OSGi Alliance

OSGi Alliance Community Event
June 26th - 27th, 2007
Siemens AG Campus - Munich, Germany

OSGi Alliance
Community Event

The Bundle Dilemma

Richard S. Hall
Laboratoire d'Informatique de Grenoble,
Grenoble University, France

Agenda

- The Bundle Dilemma
- History
- OSGi Bundle Repository (RFC 112)
- Apache Felix OBR Implementation
- OBR Bundle Repositories
- Issues
- Conclusion

The Bundle Dilemma

Introduction

- The OSGi framework provides a sophisticated, general modularity mechanism for Java
  - Focus is on keeping the core small
  - Push additional functionality out of the framework
- Huge success at promoting and simplifying the creation of modular Java systems
  - Used in embedded to enterprise domains
- The number of available bundles is growing quickly

The Bundle Dilemma (1/3)

- Its success at being modular results in the ***bundle dilemma***
  - Core is intentionally kept small with developers encouraged to create useful, re-usable bundles, but...
  - Inability of developers to discover and re-use existing bundles
  - Difficulty in deploying existing bundles

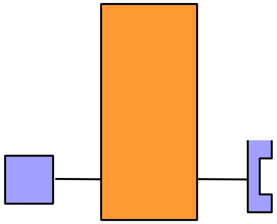
The Bundle Dilemma (2/3)

You want a bundle that provides some functionality...

?

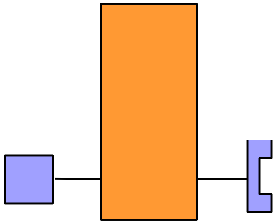
The Bundle Dilemma (2/3)

You find the bundle and install it, but then...



The Bundle Dilemma (2/3)

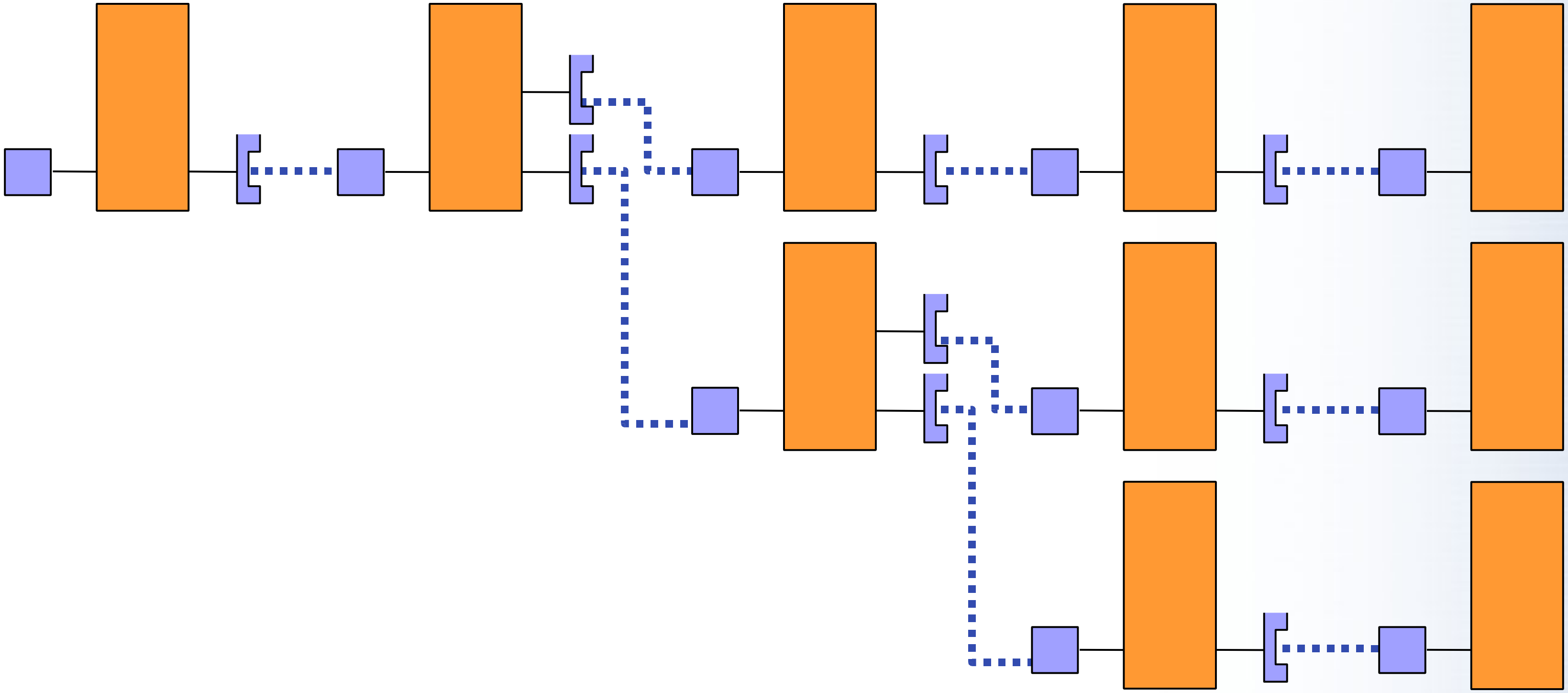
You find the bundle and install it, but then...



```
-> start 4
org.osgi.framework.BundleException:
   Unresolved package in bundle 4: package;
      (&(package=org.apache.felix.foo)
          (version>=1.0.0))
->
```

