Developing a Platform

Pawel Pogorzelski
epawpog
Eclipse 3.7 Indigo

› Released on 22/06/2011
  - 62 project teams participated in Indigo
  - 46 million lines of code released on the same day
  - 408 committers contributed code
  - 49 organizations collaborated

› Eclipse a leader
  - for java client development
  - gaining more ground on the server side
Transparency

› Eclipse Foundation is an independent company
  – well defined, transparent rules
› All the decisions made in public
› All the work done though Bugzilla
Stable rythm

› Shipping end of June each year (for 10 years)
› 7 milestones
  - 6-7 weeks each
  - M6 API freeze (this year March 18)
  - M7 feature freeze (this year May 6)
  - RC1 (this year May 20) – RC4 (this year June 10)
› Weekly integration builds
› Nightly builds
› Throwing out features rather than slipping
Technologies

› Lazy loading
› OSGi
› JDT
› PDE

› Technologies to build on
  - extensibility
API

› Stable API
  − one of key factors Eclipse is so successful
  − around 500 products build on Eclipse only in IBM
  − other 5000 with direct and indirect dependencies
  − huge focus on wise and stable API

› API plays nice with OSGi
  − non API is not referable
  − versioning used to specify requirements
  − multiple API versions coexisting in one JVM
What is API?

 › org.eclipse.xyz.*
 › org.eclipse.xyz.internal.*
 › org.eclipse.xyz.internal.provisional*

 › API class or interface
   - a public class or interface in an API package, or a public or protected class or interface member declared in or inherited by some other API class or interface

 › Similarly for methods and fields
API Tools

› Tools designed to facilitate API evolution

› Features
  - update version numbers (Eclipse versioning scheme)
  - identify binary compatibility issues
  - identify usage of non API packages
  - identify internal types leakages into API
  - provide a set of additional markers
Setup

› The baseline
  - new API developed based on the last released

› Add the builder

› You're ready to go
API Baseline

› Defines the what to compare workspace bundles against
› Set to 3.7 at the beginning of 3.8 stream
Configuring bundles

› Add API
Configuring bundles

![API Tools Setup window](image)

**API Tools Setup**

Configure projects for API usage and compatibility checks.

Select the Java projects to update.

Choose an element (? = any character, * = any string)

Projects:

- org.eclipse.core.resources.compatibility

- [ ] Select All
- [ ] Deselect All

1 item(s) checked.

- [ ] Delete component.xml after update is complete

Next > Finish Cancel
Configuring bundles

![New Plug-in Project dialog box](image)

- **ID:** org.eclipse.pde.api.tools.demo
- **Version:** 1.0.0.qualifier
- **Name:** org.eclipse.pde.api.tools.demo
- **Provider:**
- **Execution Environment:** JavaSE-1.6
- **Generate an activator, a Java class that controls the plug-in’s life cycle:**
  - **Activator:** org.eclipse.pde.api.tools.demo.Activator
- **This plug-in will make contributions to the UI:**
- **Enable API Analysis**
Configuring bundles

› As a result **API Analysis Builder** is added
Version numbering

› Bundle versioning:

```
org.eclipse.core.resources_3.7.100.v20110510-0712.jar
```

pluginname_major.minor.service.qualifier

› The same for packages:

```
org.osgi.framework;version="1.6",
org.osgi.framework.hooks.bundle;version="1.0",
org.osgi.framework.hooks.resolver;version="1.0",
```
Major segment

» Incremented when API breaking changes occurs

   * @see IFile#create(java.io.InputStream, int, IProgressMonitor)
   * @see #VIRTUAL
   * @since 3.6

```java
public boolean isVirtual(int options);
```

Problem: Adding an argument for API interface

Fix: Change plug-in version from 3.7.100 to 4.0.0

» Maintenance branch: not possible
Minor segment

› Increased for "externally visible" changes
  - binary compatible API changes
  - significant performance changes
  - major code rework

› Still not decided?
  - if too big to consider bug fixes but not breaking

› Maintenance branch
  - not recommended
  - known to consumers in advance
  - 1.0 release – version 2.2.7
  - 1.0.1 release – version 2.3.0
  - 1.1 release – version 2.4.0
Minor segment

**Import-Package**: org.eclipse.xyz;\textit{version}="[1.6.0,2.0.0)"

**Export-Package**: org.eclipse.xyz;\textit{version}="1.5.0"

```java
public class IMessaging {
    public String sendRequest(String request); 
}
```
Service segment

