Introduction to OSGi

Marcel Offermans

luminis

Introduction

- Marcel Offermans
  - marcel.offermans@luminis.nl
- Luminis
  - IT solutions from idea to implementation
  - with and for customers: consultancy and projects
  - experts in Java, OSGi, .NET, Agile/Scrum
- Apache Felix and Ace PMC

Agenda

- History of OSGi
- The Framework
- The Compendium
- Patterns, Models & Embedding
- Open Source Frameworks

OSGi history

- Started as an embedded platform for the “home gateway”
- Originally under the JCP as JSR-8 (1999)
- OSGi alliance, consists of a large number of big companies, with the following mission:
  - Maintaining and publicizing the OSGi specification.
  - Certifying implementations.
  - Organising events.
- Current version: OSGi Release 4.1 (JSR-291)

luminis

OSGi releases

- R1: may 2000
- R2: october 2001
  - Java Embedded Server (Sun), Oscar (SourceForge)
- R3: march 2003
  - Knopflerfish
- R4: august 2005
  - IBM joined and influenced this release, Equinox (Eclipse Foundation)
- R4.1: april 2007

luminis

OSGi today

OSGi technology is the dynamic module system for Java™

OSGi technology is Universal Middleware.

OSGi technology provides a service-oriented, component-based environment for developers and offers standardized ways to manage the software lifecycle. These capabilities greatly increase the value of a wide range of computers and devices that use the Java™ platform.