The Bundle Dilemma (2/3)

You really need to install these too...



The Bundle Dilemma (3/3)

- The OSGi specification is mute on how installed bundles are discovered in the first place
  - Dependency resolution only applies to installed bundles
- This issue can only become more important as the number of bundles increases
- What can be done?

OSGi Bundle Repository (OBR)

- Attempting to define a common (standard?) bundle repository
  - Share
  - Discover
  - Deploy

History

Oscar Bundle Repository (OBR1)

- Initial bundle repository effort started as part of Oscar circa 2003/2004
  - Minimize Oscar download size
  - Provide a repository of bundles for easy deployment into OSGi frameworks
  - Promote a community effort around bundle creation
- Multiple means of access
  - Web page, service interface, shell command
- Low barrier for participation
- Turned out to be more popular than I imagined

OBR1 Issues

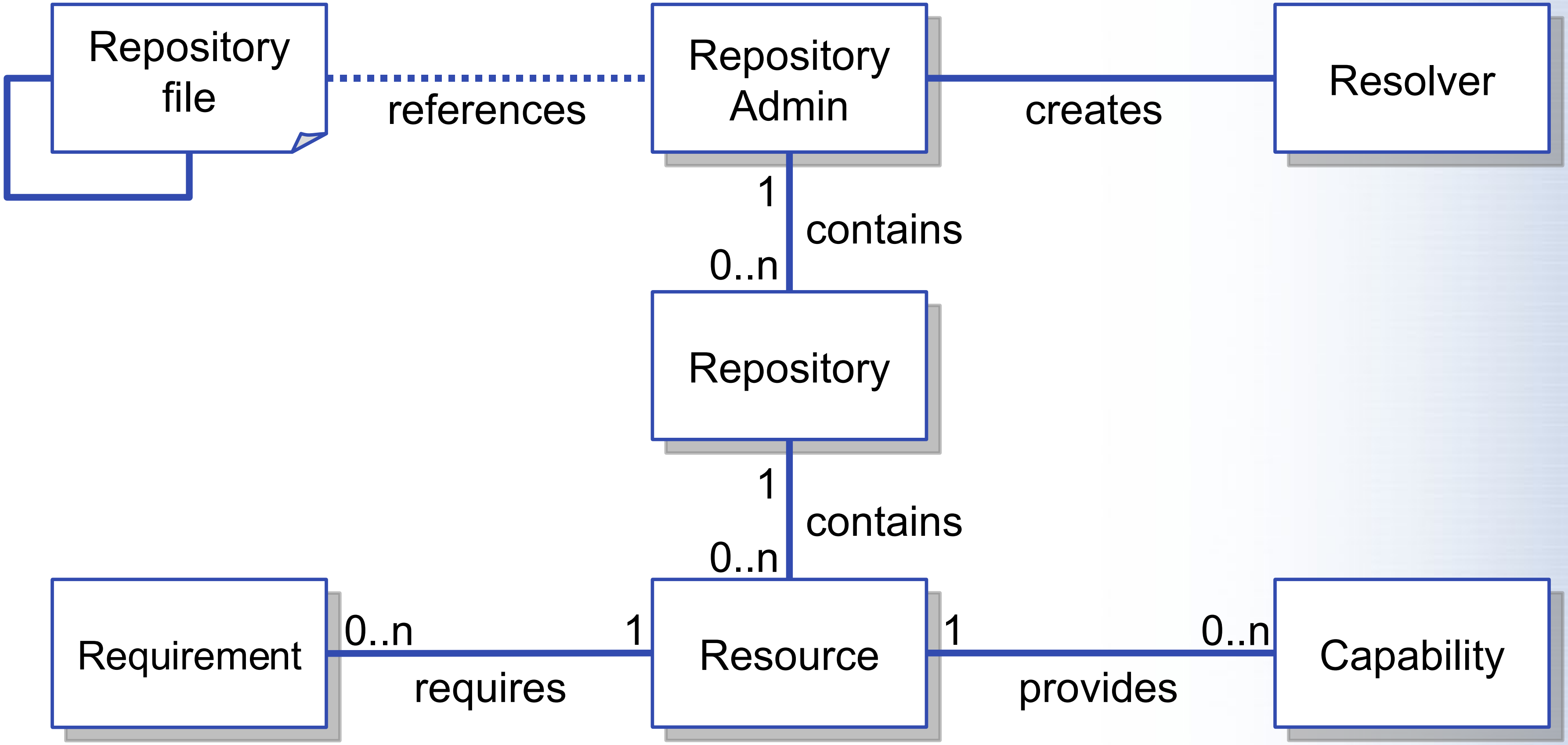
- Too simplistic
  - Only truly supported resolving package dependencies
  - Could not handle multiple versions of packages in the framework and only awkwardly handled multiple versions of bundles in the repository
  - No easy way to diagnose deployment errors
- Started to think about how to deal with these issues in April 2005
  - Wanted to improve version handling
  - Wanted a generic capability/requirement model

