Aaron Schultz
Software Developer
National Center for
Ecological Analysis and Synthesis

Kepler Architecture Solution
OSGI Bundles

July 16, 2008

Outline

- OSGi Background
- OSGi R4 Implementations
- OSGi Bundles
- OSGi Fragments
- Exporting & Importing Packages
- OSGi Class Space
- OSGi Extensions
- OSGi Services
- Framework as hosted or host
- Eclipse Plug-in Development Environment
- Possible Usage of Bundles in Kepler
- Reference

OSGi Background

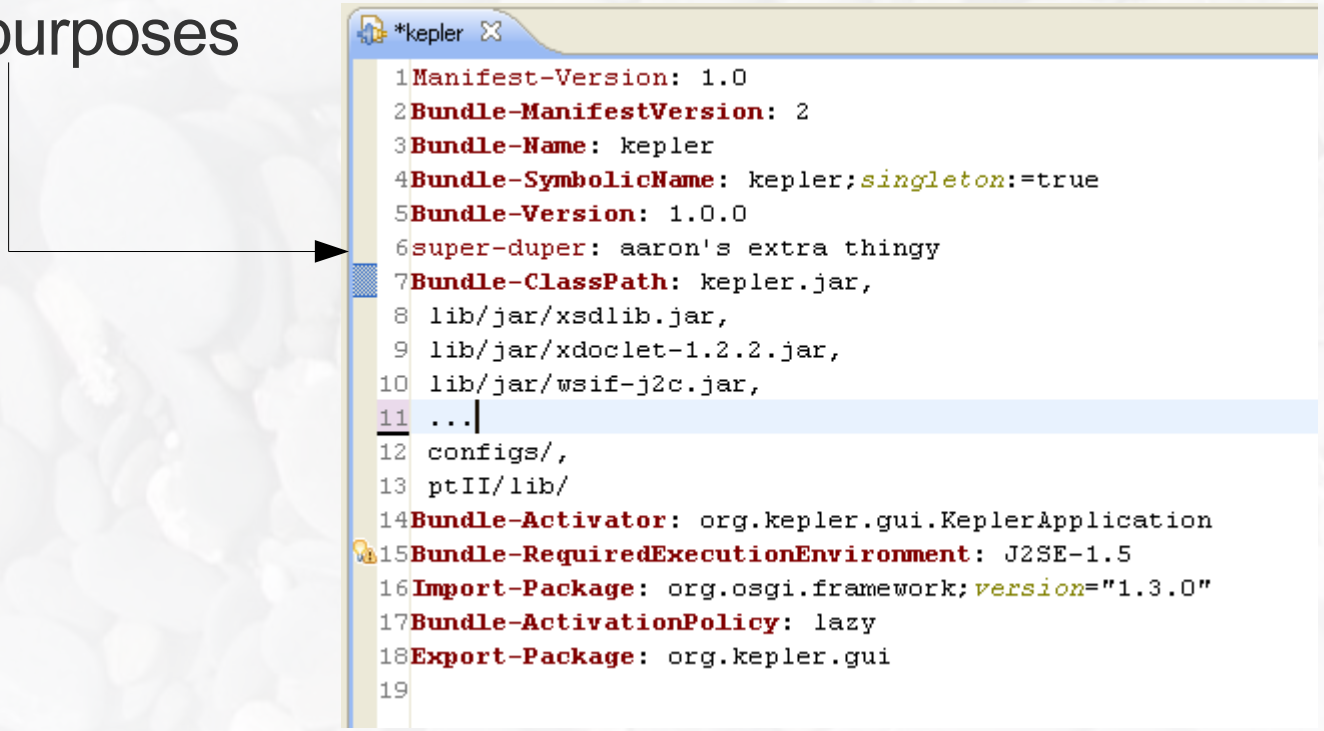
- OSGi Release 1 (R1): May 2000
- OSGi Release 2 (R2): October 2001
- OSGi Release 3 (R3): March 2003
- OSGi Release 4 (R4): October 2005 / September 2006
  - Core Specification (R4 Core): October 2005
  - Mobile Specification (R4 Mobile / JSR-232): September 2006
- OSGi Release 4.1 (R4.1): May 2007

