Equinox:
*Improving and Evolving the Core Framework*

Tom Watson

IBM

March 28st 2013
Agenda

• History
• Motivation
• Generic Dependency Model
• Redesign Core Equinox
• What Breaks?
History - SMF

• Service Management Framework
  • IBM OSGi Framework implementation

• Focused on embedded

• OSGi R3 Core Specification

• Small API footprint
  • Comprised mostly of OSGi spec APIs
History - SMF

- Service Management Framework
  - IBM OSGi Framework implementation
- Focused on embedded
- OSGi R3 Core Specification
- Small API footprint
  - Comprised mostly of OSGi spec APIs
- Relatively Simple and Happy implementation
  - ~300 KB including a console

SMF
History – Eclipse 3.0

• Equinox was born
  • Released June 2004

• Focused on running Eclipse
  • Scaling to 1000s of installed bundles

• Input for OSGi R4 Specification
  • Still in progress when Eclipse 3.0 released

• Happy Implementation?
History – OSGi R4

- OSGi R4 Specification - in progress
  - Require Bundle
  - Fragments
  - Multiple versions of a package
  - Singletons

Looks Easy, Right?

Equinox
History – OSGi R4

- OSGi R4 Specification - in progress
  - Require Bundle
  - Fragments
  - Multiple versions of a package
  - Singletons

- Class Space Consistency
  - The 'uses' directive was introduced
  - NP-Complete problem to find a resolution solution

Uh Oh!
History – OSGi R4

• Solution
  • Create API to represent resolution State and Resolver implementation
  • A common implementation used by the runtime and tools (PDE)

Anything can be done with a layer of abstraction

Equinox
History – Eclipse 2.x support

- Eclipse 2.x
  - Plugins with no bundle manifest
  - Plugins as directories
  - Eclipse-style native code lookup
  - Lazy Activation
History – Eclipse 2.x support

• Solution
  • Create an 'SPI' for implementing framework details as a framework adaptor
  • Provide an adaptor for implementing Eclipse 2.x support

Anything can be done with a layer of Abstraction!
History – Adaptors are Great

• The Framework Adaptor is powerful
  • Allowed for more customized support
  • Shared classes support
  • Signed bundle support
  • Class weaving
  • And more ...

Wow! I did not know I could do that!
History – Adaptors are Great

- Framework Adaptor is monolithic
  - Only one adaptor can be configured with the framework
  - Must implement the complete adaptor API or extend an existing one
  - What if you want weaving, shared classes and Eclipse 2.x support?
History – Adaptors are Great

• Solution
  • Create an 'SPI' that hooks into a base framework adaptor
  • Allow hooks to focus on isolated responsibilities
    • Class loading
    • Storage
    • Access to content

Anything can be done with a layer of Abstraction! (I guess)
History – Adaptors are Great

• Solution
  • Create an 'SPI' that hooks into a base framework adaptor
  • Allow hooks to focus on isolated responsibilities
    • Class loading
    • Storage
    • Access to content
  • Lots of hook implementations
    • Lazy activation, eclipse 2.x support, signed bundle support, shared classes support, equinox weaving, equinox transforms ...
History – OSGi R4.3

• OSGi Specified several new hooks
  • Bundle hooks (event, find)
  • Service hooks (event, find, listener)
  • Resolver hooks
  • Weaving hooks

I hope I can handle this
History – OSGi R4.3

• OSGi Resolver Hooks
  • Particularly hard to retrofit into the Equinox Resolver API
  • Riddled with deadlock and thread safety issues because of several coarse grain locks in the implementation
  • Hard to get a consistent view of the locking strategy
History – OSGi R4.3

- OSGi Generic Dependency Model
  - Package Admin proved to be inadequate for modelling the framework wiring state
  - Package Admin was deprecated in favour of a new wiring API
  - Wiring API is based on a generic dependency model
  - Wiring API types are used to call out to resolver and weaving hooks
  - Generic dependencies will be used to express all future OSGi module layer dependencies