OSGi Bundle Repository
RFC 112

OSGi Bundle Repository (OBR2)

- The goals of OBR2 are essentially the same as OBR1, but just doing it better
- Improved (and much debated) generic capability/requirement model
  - XML representation
  - Models package, bundle, fragment, native, and service dependencies (plus arbitrary ones)
- Stronger focus on bundle discovery

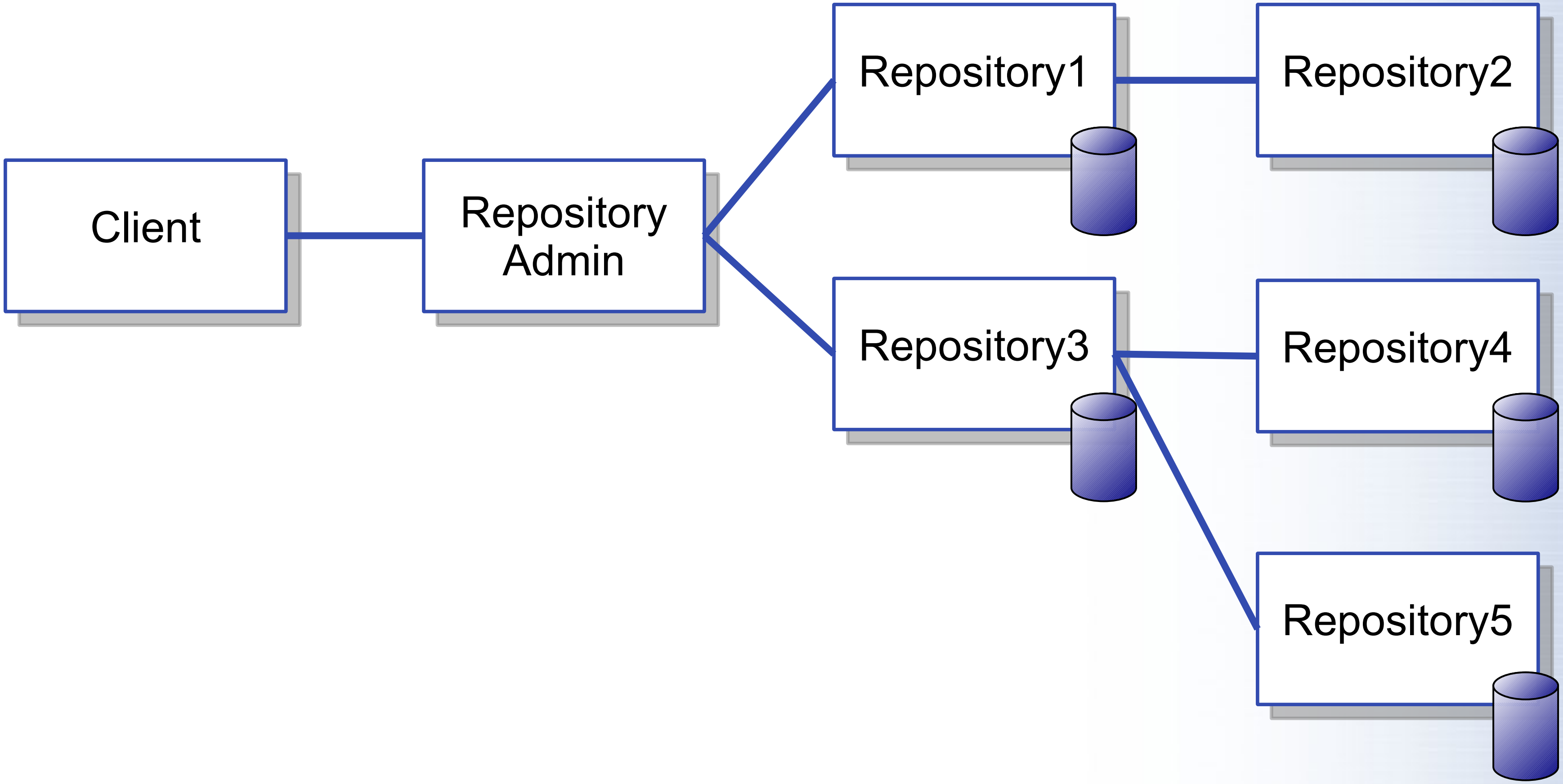
OBR2 Entities (1/2)



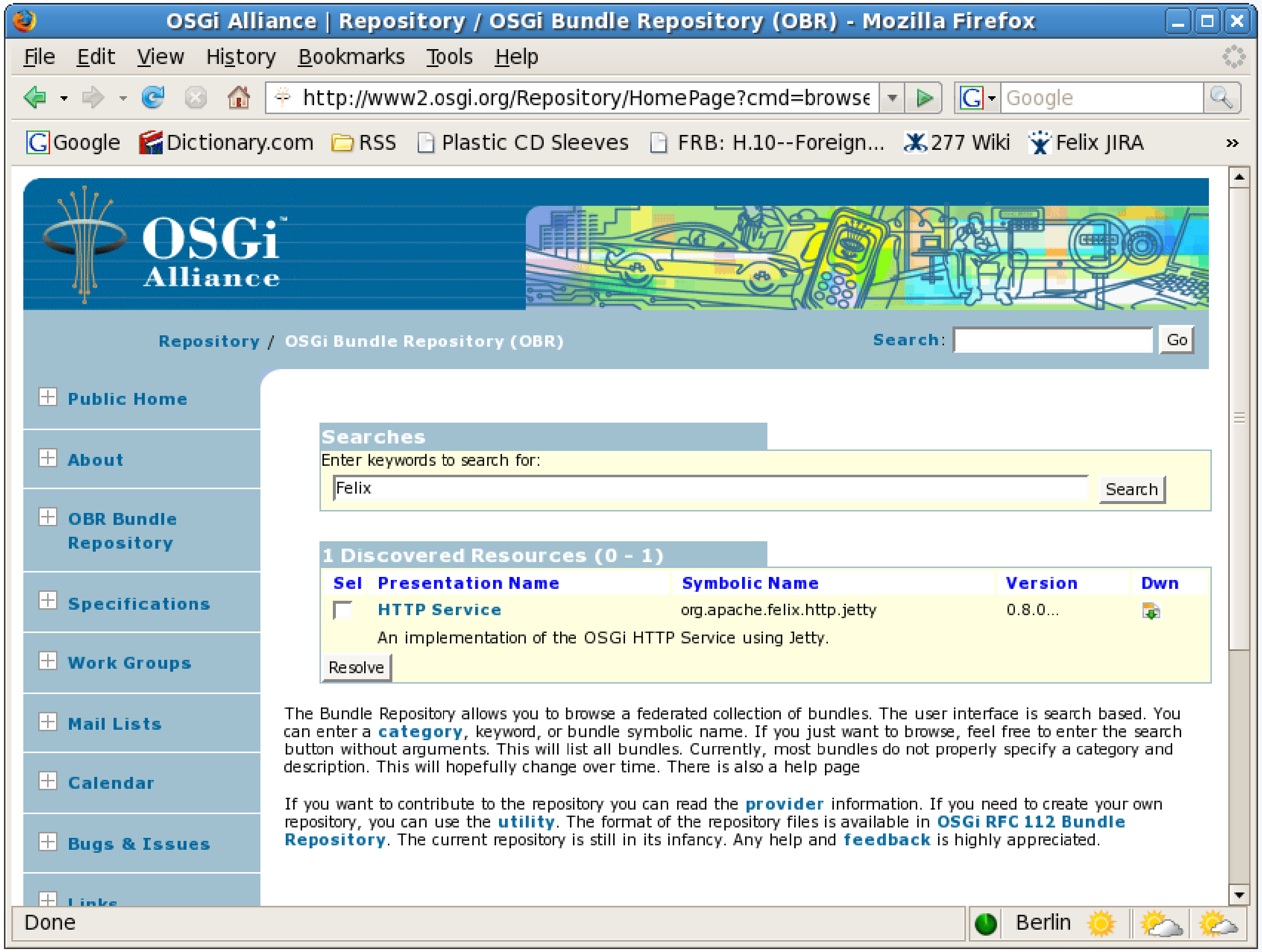
OBR2 Entities (2/2)