OSGi R4 Implementations

- Certified
  - Eclipse Equinox 3.2 (Apache License v2.0)
    - Default OSGi framework for Eclipse
    - Can be used to host an application or hosted by an application
  - Makewave Knopflerfish 2.0 (BSD license)
    - A Commercial implementation that was opened by Makewave
    - A Pro version is available for purchase
  - ProSyst Software mBedded Server 6.0 (Eclipse Public License)
    - Open version uses the Equinox framework and offers additional bundles
    - Commercial version uses ProSyst framework with many additional bundles
  - Samsung OSGi R4 Solution (Commercial)
  - HitachiSoft SuperJ Engine Framework (Commercial)
- Non-Certified
  - Apache Felix (Apache License v2.0)
    - Does not fully implement the OSGi R4 spec

OSGi Bundles

- OSGi Bundles are JAR files with standardized Manifests
  - Non-standard attributes are ignored by OSGi frameworks and can therefore be used for other purposes



See Section 3.2 “Bundles” of the OSGi version 4.1 sepecification for detail on Manifest Headers and syntax

OSGi Fragments

- Fragments are bundles that are directly associated with a Host Bundle
- Fragments are loaded with the same classloader as the host bundle
- Fragments are often used to store platform specific resources making the inclusion and exclusion of these resources for different platform configurations very easy

Screenshot: PDE Fragment Manifest Editor

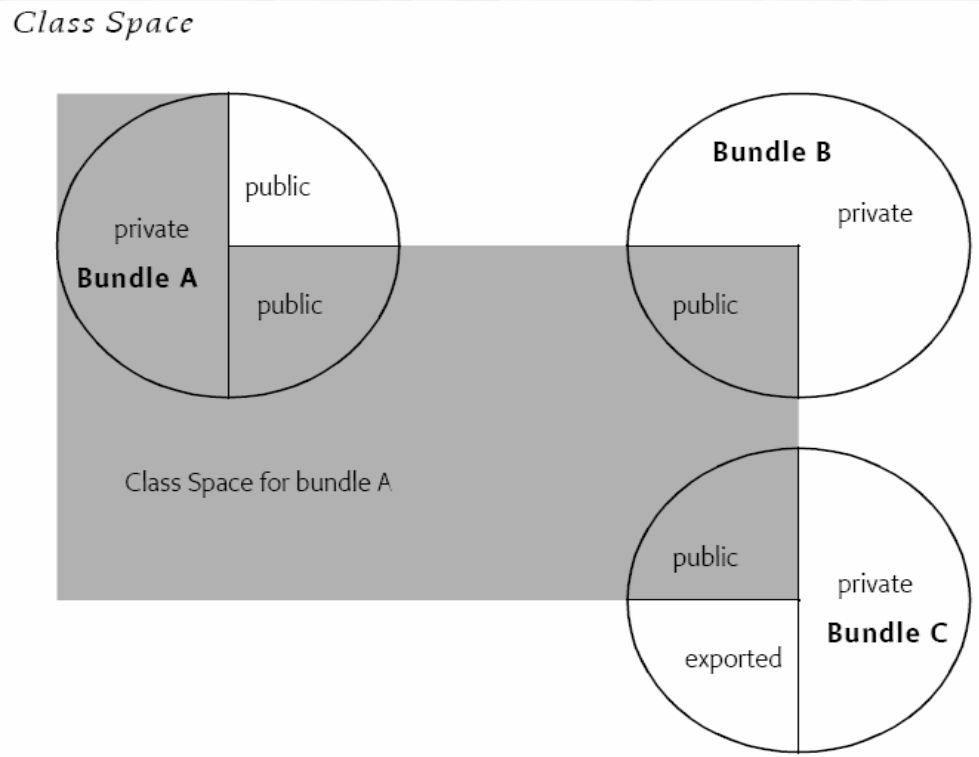
See Section 3.14 “Fragment Bundles” of the OSGi version 4.1 sepecification for details on Fragments

Exporting & Importing Packages

- Exporting
  - A bundle specifies explicitly in the Export-Package manifest attribute which packages are available for use by other bundles (think public/private packages)
  - The usual practice is to simply export (make public) all the packages of a bundle unless there is a good reason not to – the exported packages define the API
- Importing
  - Import-Package attribute allows a specific package to be used
  - Require-Bundle attribute allows all exported packages from the specified bundle to be used

OSGi Class Space

- A class space is the set of all classes that are reachable from a given bundle's class loader
  - The parent class loader (normally java.* packages from the boot class path)
  - The bundle's class path (private packages)
  - Imported packages
  - Required bundles
  - Attached fragments