Oh this is going to be fun!
History – OSGi R4.3

• **Solution**
  • Adapt the equinox resolver to provide a 'view' of the new OSGi wiring API
  • Requires not-so-nice adaptations of the equinox resolver types into the standard OSGi wiring types
  • Allows the adapted objects to be used to call resolver and weaving hooks
History – OSGi R5

- OSGi Resolver Service
  - A new Resolver service is specified
  - Specifies a standard way to provide a resolver to find a resolution solution for generic capabilities and requirements
History – OSGi R5

- OSGi Resolver Service
  - A new Resolver service is specified
  - Specifies a standard way to provide a resolver to find a resolution solution for generic capabilities and requirements

- One very unhappy implementation
  - No plans to implement the new OSGi Resolver service with the current code base
  - We need to fix this!
Equinox – Moving Forward

- Continually more difficult to provide the Reference Implementation
  - Equinox code base is complicated with too many layers and abstractions
  - Can be hard to prototype new ideas, particularly in resolution and wiring
  - Java Modularity is looming

- Many issues with thread safety have been exposed
  - Inevitable given the amount of abstractions and no clear design path for an overall locking strategy

- Embedding Equinox has issues
  - Configuration is stored and accessed statically and globally, similarly to system properties in Java.
Equinox – Moving Forward

- **Equinox Resolver**
  - Dependencies are 'strongly' typed and are not modelled internally with the OSGi generic dependency model
  - An afterthought to the whole framework implementation
  - Thread safety was not well thought out from the start
  - The Equinox resolver was never designed as the 'core' of the framework
Equinox – Moving Forward

- Equinox Resolver
  - Dependencies are 'strongly' typed and are not modelled internally with the OSGi generic dependency model
  - An afterthought to the whole framework implementation
  - Thread safety was not well thought out from the start
  - The Equinox resolver was never designed as the 'core' of the framework
  - Need to stop using it in the Framework implementation!
OSGi Generic Dependency Model

- Environment – a container where resources can be installed
- Resource – an abstract artifact that can provide some functionality when installed
OSGi Generic Dependency Model

- Environment – a container where resources can be installed
- Resource – an abstract artifact that can provide some functionality when installed
- Capability – a description of some function provided by a Resource
OSGi Generic Dependency Model

- Environment – a container where resources can be installed
- Resource – an abstract artifact that can provide some functionality when installed
- Capability – a description of some function provided by a Resource
- Requirement – an assertion on the availability of a capability in the Environment
OSGi Generic Dependency Model

- Environment – a container where resources can be installed
- Resource – an abstract artifact that can provide some functionality when installed
- Capability – a description of some function provided by a Resource
- Requirement – an assertion on the availability of a capability in the Environment
- Namespace – defines the meaning for the Environment when a Requirement matches an available Capability
- Wire – connects a matching capability to a requirement
Equinox – Redesign the Core

- Implement a core module container
Equinox – Redesign the Core

• Implement a core module container
• Responsible for managing resources
Equinox – Redesign the Core

• Implement a core module container
• Responsible for managing resources
  • Lifecycle
Equinox – Redesign the Core

- Implement a core module container
- Responsible for managing resources
  - Lifecycle
  - Metadata – Capabilities and Requirements
Equinox – Redesign the Core

• Implement a core module container
• Responsible for managing resources
  • Lifecycle
  • Metadata – Capabilities and Requirements
  • Resolution
Equinox – Redesign the Core

- Implement a core module container
- Responsible for managing resources
  - Lifecycle
  - Metadata – Capabilities and Requirements
  - Resolution
  - Persistent settings
Equinox – Redesign the Core

- Implement a core module container
- Responsible for managing resources
  - Lifecycle
  - Metadata – Capabilities and Requirements
  - Resolution
  - Persistent settings
- Uses an OSGi R5 Resolver service
  - Currently Apache Felix
Equinox – Redesign the Core

- Implement a core module container
- Responsible for managing resources
  - Lifecycle
  - Metadata – Capabilities and Requirements
  - Resolution
  - Persistent settings
- Uses an OSGi R5 Resolver service
  - Currently Apache Felix
- Concurrency in mind from the start
  - Properly handle call outs (hooks) while holding no locks
- Tools may use it for modeling OSGi wiring (PDE)
Equinox – Redesign the Core