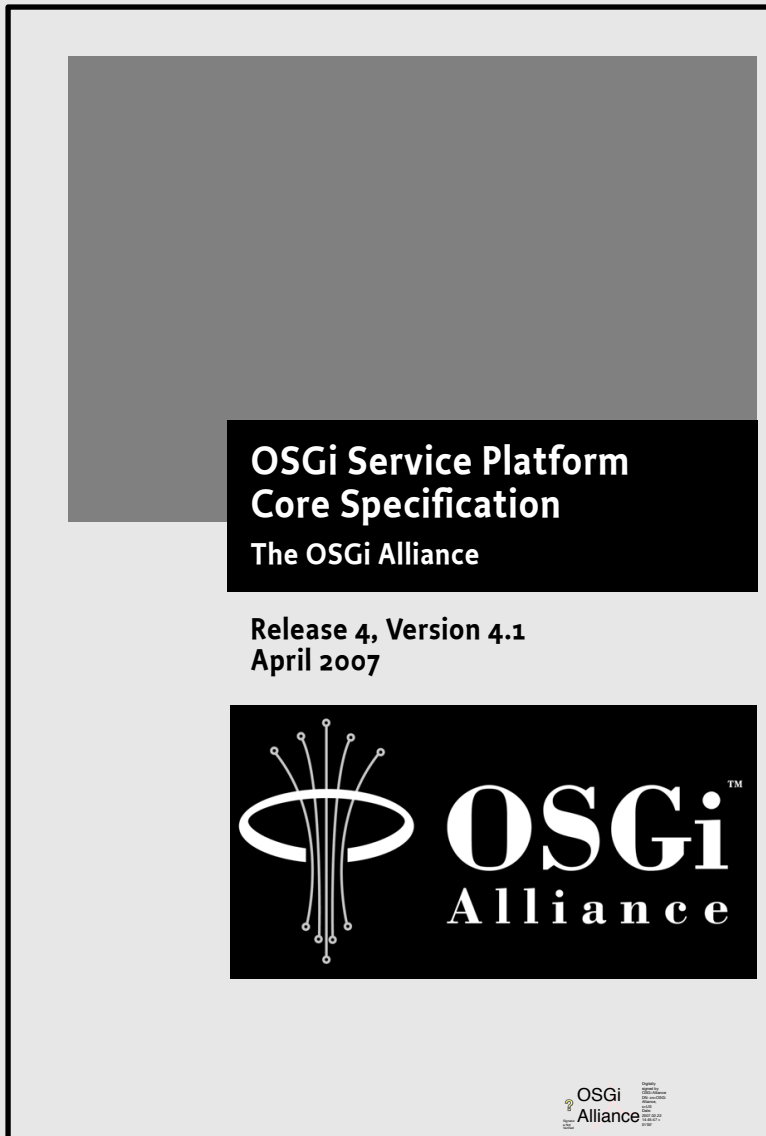
luminis

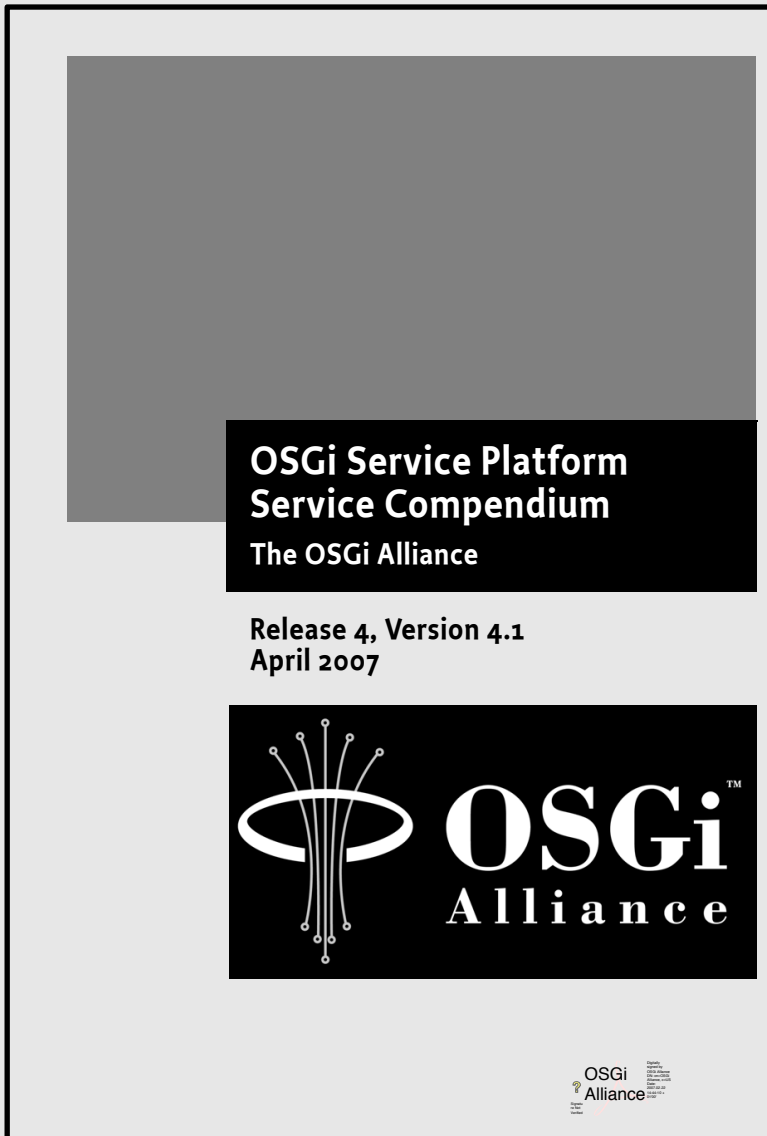
OSGi Alliance

- Expert Groups:
  - core platform (CPEG)
  - mobile (MEG)
  - vehicle (VEG)
  - enterprise (EEG)
  - residential (REG)
- Working Groups:
  - marketing
  - requirements

luminis

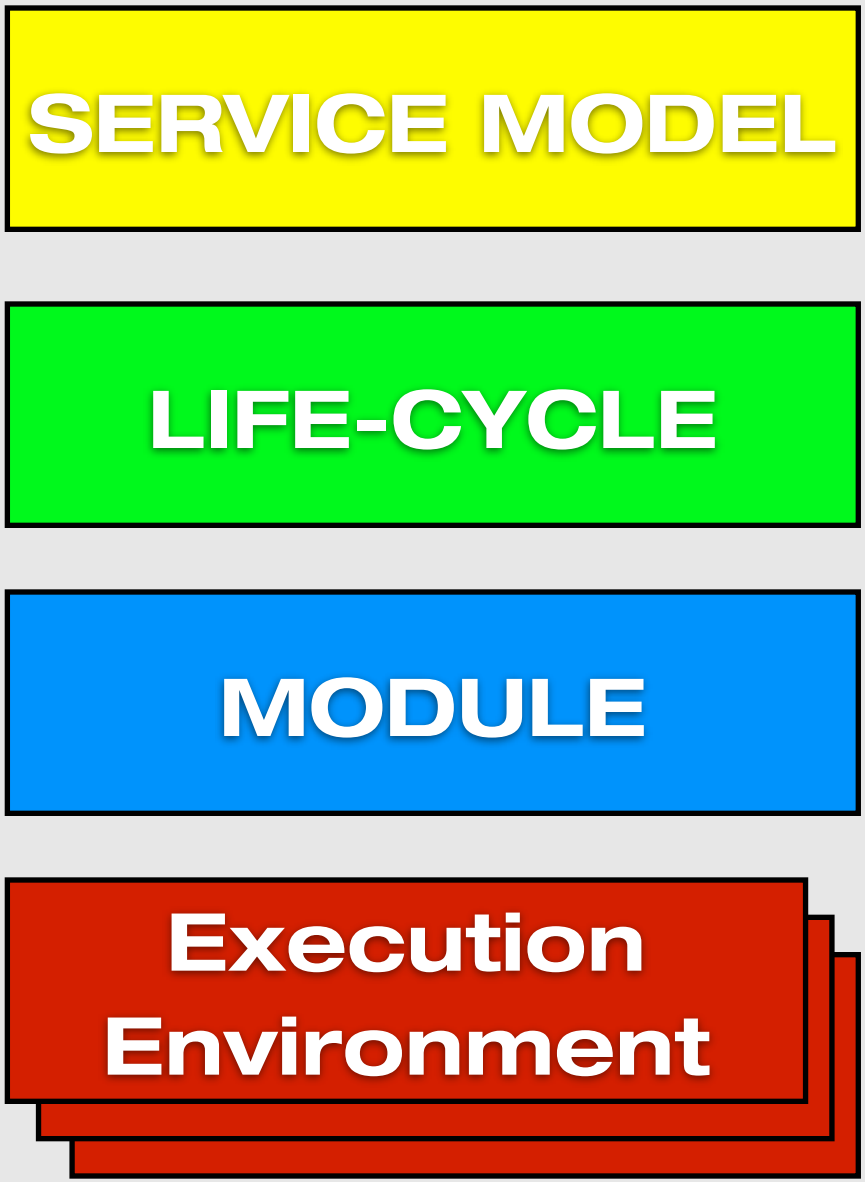
OSGi specification





luminis

OSGi Framework Layering



L3 - Provides a publish/find/bind service model to decouple bundles

L2 - Manages the life cycle of a bundle in a framework without requiring the vm to be restarted

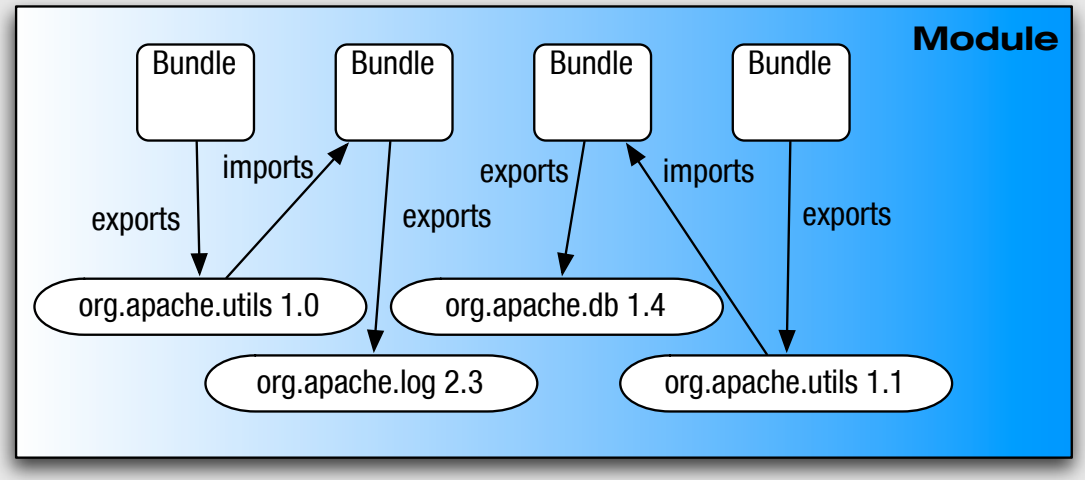
L1 - Creates the concept of a module (aka. bundles) that use classes from each other in a controlled way according to system and bundle constraints

