Banshie 1.0

Framework Prototype Design and Architecture

Willi Schönborn, 774190

Modules

Parent Module

API Module

Implementation Modules

Utility/Library Modules

Cross-Cutting Reuse

OSGi

- Module System and Service Platform
- Bundle (= Module) = Jar file + Metadata
- Hot Deployment, Dependency Management, ...
- Inter-Bundle communication only via Services
- OSGi in Banshie is supported but opt-in
- Banshie was designed to be an old-fashioned library

API

banshie-api

Public API

— Domain Model and Data Access Layer

— Execution

— Evaluation (Quality and Performance)

The only direct compile-time dependency for clients

Implementation #1

- Design Patterns/Principles
  - Separation of Concerns
  - Single Responsibility Pattern
  - Dependency Inversion Principle
  - Dependency Injection
- results in *many* modules

Implementation #2

Naming Schema

`banshie-<module>-api` and
`banshie-<module>-impl`

Internal APIs or SPIs (Service Provider Interface)

allows replacement of single modules or even whole sub-systems

Dependency Injection #1

Design Pattern

Hollywood Principle: "Don't call us, we'll call you."

```
@Inject
public Service(Dependency dependency) {
    this.dependency = dependency;
}
```

Improves Testability

Dependency Injection #2

- Standardized, extensible dependency injection API

- `javax.inject`-Package

- Implementors
  - Google Guice
  - Spring Framework
  - Dagger

Dependency Injection in OSGi

- **Challenges**
  - Dynamic Un-/Loading of bundles/classes
  - Different Classloaders
- **Peaberry (Guice Extension)**
  - One Injector per bundle
  - Dynamic Services via Proxies

Results

- highly modular framework

- OSGi-capable and embeddable in Non-OSGi applications

- simple and stable public API

- flexible, extensible internal SPI

Links

- http://www.github.com/whiskeysierra/banshie
- Sources
  - http://www.osgi.org/Specifications/HomePage
  - http://www.butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod
  - http://www.martinfowler.com/articles/injection.html