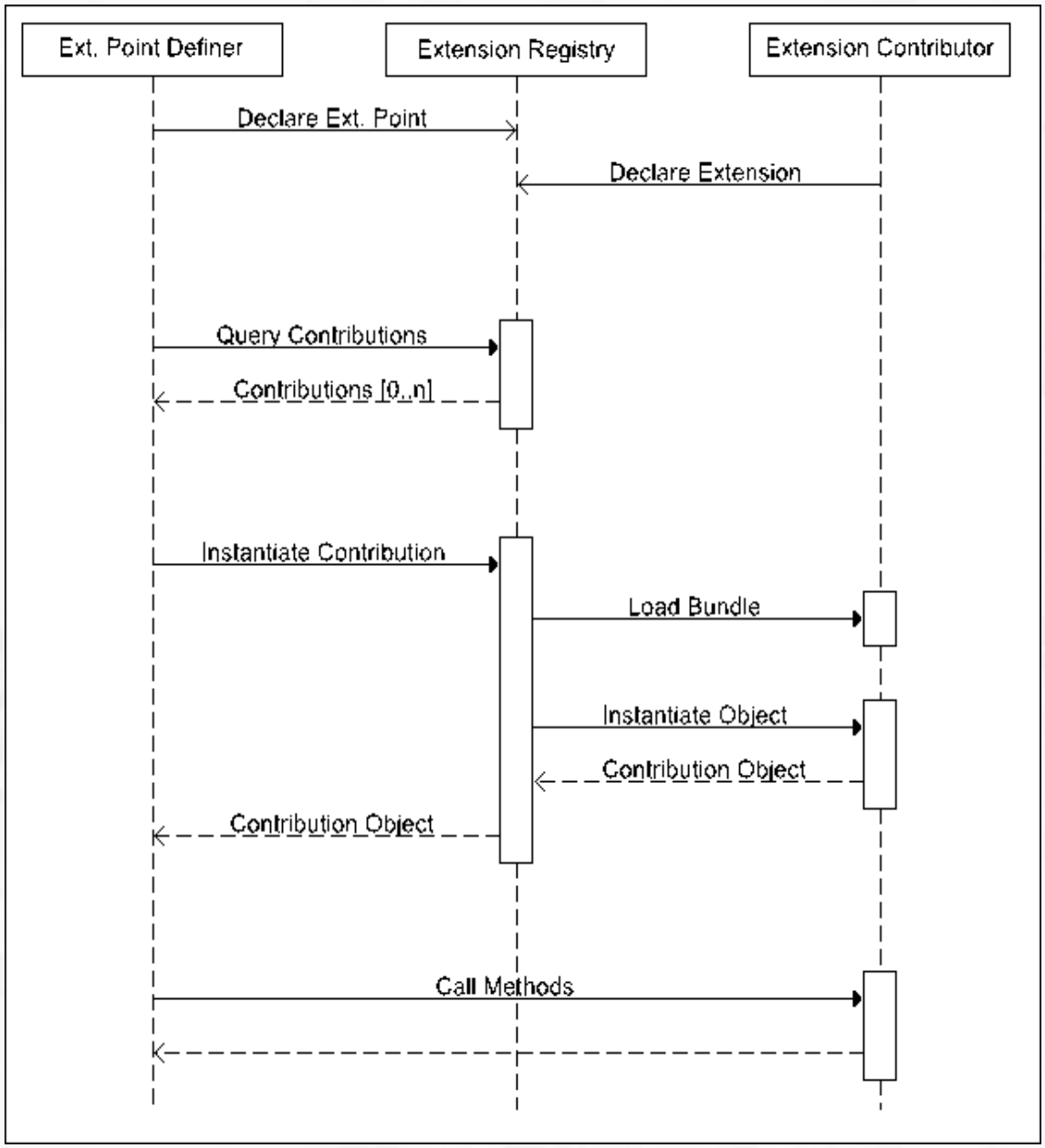


OSGi Version 4.1 Section 3.4 Class Loading Architecture

OSGi Extensions

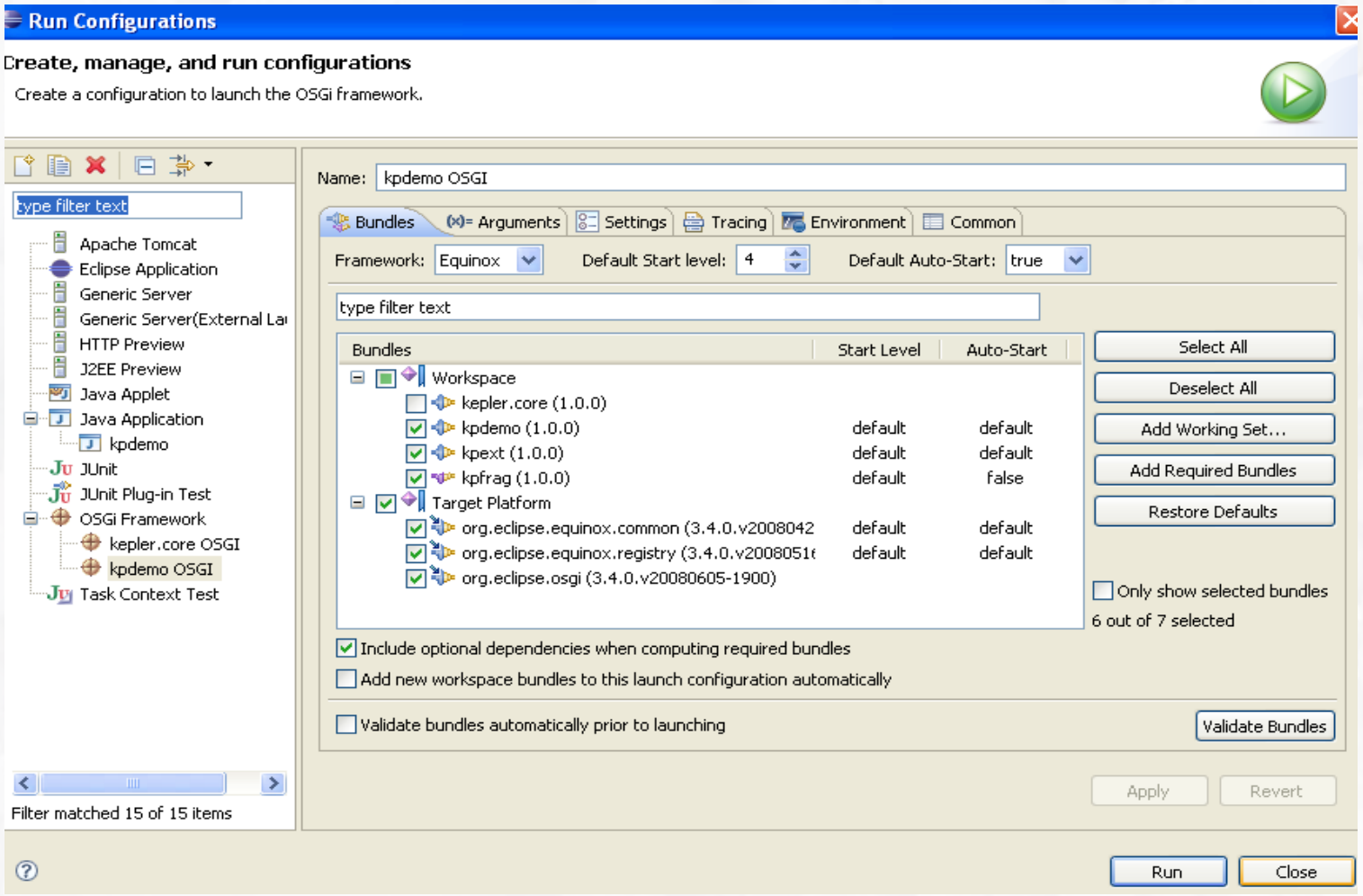
- Plugin.xml
  - Extensions
    - Explicitly declare the extension points from other bundles that we are using in this bundle
  - Extension Points
    - Explicitly define the extension points available in this bundle with a name and an ID
    - Include documentation and examples about each extension point
    - Specify how many times the extension point can be extended (just once or by many different bundles)
- Extensions get managed by the Extension Registry
- Extensions are all registered before any classes are loaded, so there is no worry about timing/ordering of extensions