L0 - well defined profiles that define the environment in which bundles can work, ie:

* CDC/Foundation
* JavaSE-6

luminis

Module Layer (1/3)



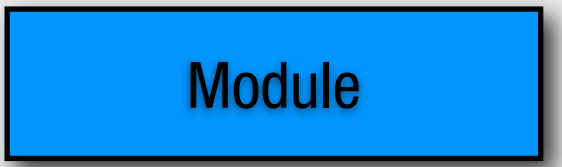
- Unit of deployment is the bundle i.e., a JAR
- Separate class loader per bundle
  - Class loader graph
  - Independent namespaces
  - Class sharing at the Java package level

Module

luminis

Module Layer (2/3)

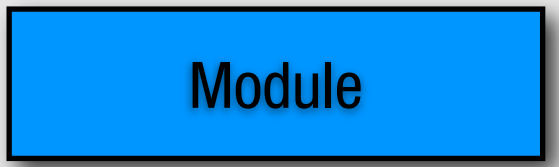
- Multi-version support
  - i.e., side-by-side versions
- Explicit code boundaries and dependencies
  - i.e., package imports and exports
- Support for various sharing policies
  - i.e., arbitrary version range support
- Arbitrary export/import attributes
  - Influence package selection



luminis

Module Layer (3/3)

- Sophisticated class space consistency model
  - Ensures code constraints are not violated
- Package filtering for fine-grained class visibility
  - Exporters may include/exclude specific classes from exported package
- Bundle fragments
  - A single logical module in multiple physical bundles
- Bundle dependencies
  - Allows for tight coupling when required

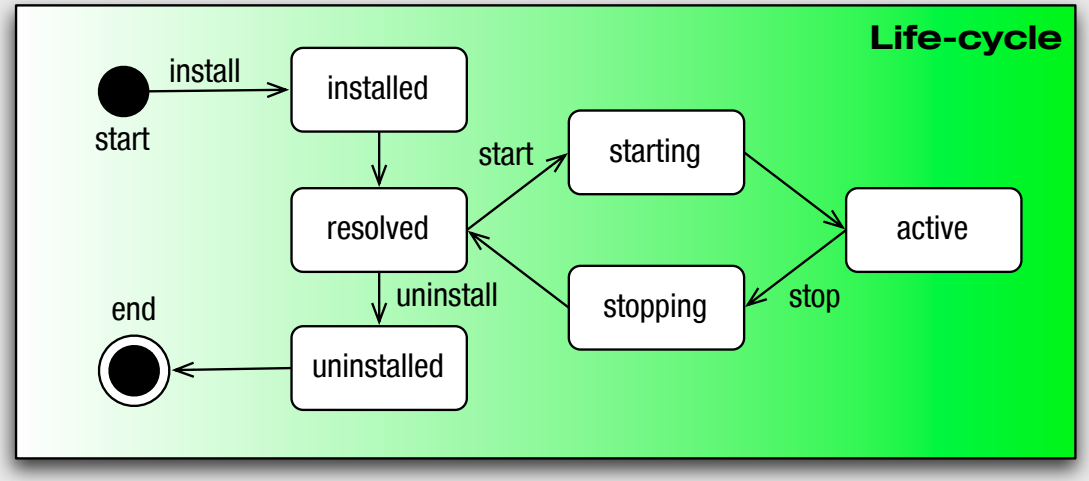


luminis

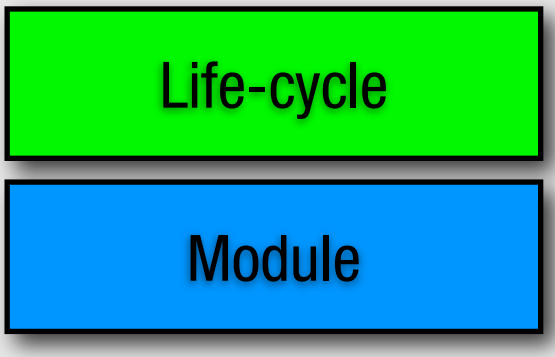
Manifest

```
Bundle-Name: Example Bundle
Bundle-SymbolicName: net.luminis.example.bundle
Bundle-Version: 1.0.0
DynamicImport-Package:
    net.luminis.jdbc.*
Import-Package:
    org.osgi.framework;version=“1.3“,
    org.osgi.service.event;version=“[1.1,2.0)“,
    net.luminis.foo;resolution:=“optional“
Export-Package:
    org.osgi.service.event;uses:=org.osgi.framework;version=“1.1“
Bundle-ManifestVersion: 2
```