› Changes not visible in API
  - bug fixes
  - the plug-in manifest has changed
  - documentation
  - compiler settings have changed

› Make room for maintenance
  - 3.0 release – version 2.4.8
  - 3.0.1 release – 2.4.9
  - 3.1 first fix – version 2.4.108
Qualifier

› Usually composed of timestamp
  ‒ derived from build input

› Enables teams to use p2 during the development cycle

› Eclipse format
  ‒ vYYYYMMDD

› Having multiple builds a day?
  ‒ vYYYYMMDD-HHMM

› Use different qualifier for maintenance streams
  ‒ R20x
  ‒ be consistent with the prefix
  ‒ 1.0.1.R20x_v20090629 may appear in both 1.0.1 and 1.0.2
Evolving version numbers

- First dev stream
- Second dev stream
- Third dev stream
Specifying requirements

› Including all non breaking updates from 3.7
  - “[3.7.0, 4.0.0)“

› Only non visible changes
  - “[3.7.0, 3.8.0)“

› No changes
  - “3.7.0“
Binary vs Source

› Difference between JC and JVM

› Updating sources not a problem
› You care about not breaking binaries
› Decouples the system
  - components can be compiled independently
  - important in multi vendor environment
Binary vs Source

```java
public interface IMessaging {
    public static final int ASYNC = 0x1;
    public static final int CONTROL = 0x2;
    public String sendRequest(String request);
    public String sendRequest(String request, int timeout);
}
```
API restrictions

› @noimplement (Interfaces)
› @noextend (Classes, Interfaces)

```
* @noimplement This interface is not intended to be implemented by clients.
* @noextend This interface is not intended to be extended by clients.
*/

public interface IResource extends IAdaptable, ISchedulingRule {
```

› @noinstantiate (Classes)
› @nooverride (Methods)
› @noreference (Methods, constructors, fields)
Adjusting Problem Severities

The image shows a window for adjusting problem severities in software development. The window includes options for API Use, API Compatibility, Version Management, Analysis Options, and API Use Scans. The API Errors/Warnings section is highlighted, with options to set severities for different types of errors and warnings, such as invalid references to system libraries, supported environments, restrictions on API use, and leaks. The specific severities can be set to Error, Warning, or Ignore. Additionally, there is a note indicating that there are no execution descriptions installed and additional environment descriptions are available to install them now.
Filtering Known Problems

```java
IPath getFullPath(int type);
```

The method `org.eclipse.core.resources.IStorage.getFullPath(int)` in an interface that is intended to be implemented has been added.

2 quick fixes available:
- Create a compatibility problem filter for 'getFullPath(int)'
- Create a commented compatibility problem filter for 'getFullPath(int)'

Press 'F2' for...
Filtering Known Problems

/org.eclipse.core.resources/.settings/.api_filters

<resource path="src/org/eclipse/core/resources/IStorage.java" type="org.eclipse.core.resources.IStorage">
  <filter comment="Planned breakage" id="403804204">
    <message_arguments>
      <message_argument value="org.eclipse.core.resources.IStorage"/>
      <message_argument value="getName(int)"/>
    </message_arguments>
  </filter>
  <filter id="1211105284">
    <message_arguments>
      <message_argument value="getName(int)"/>
    </message_arguments>
  </filter>
</resource>
API use scan

› Who is using my API?
› Who refers to internals?

List of bundles using org.eclipse.ant.core

Terminology

1. API References - are references that are only made to public API of a bundle
2. Internal References - are references that are only made to internal non-API of a bundle
3. Internal-Permissible References - are references that are only made to internal non-API of a bundle, but are allowed via the x-friends directive
4. Other References - are references that have been detected but could not be resolved

<table>
<thead>
<tr>
<th>Bundle</th>
<th>API References</th>
<th>Internal References</th>
<th>Internal-Permissible References</th>
<th>Other References</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.eclipse.ant.ui</td>
<td>323</td>
<td>0</td>
<td>62</td>
<td>0</td>
</tr>
<tr>
<td>org.eclipse.pde.build</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>org.eclipse.pde.core</td>
<td>48</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
How does it work?

› API Tools relies on bytecode manipulation
  - generates .class files stubs retaining API information
  - operates on binaries (guarding binary compatibility)
Summary

› Use OSGi to ensure your assumptions are met
› Bundle evolution is complex
   - breaking/not breaking
   - source vs. binary
› Other considerations
   - reexporting packages
   - non API bundles
   - external libraries
   - versioning features
› Solution
   - use PDE API Tools
   - visit http://wiki.eclipse.org/Version_Numbering