OSGi Extensions



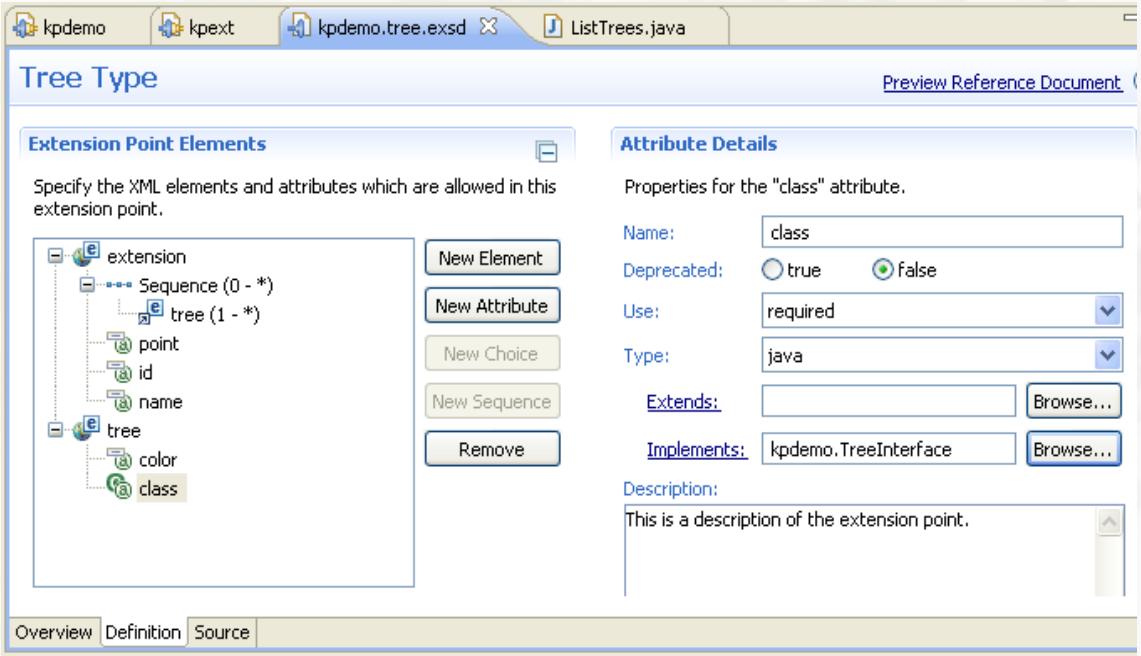
The Equinox Extension Registry

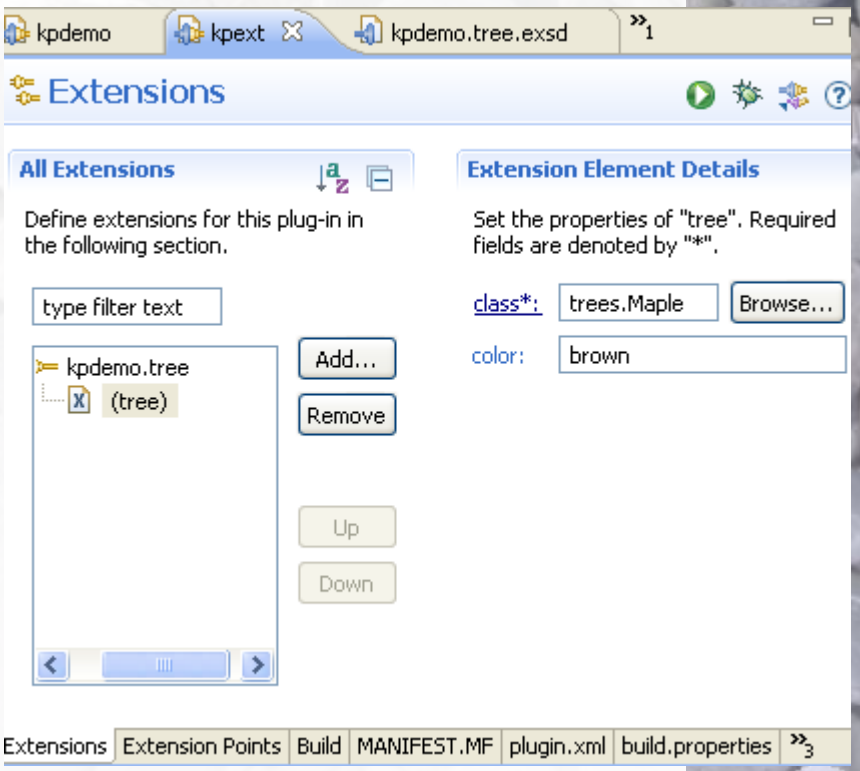
- To use the Equinox extension registry the bundle org.eclipse.equinox.registry needs to be installed in the OSGi configuration