Life-cycle Layer

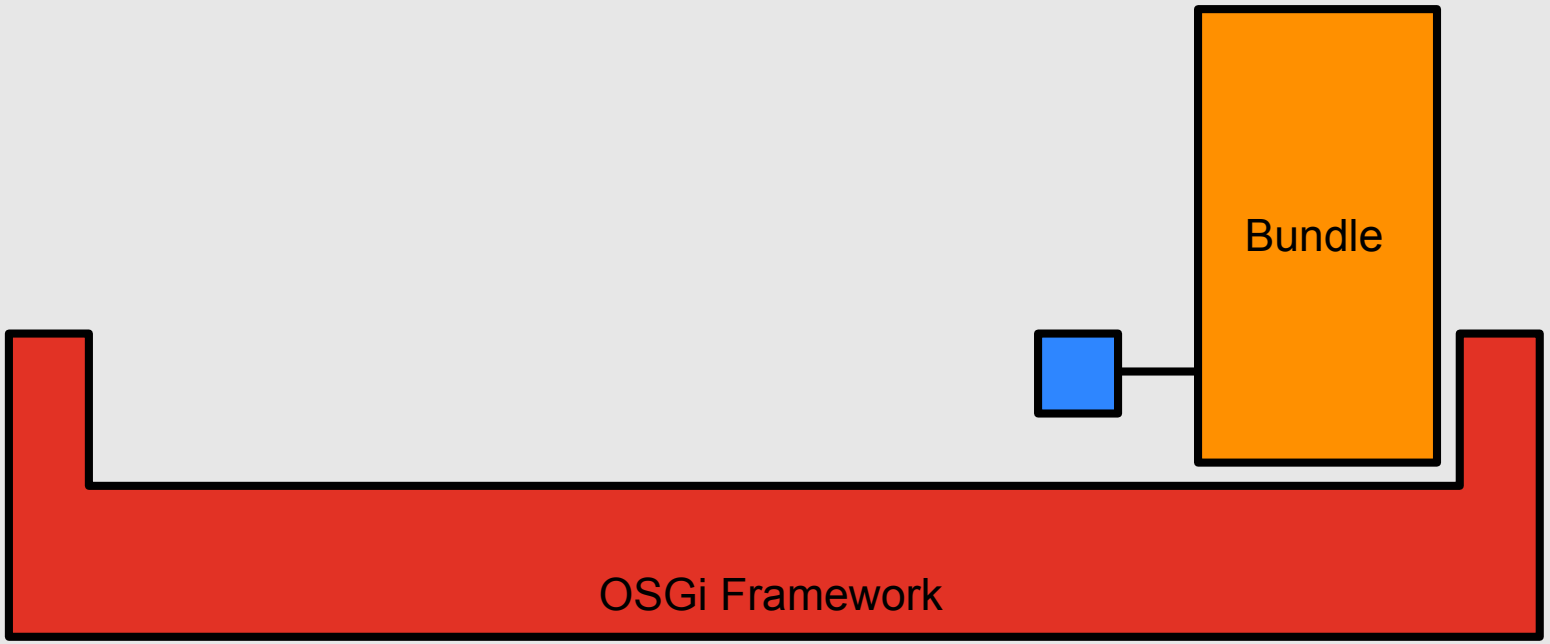


- Managed life cycle
  - States for each bundle;
- Allows updates of existing bundles
  - Dynamically install, start, update, and uninstall