- **Repository Admin** – a service to access a federation of repositories
- **Repository** – provides access to a set of resources
- **Resource** – a description of an artifact to be installed on a device
- **Capability** – a named set of properties
- **Requirement** – an assertion on a capability
- **Resolver** – an object to resolve resource dependencies and to deploy them
- **Repository file** – XML file containing resource meta-data

OBR2 High-Level View



OBR2 Web Site



Searches

Enter keywords to search for:

Felix

Search

1 Discovered Resources (0 - 1)

Sel	Presentation Name	Symbolic Name	Version	Dwn
□	HTTP Service	org.apache.felix.http.jetty	0.8.0...	
	An implementation of the OSGi HTTP Service using Jetty.			

Resolve

The Bundle Repository allows you to browse a federated collection of bundles. The user interface is search based. You can enter a **category**, keyword, or bundle symbolic name. If you just want to browse, feel free to enter the search button without arguments. This will list all bundles. Currently, most bundles do not properly specify a category and description. This will hopefully change over time. There is also a help page

If you want to contribute to the repository you can read the **provider** information. If you need to create your own repository, you can use the **utility**. The format of the repository files is available in **OSGi RFC 112 Bundle Repository**. The current repository is still in its infancy. Any help and **feedback** is highly appreciated.

OBR2 Repository File

```
<repository presentationname=“...” symbolicname=“...” ... >
   <resource>
      <description>...</description>
      <size>...</size>
      <documentation>...</documentation>
      <source>...</source>
      <category id=“...”/>
      <capability>...</capability>
      ...
      <requirement>...</requirement>
      ...
   </resource>
   ...
</repository>
```

[Peter Kriens created a tool, called bindex, to generate repository files.]

OBR2 Generic Capability Concept

- Resources can provide any number of capabilities
  - Simply a “typed” set of properties

```
<capability name='package'>
   <p n='package' v='org.foo.bar'/>
   <p n='version' t='version' v='1.0.0'/>
</capability>
```

OBR2 Generic Requirement Concept

- Resources can provide any number of requirements
  - Simply a “typed” LDAP query

```
<require name='package' extend='false'
   multiple='false' optional='false'
   filter='(&amp;(package=org.foo.bar)(version&gt;=1.0.0))'>
   Import package org.foo.bar
</require>
```

OBR2 Capability/Requirement Mappings

- Mappings provided for
  - Import/export package
  - Provide/require bundle
  - Host/fragment
  - Import/export service
  - Execution environment
  - Native code
- Custom mappings to arbitrary capabilities/ requirements