- Implement a core module container
- Responsible for managing resources
  - Lifecycle
  - Metadata – Capabilities and Requirements
  - Resolution
  - Persistent settings
- Uses an OSGi R5 Resolver service
  - Currently Apache Felix
- Concurrency in mind from the start
  - Properly handle call outs (hooks) while holding no locks
- Tools may use it for modeling OSGi wiring (PDE)
- Does not deal with bundle content, class loading, service registry, event listeners, security
Equinox – Redesign the Framework

Unhappy Equinox

Happy Container

I care about that big X

bundle content
class loading
service registry
event listeners
security
Equinox – Redesign the Framework

- Uses the container as the core
- Keeps complicated concurrency concerns in one place
  - Uses reentrant read/write lock for the module 'database'
  - For long running operations optimistic locking is used. For example, resolution.
Equinox – Redesign the Framework

• Uses the container as the core
• Keeps complicated concurrency concerns in one place
  • Uses reentrant read/write lock for the module 'database
  • For long running operations optimistic locking is used. For example, resolution.
• Implements Framework bits outside of the container
  • Bundle content
  • Class loading
  • Service registry
  • Event listeners
  • Security

Happy Equinox
Equinox – Configuration

• Don't access config statically
• Get rid of static class FrameworkProperties
• Instead make available an instance of EquinoxConfiguration
• Greatly simplifies and optimizes embedding and starting multiple Framework instances
  • A special class loader was needed to isolate static config data for each framework instance

Happy Equinox
Equinox – Hooks

• Equinox Hooks are still supported
  • Much of the internal details changed requiring an overhaul of the Equinox Hooks SPI
  • ALL existing Equinox Hook implementers are broken and will need migration

• Unhappy Equinox had too many Hooks
  • Three class loading hooks – now one, but no ability to override bundle class loader implementation
  • Bundle File hook – removed, use wrapper hook instead
  • Adaptor hook – removed, no more adaptor
  • Bundle Watcher – removed, was a glorified bundle listener
  • Added ability to specify a bundle activator that is called during framework start/stop
  • Improved Storage Hook to handle hooks being added/removed better
Equinox – Old State/Resolver

• Equinox Container has no implementation of old Equinox resolver API
  • Still exports the Equinox resolver API
  • No longer registers the PlatformAdmin service
Equinox – Old State/Resolver

- Equinox Container has no implementation of old Equinox resolver API
  - Still exports the Equinox resolver API
  - No longer registers the PlatformAdmin service

- Equinox Compatibility fragment
  - Hooks into Equinox to register a PlatformAdmin implementation
  - Attempts to mirror the Equinox wiring state in the PlatformAdmin implementation
  - Has some limitations modeling the running state
  - Useful for offline modeling, for example: PDE
Equinox – What Breaks?

- All Equinox Framework Hook implementations must be migrated
- Some limitations for the compatibility resolver
  - Disconnected from the running wiring state
  - Cannot be used to diagnose resolution errors
  - Work to be done to provide better diagnosis of resolution issues in the core container
  - Cannot be used to disable bundles at runtime
Equinox – Current Status

• Work is being done in the git twatson/container branch
  • http://git.eclipse.org/c/equinox/rt.equinox.framework.git/log/?h=twatson/container
  • http://git.eclipse.org/c/equinox/rt.equinox.bundles.git/log/?h=twatson/container

• Plan is to have the implementation ready to drop in for Luna M1

• With CBI we should be able to run full Eclipse/Equinox builds with the new framework

• Successful if nobody notices
  • Except for the poor Equinox Adaptor Hook implementors
Summary

• Years evolving and prototyping the OSGI specification in Equinox have resulted in an unhappy implementation
• Need to redesign Equinox to more closely align with the specification design
  • Based on the OSGi generic dependency model
• Remove unnecessary abstractions
  • Remove Framework Adaptor
  • Remove old Equinox Resolver
  • Simplified Equinox Hooks
• Make the Equinox implementation Happy
Give Feedback on the Sessions

1. Sign In: www.eclipsecon.org

2. Select Session Evaluate

3. Vote +1 0 -1
Trademarks

IBM and WebSphere are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.