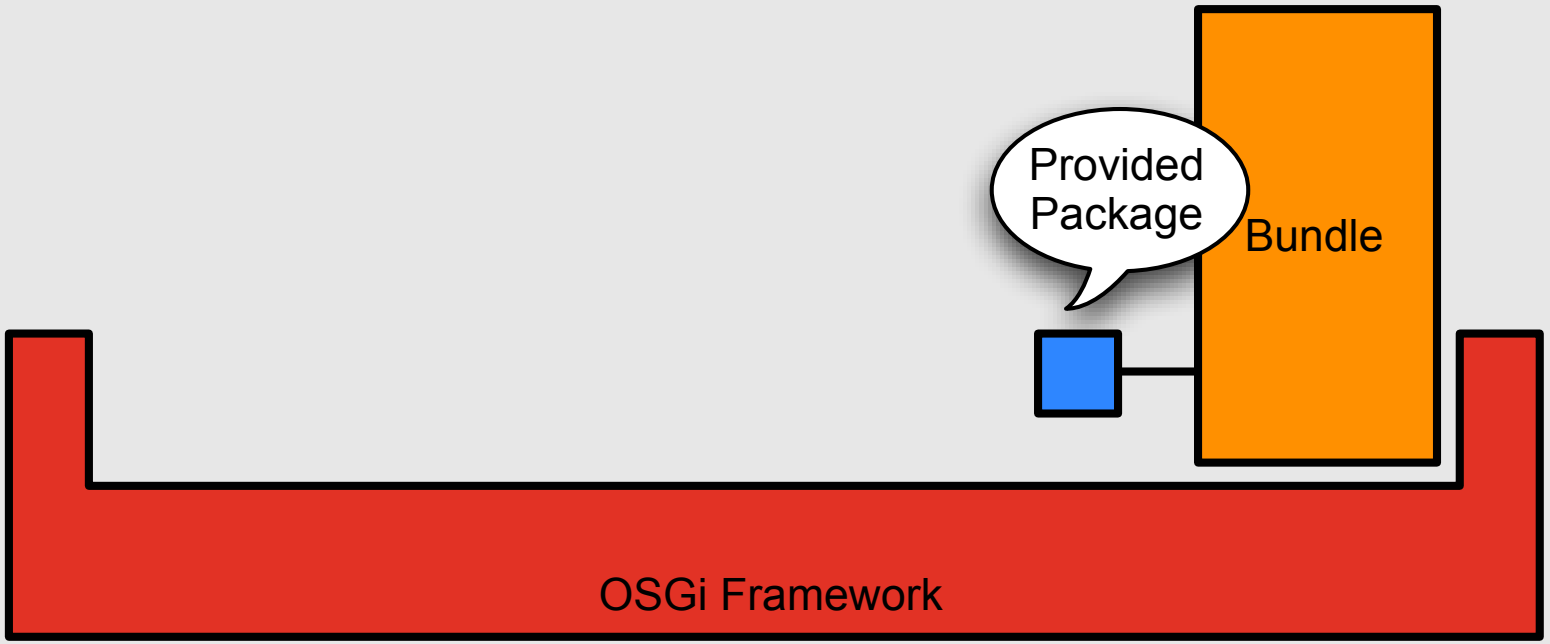
luminis

Life-cycle Example



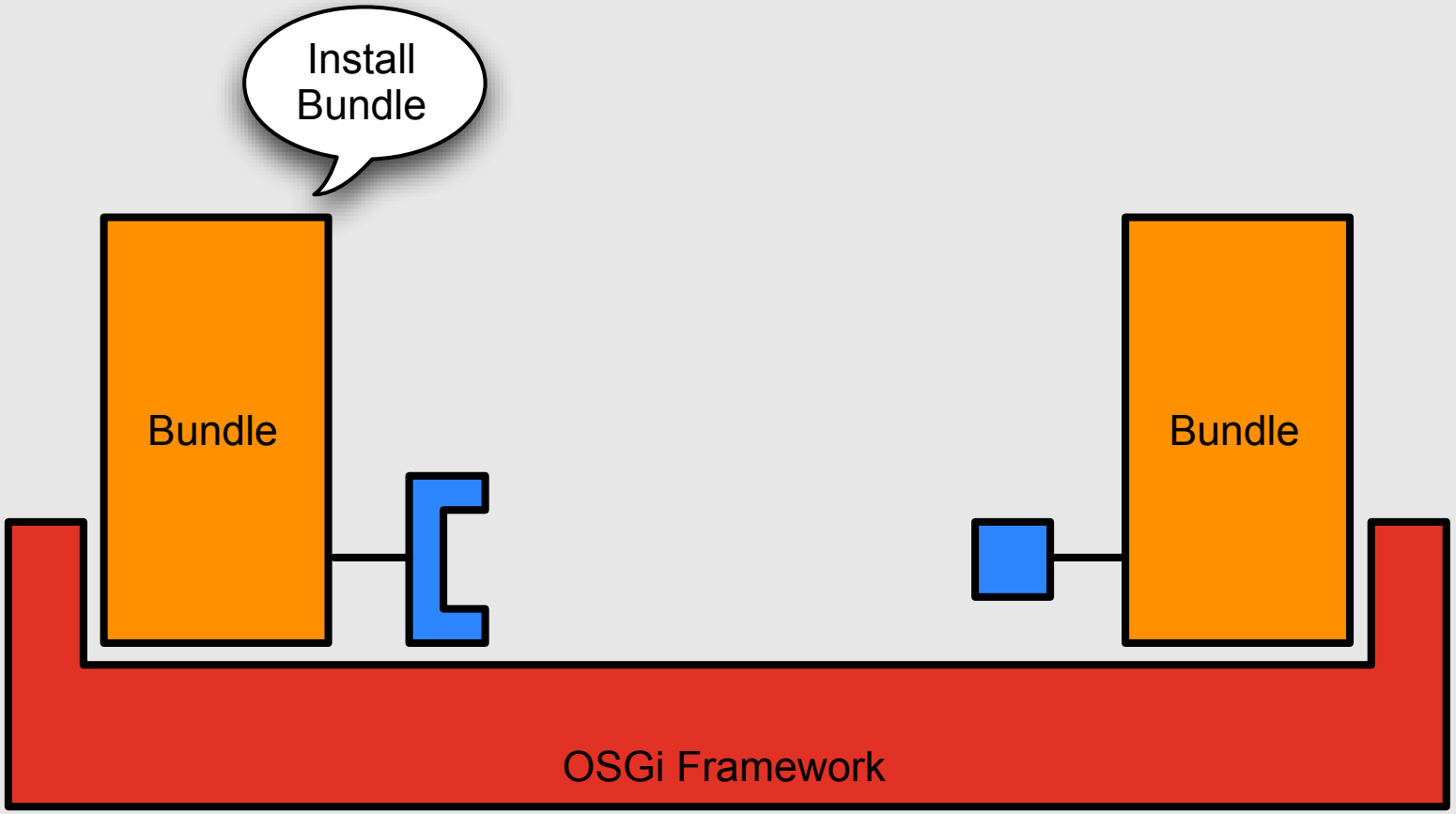
luminis

Life-cycle Example