OBR2 Repository Admin Service

```
public interface RepositoryAdmin
{
    public Resource[] discoverResources(String filterExpr);
    public Resolver resolver();
    public Repository addRepository(URL repository)
        throws Exception;
    public boolean removeRepository(URL repository);
    public Repository[] listRepositories();
    public Resource getResource(String respositoryId);
}
```

OBR2 Resolver Object

```
public interface Resolver
{
    public void add(Resource resource);
    public Requirement[] getUnsatisfiedRequirements();
    public Resource[] getOptionalResources();
    public Requirement[] getReason(Resource resource);
    public Resource[] getResources(Requirement requirement);
    public Resource[] getRequiredResources();
    public Resource[] getAddedResources();
    public boolean resolve();
    public void deploy(boolean start);
}
```

OBR2 Usage Scenario

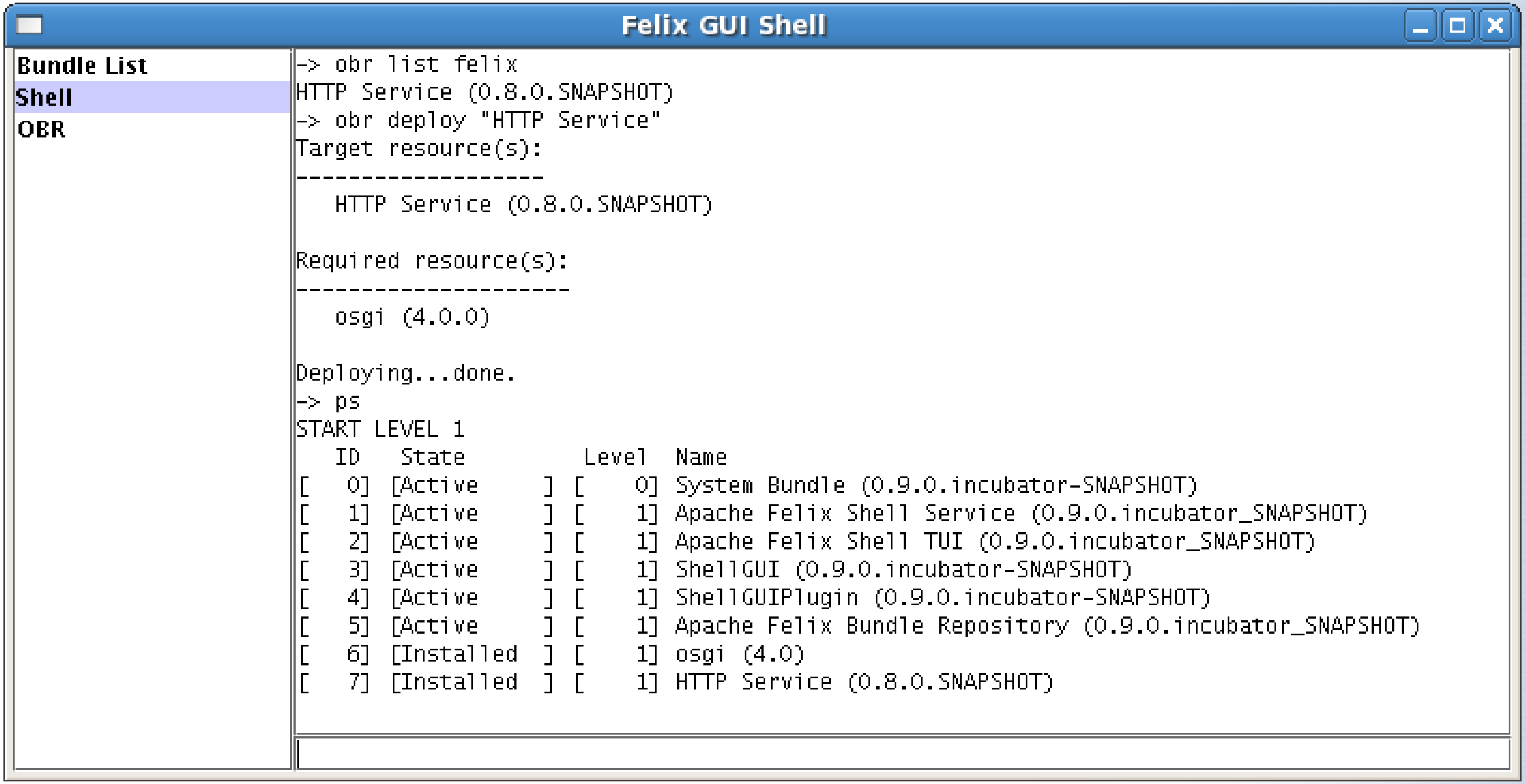
```
RepositoryAdmin repoAdmin = ... // Get repo admin service
Resolver resolver = repoAdmin.resolver();
Resource resource = repoAdmin.discoverResources(filterStr);
resolver.add(resource);
if (resolver.resolve()) {
    resolver.deploy();
} else {
    Requirement[] reqs = resolver.getUnsatisfiedRequirements();
    for (int i = 0; i < reqs.length; i++) {
        System.out.println("Unable to resolve: " + reqs[i]);
    }
}
```

