Equinox Project Update

Jeff McAffer,
Senior Technical Staff Member
IBM Software Group
Eclipse RCP and Equinox Lead
What is Equinox?

• Small, performant OSGi R4.1 framework implementation
• Collection of service implementations
  • Standard OSGi services
  • Eclipse services (e.g., Extension Registry)
  • Others…
• Server side infrastructure
• Provisioning infrastructure
• Security infrastructure
Write Once, Run Anywhere is a Lie!

• Perhaps true across machines
• Ironically not true across Java™ “editions”
• Java ME, SE and EE all have different programming models
  • Midlets, Applets, Servlets, EJBs, …
• The same program does not have a hope of running everywhere
Enter Equinox and OSGi™

• OSGi, the Dynamic Module System for Java™
• Originally targeted at embedded market
  • Set-top boxes, appliances, then automotive etc

• Eclipse quietly adopted OSGi as core runtime in 2003 (Eclipse 3.0)
• Eclipse RCP evolution was powered by Equinox
  • Dynamic modularity
  • Improved API and lifecycle definitions
  • Smaller footprint
  • Specified, standardized runtime ⇒ wider appeal
  • High performance classloading
The Emergence of Eclipse as a Runtime

- RCP was a “hidden gem” in the Eclipse Tooling Platform
- Equinox emerging as a powerful, general-purpose runtime

One component model across all execution environments

- Extremely appealing and already showing up in products
- Today Equinox is:
  - The reference implementation for OSGi R4.1 framework and JSR 291
  - Well-tooled by PDE
  - Widely adopted
Servers
Equinox on the Server

- Server-side Symposium at ESE 2006
- Use Equinox to manage and run traditional servlets and JSPs
  - Then a new and interesting idea
  - Now shipping in Eclipse 3.3

- Two main deployment forms
  - Embedded Equinox
  - Embedded Server
Traditional Server Example

- Server function (e.g., servlets) packaged in a WAR
- Application Install/Update/Manage whole WARs
- Application isolation
- No OSGi

![Diagram of Traditional Server Example](image)
Embedded Equinox: Equinox in an App Server

- Bridge servlet hosts Equinox in traditional App Server
- Application isolation
- Integration with existing infrastructure
- Forwarding (Lite) HTTP Service
  - Expose underlying App Server capabilities
- Add application function as bundles or servlets or JSPs, ...
- Install/Update/Manage “WAR” by managing bundles
Embedded Server

- Run application server in Equinox directly
- **Process** isolation
- HTTP Service (e.g., embedded Jetty bundle)
- Add application function as bundles or servlets or JSPs, …
- Install/Update/Manage server by managing bundles
- Web Services

![Diagram of Embedded Server](image-url)
But that’s not the end…

- It is really just the beginning
- Eclipse as a runtime expanding
  - Jochen Hiller WAR support (OSGi RFC 66)
  - RAP
  - Swordfish
  - Riena
  - EclipseLink
  - Others…

- Lots of other talks and people around for the rest of the conference
Commercial Adoption

- Major enterprise software providers are shipping or have plans or OSGi/Equinox-based products:
  - IBM WAS 6.1
  - BEA mSA
  - Oracle
  - Interface21 (Spring)
  - IBM Rational Jazz
Provisioning
What to do with all these components?

- Eclipse has shown the value of components
- Small, medium and large organizations are building complex solutions
- Delivery and management is a challenge
Pause for Entertainment
Update Manager

- Originated when Eclipse was static, tooling, …
- Since then Eclipse has become
  - Dynamic, RCP, OSGi, server, embedded, enterprise, …
- Tendency towards coarse-grained “products” via features
- Rigorous end-user workflow

- We never ate our own dog food
Enter Equinox p2

- Provisioning Platform? Provisioning 2.0?
- Equinox Incubator effort destined Eclipse 3.4 (Ganymede)

- Goals for p2:
  - Functionally equivalent replacement for UM
  - Solution must scale down and up
  - Lightweight, dynamic provisioning choices and selections
  - Standards based, standards influencing or *de facto* standard
  - Easy API
  - Easy User Experience
  - Powerful underpinnings
Brief Terminology Lesson

