          .withBody(msgBody)
          .build();
        boolean result = sendMessage(xMessage, jid);
        logger.info("result of sending the XMPP message: " + result);

        // next, email
        javax.mail.Message msg = new MimeMessage(session);
        msg.setFrom(new InternetAddress("aunruh@gmail.com", "The Admin"));
        msg.addRecipient(Message.RecipientType.TO,
                         new InternetAddress("amy@infosleuth.net", "Ms. User"));
        msg.setSubject("A book has been added to the catalog.");
        msg.setText(msgBody);
        Transport.send(msg);
      } ...
```

Accessing Java Libraries from Rails on App Engine

- Add third-party jar files to `WEB-INF/lib`
- Add jars to `CLASSPATH` when running locally
- Access in your Ruby code as shown
- Not all third-party Java libraries will run on App Engine

Summary

- GAE/J + Rails is a promising and robust integration of technologies
  - Access to App Engine services & automatic scalability is powerful
  - Rails 3 support in the works
  - …just need to get that spin-up-time issue resolved…
- To keep an eye on: Duby (Mirah)
  "Mirah is essentially Ruby syntax for writing Java code"
  `http://github.com/headius/mirah`

END

Many-to-Many Associations using property lists

```
class Author
  include DataMapper::Resource

  property :id,       Serial
  property :name, String,        :required => true, :length => 500
  timestamps :at

  has n, :books, :child_key => [:author_ids]

end
```

Many-to-Many Associations using property lists (cont'd)

```
class Book
  include DataMapper::Resource

  property :id,       Serial
  property :title,  String,   :required => true, :length => 500
  property :price,  Float,    :required => true
  timestamps :at

  property :author_ids, List
  has n, :order_items
  belongs_to  :category

  def authors
    auths = []
    for a in author_ids do
      auths << Author.get(a)
    end
    return auths
  end
 end
```

Many-to-Many Associations using an intermediate 'table'

```ruby
class Book
  include DataMapper::Resource

  property :id,    Serial
  property :title,  String,  :required => true, :length => 500
  property :price,  Float,   :required => true
  timestamps :at

  has n, :authorships
  has n, :order_items
  belongs_to  :category

  def authors
    auths = []
    for athsh in self.authorships do
      auths << athsh.author
    end
    return auths
  end

end
```

Many-to-Many Associations using an intermediate 'table' (cont'd)

```ruby
class Author
  include DataMapper::Resource

  property :id,  Serial
  property :name, String,  :required => true, :length => 500
  timestamps :at
  has n, :authorships

  def books
    books = []
    for athsh in self.authorships do
      books << athsh.book
    end
    return books
  end

end
```