Apache Felix OBR Implementation

Apache Felix OBR Implementation

- Bundle Repository sub-project
  - Not 100% complete with respect to the RFC
- Resolves bundle requirements taking into account locally installed bundles
- Resolver and deployment algorithms try to minimize number of installed bundles

Apache Felix OBR Shell Command

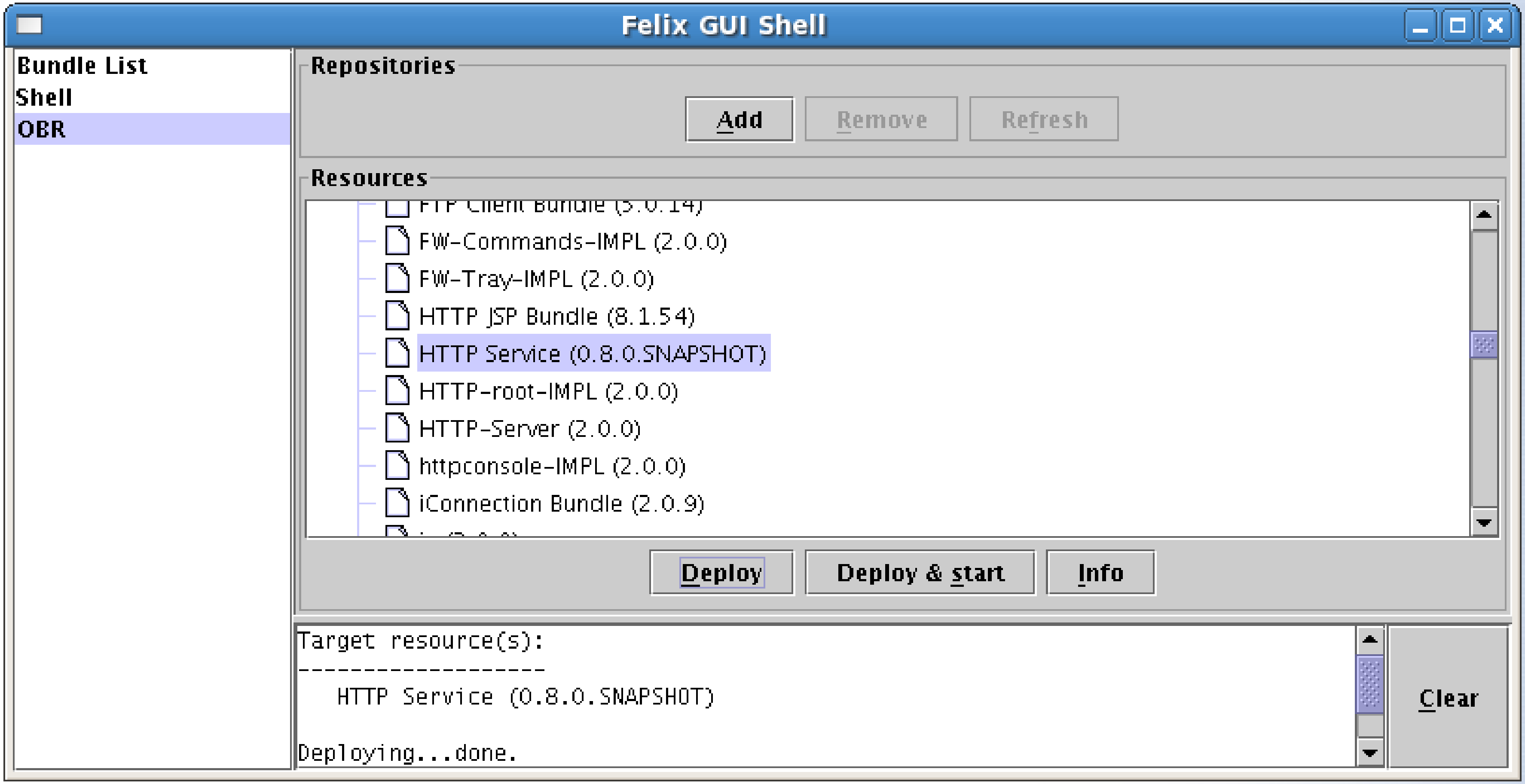


```
-> obr list felix
HTTP Service (0.8.0.SNAPSHOT)
-> obr deploy "HTTP Service"
Target resource(s):
-------------------
   HTTP Service (0.8.0.SNAPSHOT)

Required resource(s):
---------------------
   osgi (4.0.0)

Deploying...done.
-> ps
START LEVEL 1
   ID   State         Level  Name
[   0] [Active     ] [    0] System Bundle (0.9.0.incubator-SNAPSHOT)
[   1] [Active     ] [    1] Apache Felix Shell Service (0.9.0.incubator_SNAPSHOT)
[   2] [Active     ] [    1] Apache Felix Shell TUI (0.9.0.incubator_SNAPSHOT)
[   3] [Active     ] [    1] ShellGUI (0.9.0.incubator-SNAPSHOT)
[   4] [Active     ] [    1] ShellGUIPlugin (0.9.0.incubator-SNAPSHOT)
[   5] [Active     ] [    1] Apache Felix Bundle Repository (0.9.0.incubator_SNAPSHOT)
[   6] [Installed  ] [    1] osgi (4.0)
[   7] [Installed  ] [    1] HTTP Service (0.8.0.SNAPSHOT)
```