Extension Attributes

- An extension point defines a set of attributes that can be set by implementing extensions
  - Attribute values are available before any classes are loaded
  - If a class in the extension is to be executed the name of the class is stored in an attribute (called class in this example)





Executing Extension Classes

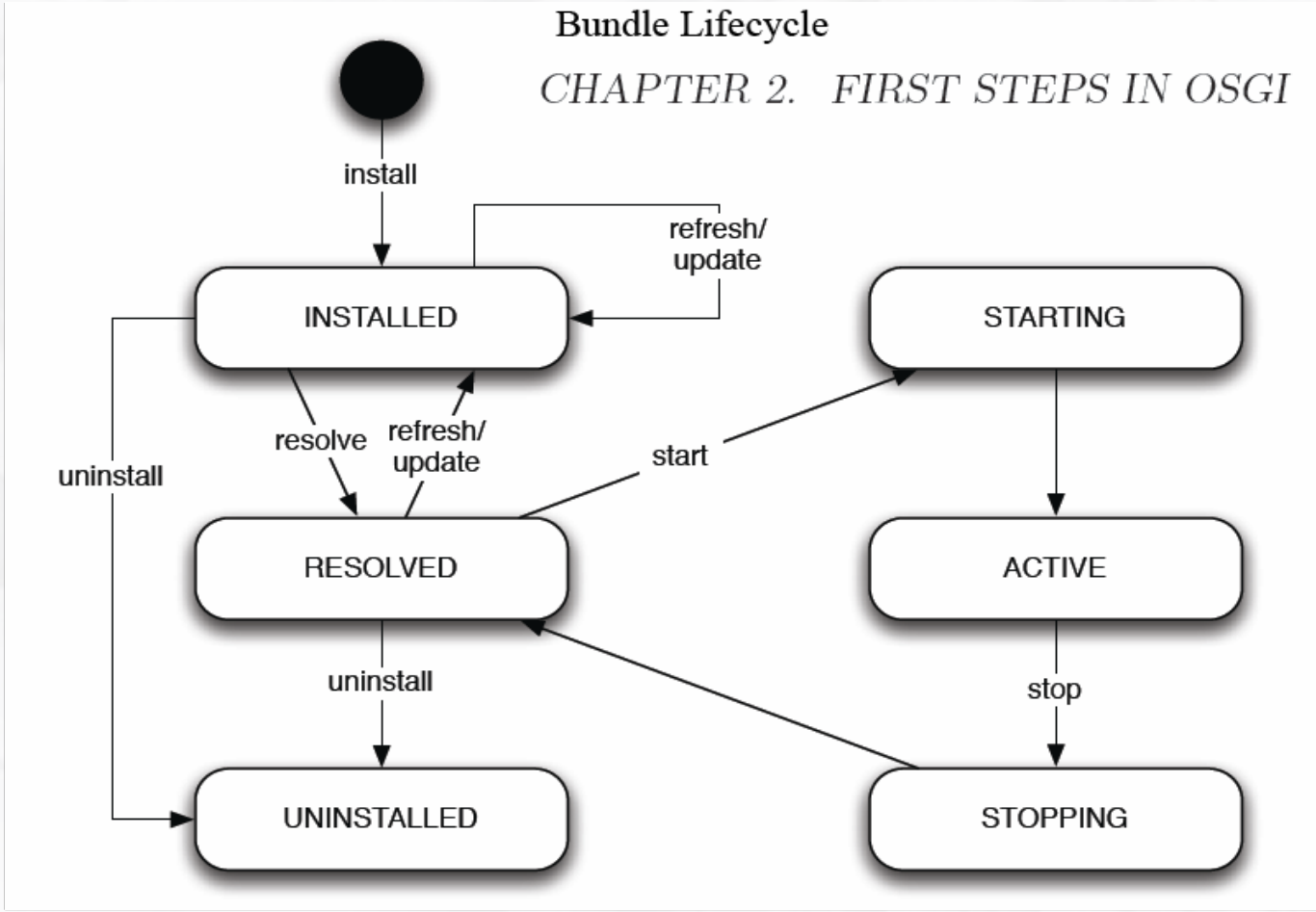
- 3 lines of code are needed in the bundle that defines the extension point to retrieve and execute classes supplied from extensions (lines 8, 9, and 12 in the sample below)
- No special Java code is needed in the extensions that are contributing to an extension point – only the xml definitions are needed