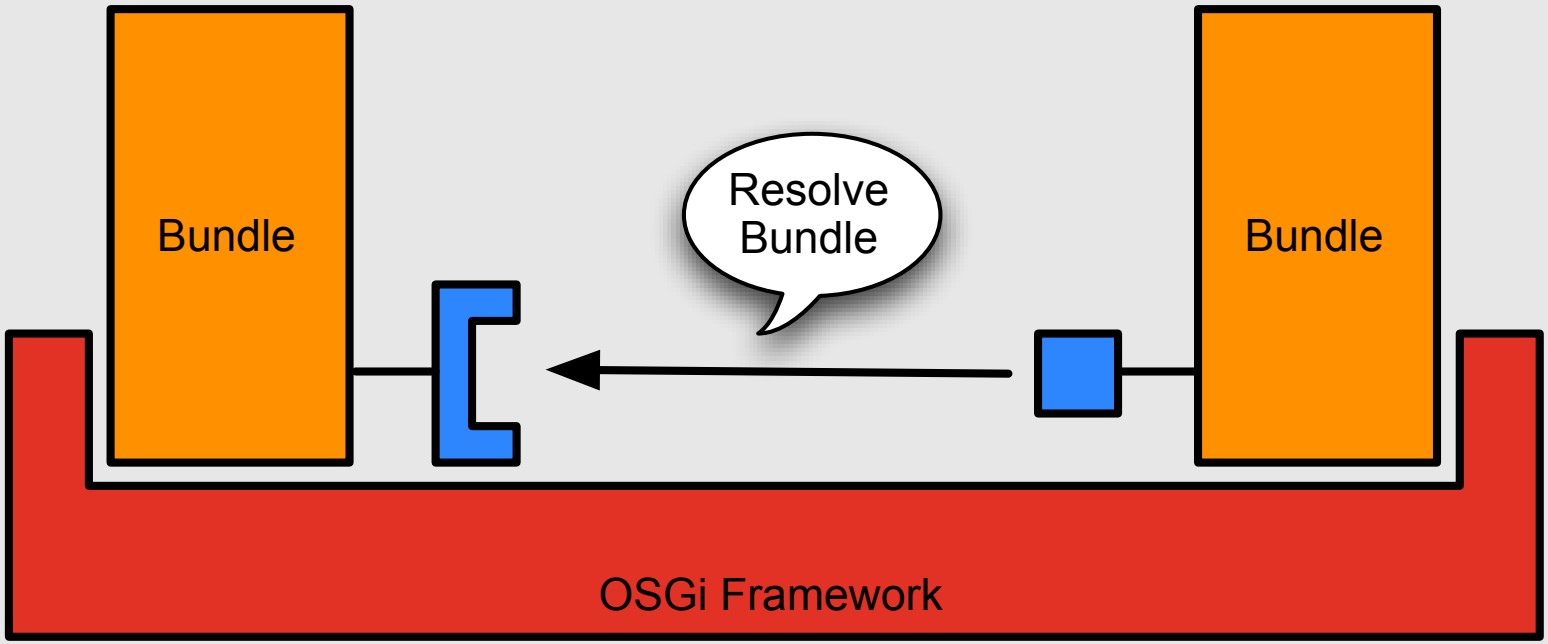
luminis

Life-cycle Example



luminis

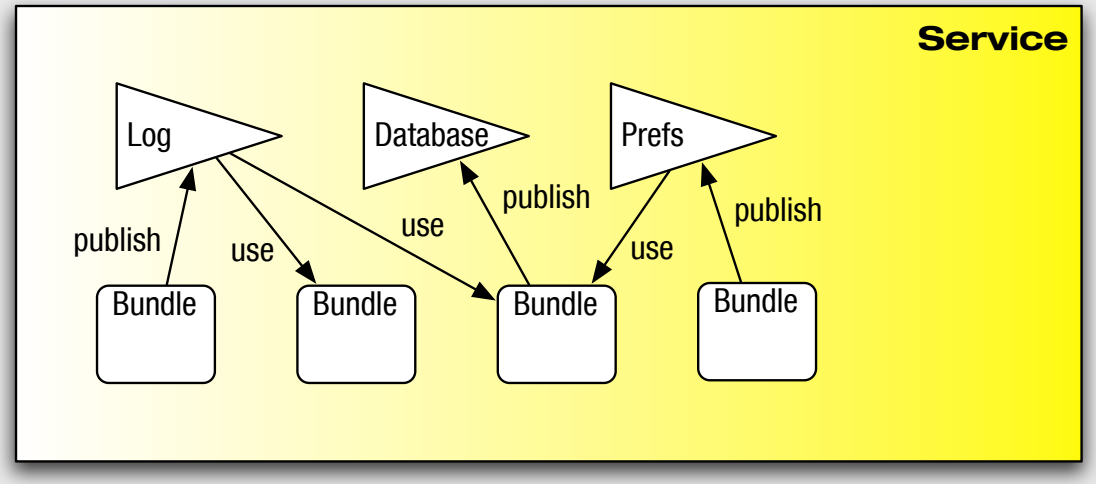
Life-cycle Example

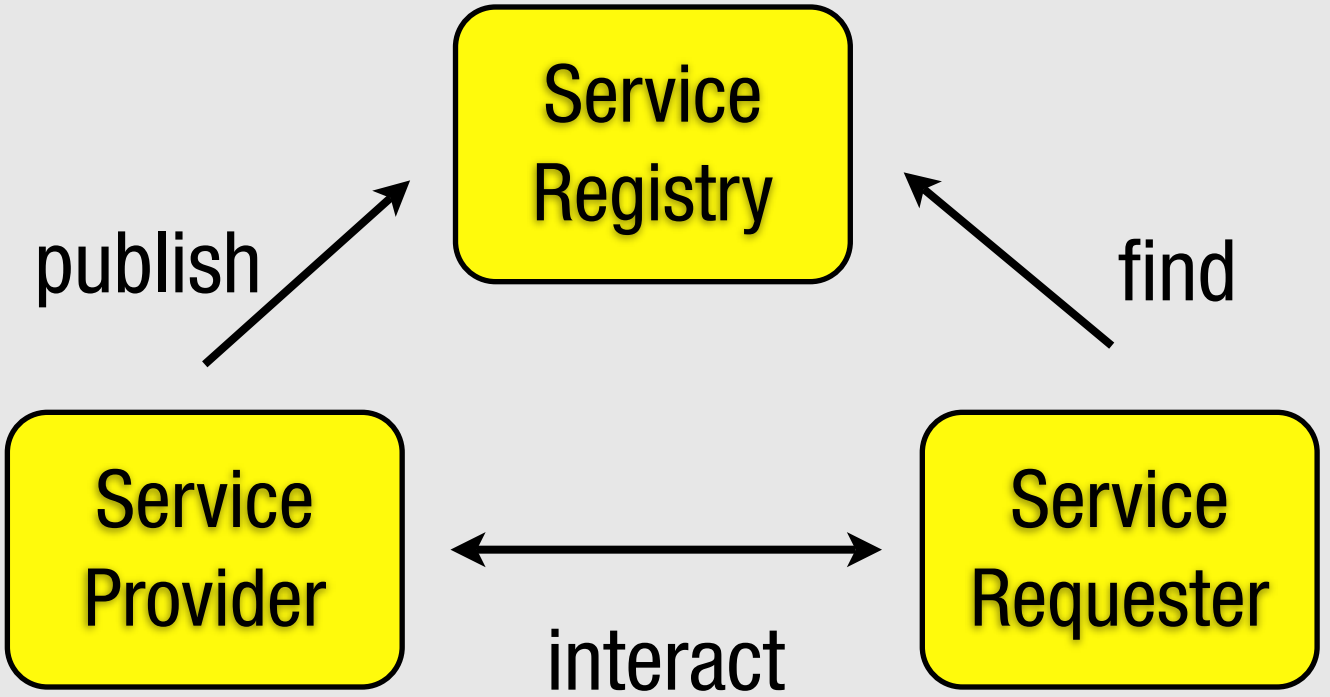