Apache Felix OBR GUI



FTP Client Bundle (3.0.14)
FW-Commands-IMPL (2.0.0)
FW-Tray-IMPL (2.0.0)
HTTP JSP Bundle (8.1.54)
HTTP Service (0.8.0.SNAPSHOT)
HTTP-root-IMPL (2.0.0)
HTTP-Server (2.0.0)
httpconsole-IMPL (2.0.0)
iConnection Bundle (2.0.9)

```
Target resource(s):
-------------------
   HTTP Service (0.8.0.SNAPSHOT)

Deploying...done.
```

Apache Felix Maven Bundle Plugin

- Main purpose is to simplify bundle development
  - Uses BND
    - Bundle packaging
    - Automates bundle meta-data generation
- Prototyping OBR support
  - Specify that resulting bundle JAR files are added/updated in a repository XML file
    - Uses bindex

Apache Felix Framework

- Deployment vs. runtime resolver
  - Each resolver does the same work, but the result of the former is deployment and the latter is a set of wires
- Felix framework adopted the generic OBR model
  - Goal is to make one resolver that is used in both the framework and the OBR implementation
  - Could lead to exposing generic dependencies in bundles

OBR Bundle Repositories

OBR Bundle Repositories

- Apache Felix Project
  - In the works, will include Felix sub-project bundles
- Apache Felix Commons
  - In the works, currently available from Maven
  - Bundled versions of common open source libraries
- Equinox Orbit (http://www.eclipse.org/orbit)
- Knopflerfish (http://www.knopflerfish.org/repo/)
- ProSyst (http://dz.prosyst.com/pdoc/repository.xml)

Issues

Issues

- Deployment vs. runtime requirements
  - Potentially need some tweaks to use as a framework resolver
- Uses constraints
  - Related to above point, are not currently addressed
- Local resources
  - Not cleanly integrated
- Bundle “applications”
  - There is a need for a higher level view, but probably on top of OBR

Conclusion

Conclusion

- OSGi technology is a success...
  - ...now we have to deal with it
- To keep momentum going we must make it easier for developers
  - To find existing bundles
  - To use existing bundles
  - To share their own bundles
- OBR is addressing these needs