kpdemo | kpext | kpdemo.tree.exsd | ListTrees.java

```
1  package kpdemo;
2  import org.eclipse.core.runtime.*;
3
4  public class ListTrees {
5      public ListTrees() {
6          String treeList = new String();
7
8          IExtensionRegistry reg = RegistryFactory.getRegistry();
9          IConfigurationElement[] extensions = reg.getConfigurationElementsFor("kpdemo.tree");
10         for (int i = 0; i < extensions.length; i++) {
11             try {
12                 TreeInterface ti = (TreeInterface) extensions[i].createExecutableExtension("class");
13                 treeList += "Tree Name: " + ti.getName() + "\n"; // method call to class
14                 if (extensions[i].getAttribute("color") != null) {  // attribute from extension definition
15                     treeList += "Tree Color: " + extensions[i].getAttribute("color");
16                 }
17                 treeList += "\n";
18             } catch (Exception e) {
19                 System.out.println(e.getMessage());
20                 e.printStackTrace();
21             }
22         }
23         System.out.println(treeList);
24     }
25 }
```

OSGi Services

- Services can come and go dynamically during program execution
- Because of this a developer has to take special care to check and handle each state of the bundle lifecycle



Framework as host or hosted

- Framework as Host
  - The OSGi framework is run with a specific configuration of bundles
  - The configuration points to and optionally starts the specified bundles
  - In Eclipse an executable is used to run the Equinox bundle which then reads the configuration and loads all the bundles
- Hosted Framework
  - A running application can call Equinox and load bundles according to a configuration defined by the application

Eclipse Plug-in Development Environment

- Extension schemas and bundle manifests can be edited by hand in text editors although most developers prefer to use the Eclipse PDE
- PDE provides a nice set of forms and automation tools that allow easy creation of bundle manifests, plugin.xml, and extension schemas
- PDE auto detects classpaths and bundle dependencies for you
- PDE is geared towards extension development and does not have strong support for services