- **Agent**
  - The provisioning infrastructure on client machines
- **Installable Unit (IU)**
  - *Metadata* that describes things that can be installed/configured
- **Artifact**
  - The actual content being installed/configured (e.g., bundle JARs)
- **Repository**
  - A store of metadata or artifacts
- **Profile**
  - The target of install/management operations
- **Director**
  - The decision-making entity in the provisioning system
- **Engine**
  - The mechanism for executing provisioning requests
- **Touchpoint**
  - The part of the engine responsible for integrating the provisioning system to a particular runtime or management system
Inside the Agent

- Director
  - Provisioning operation requested
  - Metadata fetched and constraints analyzed

- Transports
  - Http/Https
  - File system
  - Volume
  - Data transfer

- Repositories
  - p2
  - Update Site
  - Artifact availability and mirroring

- Engine
  - Eclipse/OSGi
  - Native/OS
  - IUs configured into runtimes

- IUs configured into runtimes

- Profiles

- Profiles
The Eclipse Touchpoint

Eclipse/OSGi Touchpoint

Profile 1
IU1, IU2, ...
Settings

Profile 2
IU1, IU2, ...
Settings

Profiles map onto Eclipse configurations

Bundle Pool

Bundle Pool shared by all configurations

Configuration 1
config.ini
eclipse.ini
bundles.txt
...

Configuration 2
config.ini
eclipse.ini
bundles.txt
...

Engine

Profiles map onto Eclipse configurations
Scenarios

Eclipse Classic

Eclipse for C++

Client Agent

Full central management of client profiles
Things To Look For
Security

- Signature-based code authorization solutions
- Trade-off security/complexity vs. performance
- Install-time (current), bundle load-time, and code run-time (Java2 permissions)
- Integrated user authentication framework
  - Platform login configuration and lifecycle
- Mechanisms for user credential management
  - Manage trusted roots, private keys, passwords, etc
- Use Java security provider architecture as appropriate
  - KeyStore, CertStore, etc.

- Participate in the Equinox Security Incubator
  - http://eclipse.org/equinox/incubator/security/
Component Programming Models

- Decoupled components need to be composed to form solutions
- Several composition mechanisms
  - Eclipse Extension Registry
  - OSGi Services and Declarative Services (DS)
  - Spring for OSGi
  - Service Activator Toolkit (SAT)
  - iPOJO
  - …
- All are subtly different and all have pitfalls
- Developer left holding the bag

- Look for synergies and compatibility/interaction stories
- Seek to codify in OSGi
OSGi Enterprise Expert Group

- OSGi moving to enterprise computing
- Enterprise Expert Group working in several areas
  - Distributed/federated service model
  - Persistence
  - JEE interop
  - Enterprise-class life-cycle and configuration management
  - Scalability
- Lots of interesting discussions
API Tools

- Components are defined by their API so tooling is very important
- Backward compatibility
  - Is version X binary compatible with version X-n
  - Detect problems ASAP
- Global references
  - Who’s using my code?
  - What am I using?
- Version management
  - Ensure code changes are properly reflected in version numbers
- API refinement
  - Public/private, API/internal is not enough
  - SPI, client implementable, client callable, …
- Participate in the PDE API Tooling Incubator
  - http://wiki.eclipse.org/PDE_UI_Incubator_ApiTools#Use_cases
Equinox: Powering Components

• The same power of Eclipse brought to:
  • Desktop
  • Client
  • Server
  • Embedded

• One component model across all execution environments
Further Info

- **Equinox BoF Wed @ 1930**
- http://eclipse.org/equinox/

- Provisioning Symposium
- Server-side Symposium

- http://eclipse.org/equinox/server
- http://eclipse.org/equinox/incubator/provisioning

- http://eclipse.org/mayninstall/ (talk Wed @ 1700)
- http://eclipse.org/ecf/ (talk Wed @ 1540)
- http://eclipse.org/rap/ (talk Wed @ 1540)
- http://eclipse.org/swordfish/
- http://eclipse.org/proposals/riena/ (talk Wed @ 1730)
- http://eclipse.org/ercp/ (talk Thur @ 1000)