Croquet

William R. Speirs, Ph.D. (wspeirs@metrink.com)
Founder & CEO of Metrink

About Me

- BS in CS from Rensselaer; PhD from Purdue
- Founder and CEO of Metrink (www.metrink.com)
  - Simple yet powerful query language
  - Stateful alerting w/easy-to-use configuration
  - Correlation across all metrics
  - Agent-less collection
  - Elastic scaling

Get visibility into all 7 layers of your infrastructure.

Why Build Croquet?

- Wicket makes it super easy for non-JavaScript devs
  - Web framework used at Metrink
- Jetty provides WebSocket, SPDY, etc
- Hibernate/JPA makes it easy to SQL without SQL
- Guice makes it easier to “do the right thing”

Continually had to write boilerplate code to make it all work.

Why Build Croquet?

- Wicket-IOC isn't great
  - Cannot use constructor injection; field only
  - Manually need to call inject methods
- Configuring Jetty and Guice is non-trivial
  - Even with `GuiceWebApplicationFactory`
- Wiring JPA into Jetty/Wicket w/Guice isn't out-of-the-box
  - Entity Managers don't serialize nicely

Should be able to use these great components with ease!

What is Croquet?

- Croquet is to Wicket at DropWizard is to Jersey
  - Java framework for developing ops-friendly, high-performance, RESTful web services

- Combination of 4 frameworks/libraries
  - Apache Wicket: component based web framework
  - Jetty: servlet container with WebSocket support
  - Hibernate: an ORM framework and JPA provider
  - Google's Guice: dependency injection framework

How Do I Use Croquet?

- Everything driven off configuration
  - Code: things that won't change
  - File: things that change from env to env

- 3 simple steps:

  1) Configure a `Croquet` object w/`CroquetBuilder`

  2) Add any modules (Guice or Managed)

  3) Call `run()`

`CroquetBuilder`

- Used to configure a `Croquet` object
- Parses a configuration file
- Optionally set application class (default usually good)
- Sets the homepage of the application
- Add page mounts to the application
- Adds resources to the application
- Configure a health check page
- Add JPA entities
- Set the SQL dialect

Adding Modules

- 2 types of modules: Guice & Managed

- Guice modules allow you to provide additional bindings
  - These modules are added to the Guice Injector

- Managed modules are started & stopped with Jetty
  - Great for things like HTTP clients, etc
  - You only add classes, dependencies are injected

Call `run()`

- Creates the Guice injector
- Creates each managed module
- Creates a shutdown hook
- Starts the Jetty server
- Drops a PID file on Linux (complains on Windows)

How Is Wicket Configured?

- Extends `AuthenticatedWebApplication`
  - Defaults to an unauthenticated site
- In devel mode:
  - Add `StatelessChecker`
  - Add `DebugBar` in devel mode
  - Doesn’t strip Wicket tags
- Minify both JavaScript & CSS
- Uses `IPageFactory` that creates all pages with Guice

How is Jetty Configured?

- Add the Guice injector and Web Application Factory
- Add a `PersistFilter`
  - Ensures a new EntityManager for each request
- Add a `Jetty9WebSocketFilter`
- Prevent JSESSIONID from appearing in query param
- Set idle timeout to 1 hour, and linger time to infinity
- Change by overloading `configureJetty` method

More settings to come via configuration in the future.

How is JPA Configured?

- Can use either persistence.xml or YAML config file
- Hibernate 4.3.1 is the JPA provider
- `EntityManager` is constructed by the Servlet Filter
  - Wrapped by a proxy so it can be serialized
  - Transactions must be handled manually
- If YAML file used to configure your DB
  - Leverages the Tomcat JDBC Connection Pool
  - Entities added via code through `CroquetBuilder`

How is Guice Configured?

- `CroquetModule`
  - Binds the proper `Settings` class
  - Binds the `WebApplication` class
  - Binds the `IPageFactory` class
- `HibernateModule`
  - Binds everything required by the `PersistFilter`
  - Depends upon method used to configure Hibernate
- Child injector to properly bind `PageParameters`

What About Testing?!?

- Build a `CroquetTest` instance from `CroquetBuilder`
  - Setup exactly the same as `Croquet` instance
  - Simply call the `buildTester()` method
  - Call `getTester()` after modules added
- Use mock instances of dependencies
  - Easiest way is to create mock Guice Modules
- Highly consider using an in-memory DB
  - Liquibase is a great tool for keeping DBs in sync

Demo

Where Can I Get Croquet?

- Maven
  - groupId: `com.metrink`
  - archiveId: `croquet-core`
- Source: github.com/metrink
- Issues: github.com/metrink/issues
- Documentation: croquet.metrink.com

How can I help?

- Use Croquet!
  - Find & report bugs
  - Will try and release as often as possible
- Built for our use cases, but maybe not yours
  - Provide feedback: wspeirs@metrink.com
- Documentation!
  - Provide additional real-world examples

Roadmap

- Create @Restore annotation for non-serializable fields
  - Non-serializable classes must have default constructor (WICKET-1130)
- @Transactional support
- Make Croquet work with Google App Engine
  - Other PaaS providers?
- OSGi integration
  - I know nothing about OSGi :-)

Questions? Comments. Concerns!