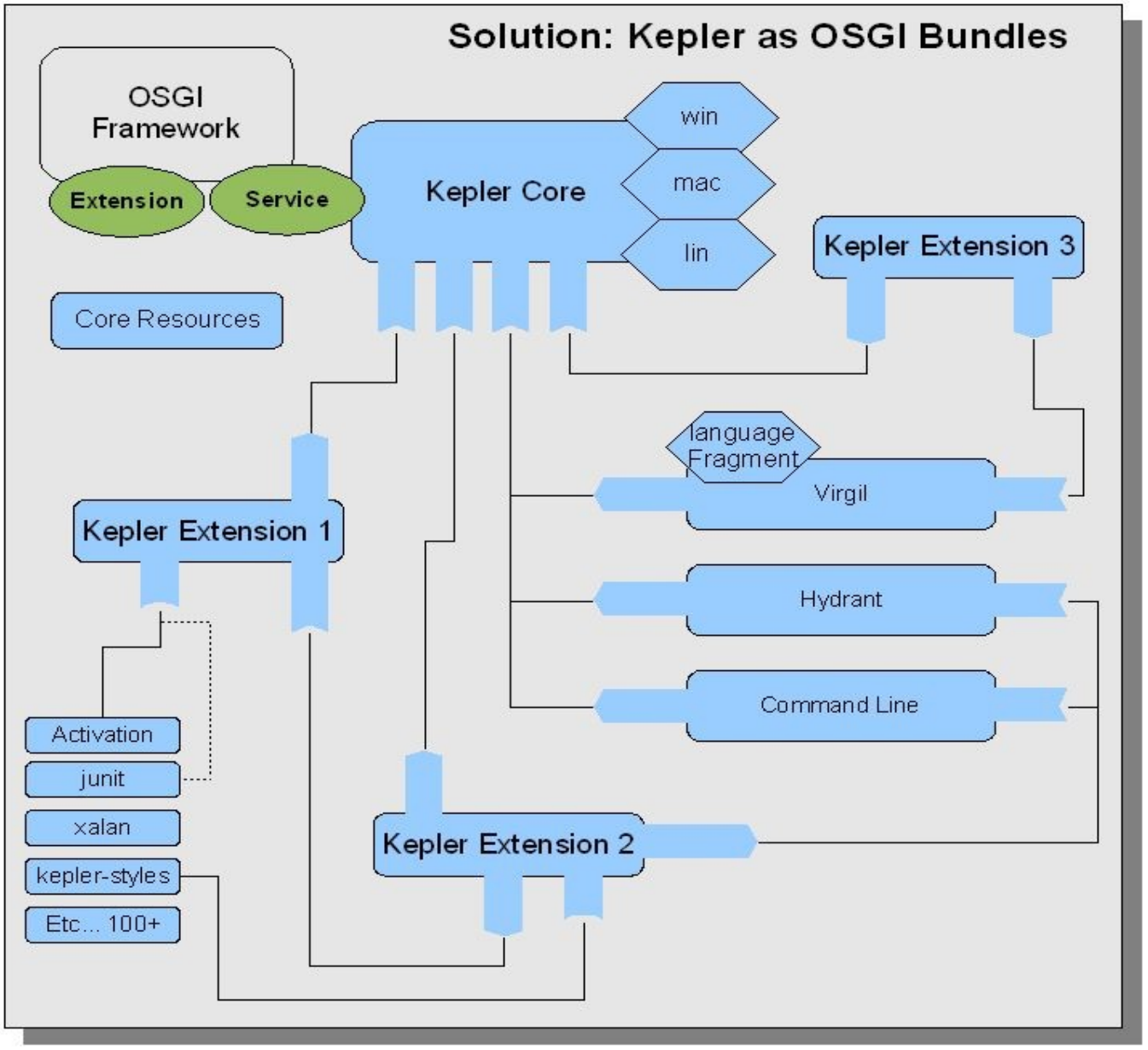
Possible Usage of Bundles in Kepler

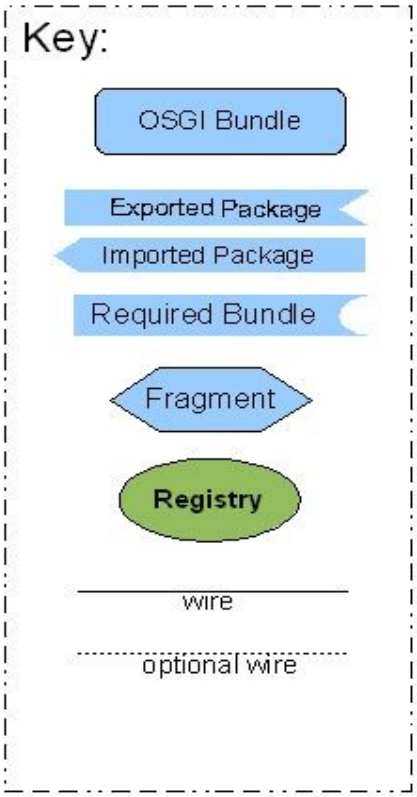
- Nightly build using Maven
- Requirement of not breaking the nightly build for too long
- Possible plan of attack (perhaps not in this order exactly)
  - Start with the existing, monolithic Kepler+Ptolemy code base all in one bundle in one branch
  - Separate out jars
  - Separate out resources
  - Separate out Ptolemy
  - Separate out platform specifics into fragments
  - Separate out GUI
  - Identify and separate out components
- Each bundle would have its own place in repository

Jar Libraries

- Including Jars within Bundles
  - slightly slower
  - packages need to be exported from bundle for use by other bundles
- Including Jars in an OSGi configuration as bundles is the preferred method (although there are other ways see Eclipse FAQ)
  - Currently there are 282 jars in Kepler (not all in use)
    - These would be separated out of the core and repackaged as bundles
      - This involves adding a few OSGi headers to the manifest of each jar which could be easily automated
      - OR it is likely that the jars can be retrieved and built as OSGi bundles from a Maven repository

Very Abstract Diagram





References

- Equinox Portal
- OSGi in Practice (open book uses Felix Framework for examples)
- Certified OSGi r4 implementations (OSGi Alliance)
- Knoplerfish Pro (Makewave)
- ProSyst Edition Comparison
- OSGi Wikipedia page
- Equinox QuickStart Guide
- A Comparison of Eclipse Extensions and OSGi Services
- Getting Started with Eclipse Plug-ins
- Eclipse FAQs
- OSGi Specification Version 4.1
  - Section 3 “Module Layer”