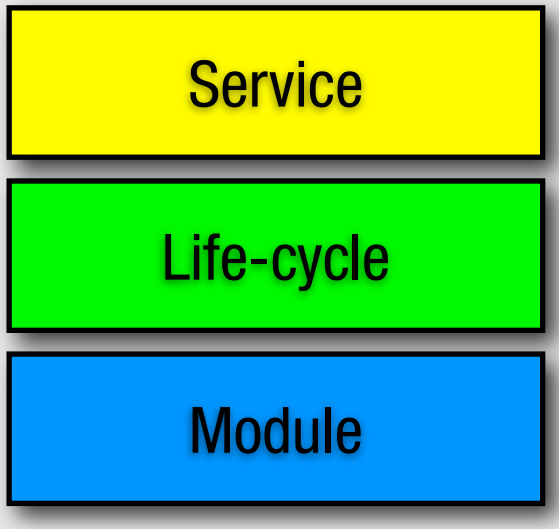
luminis

Service Layer

- OSGi framework promotes service oriented interaction pattern among bundles

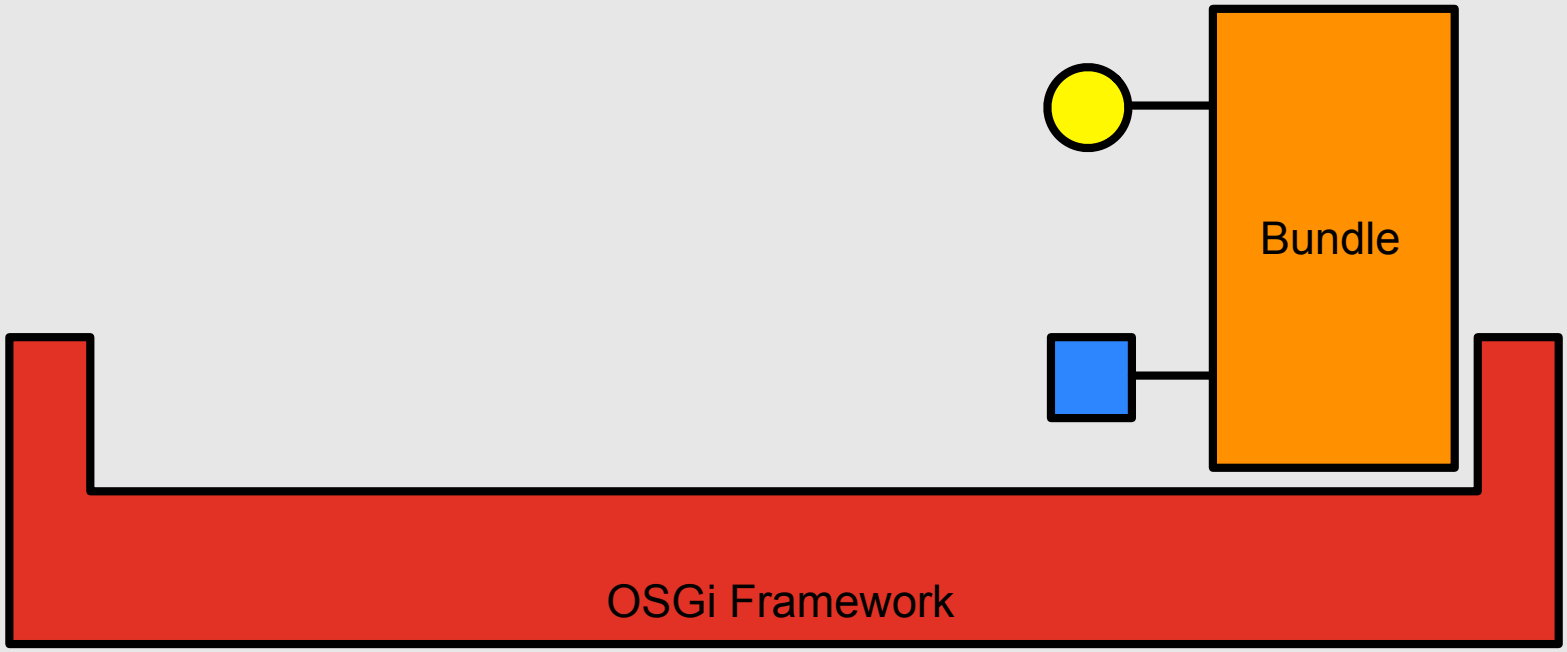






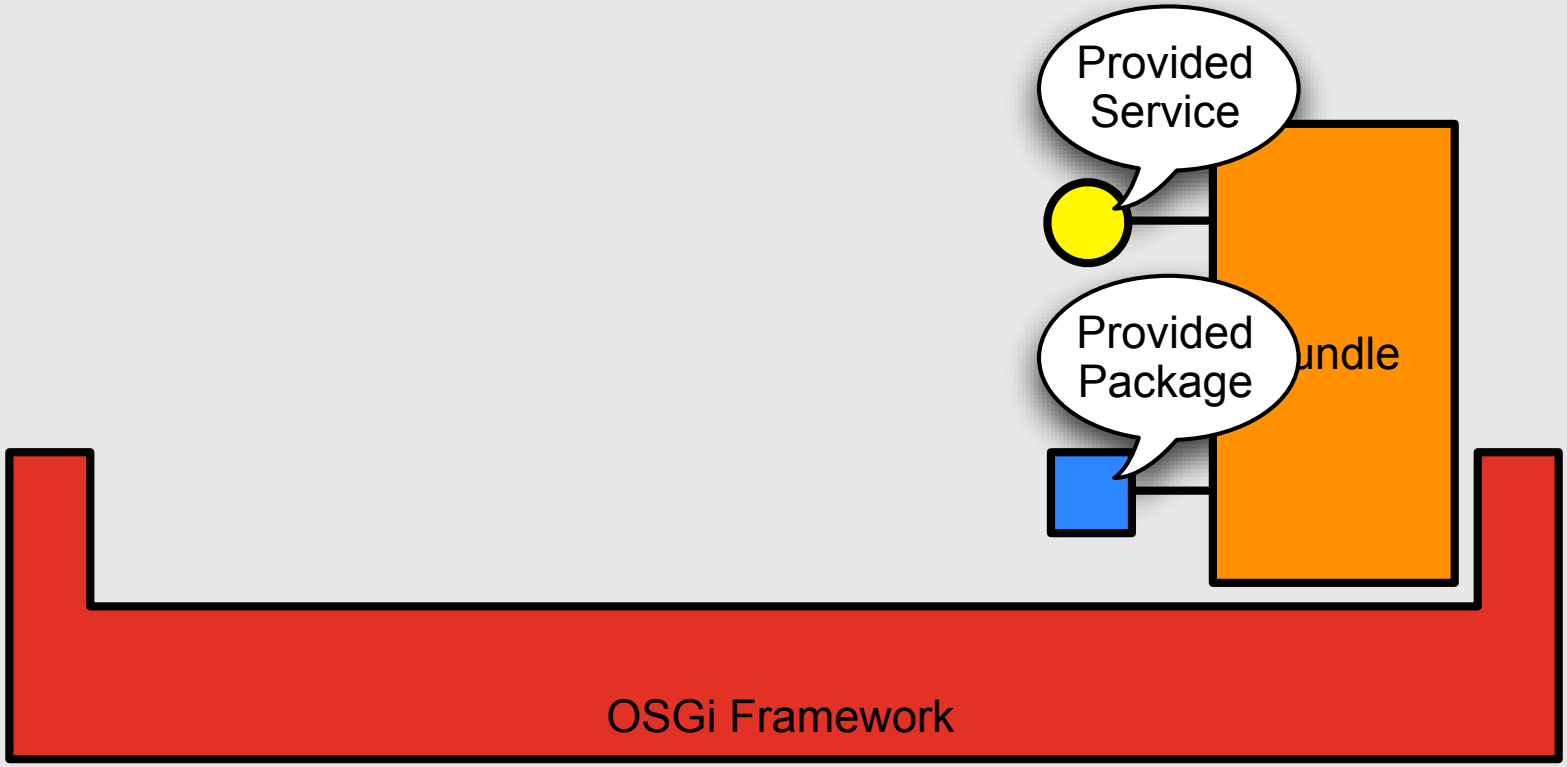
luminis

Service Example



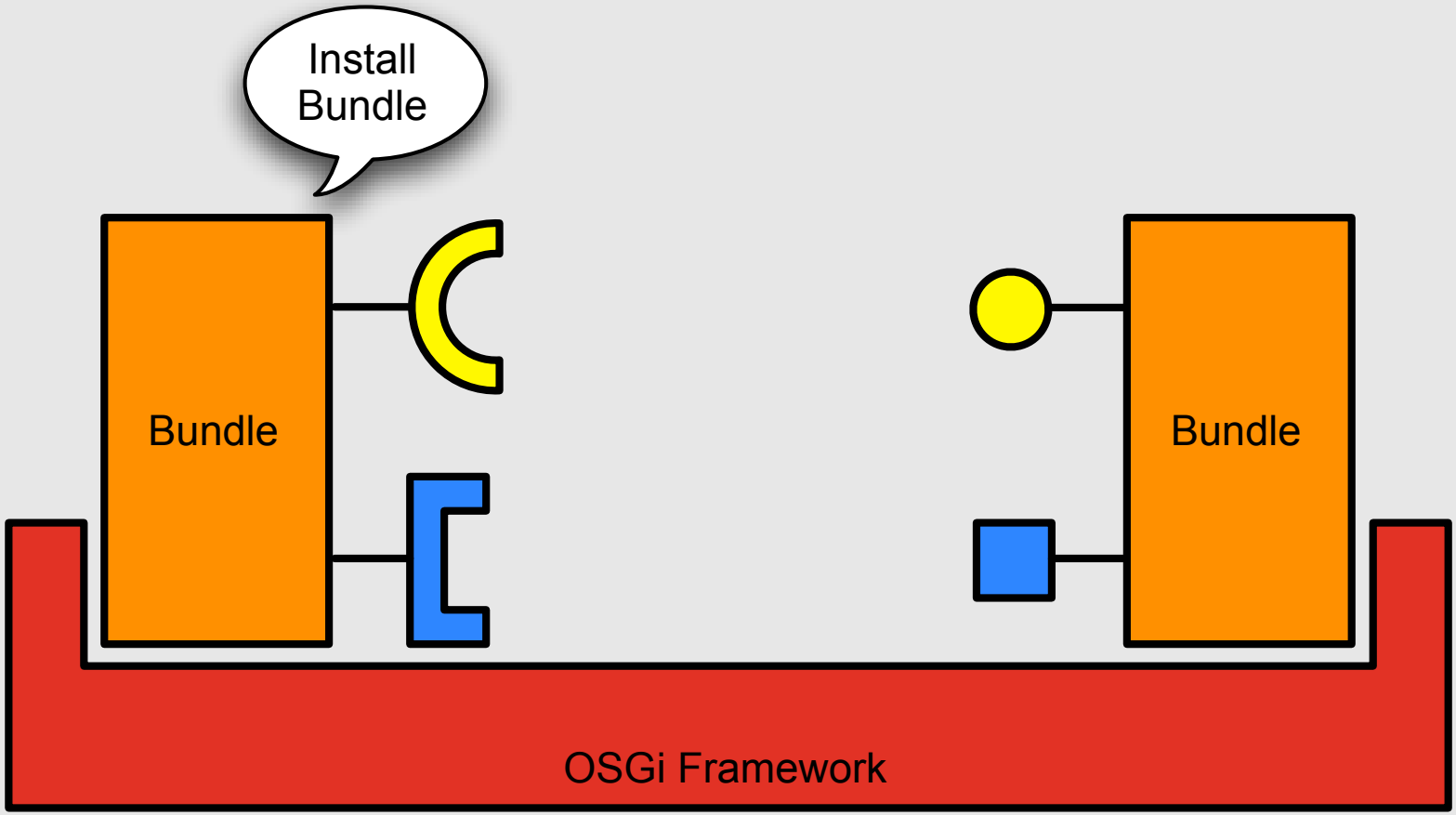
luminis

Service Example



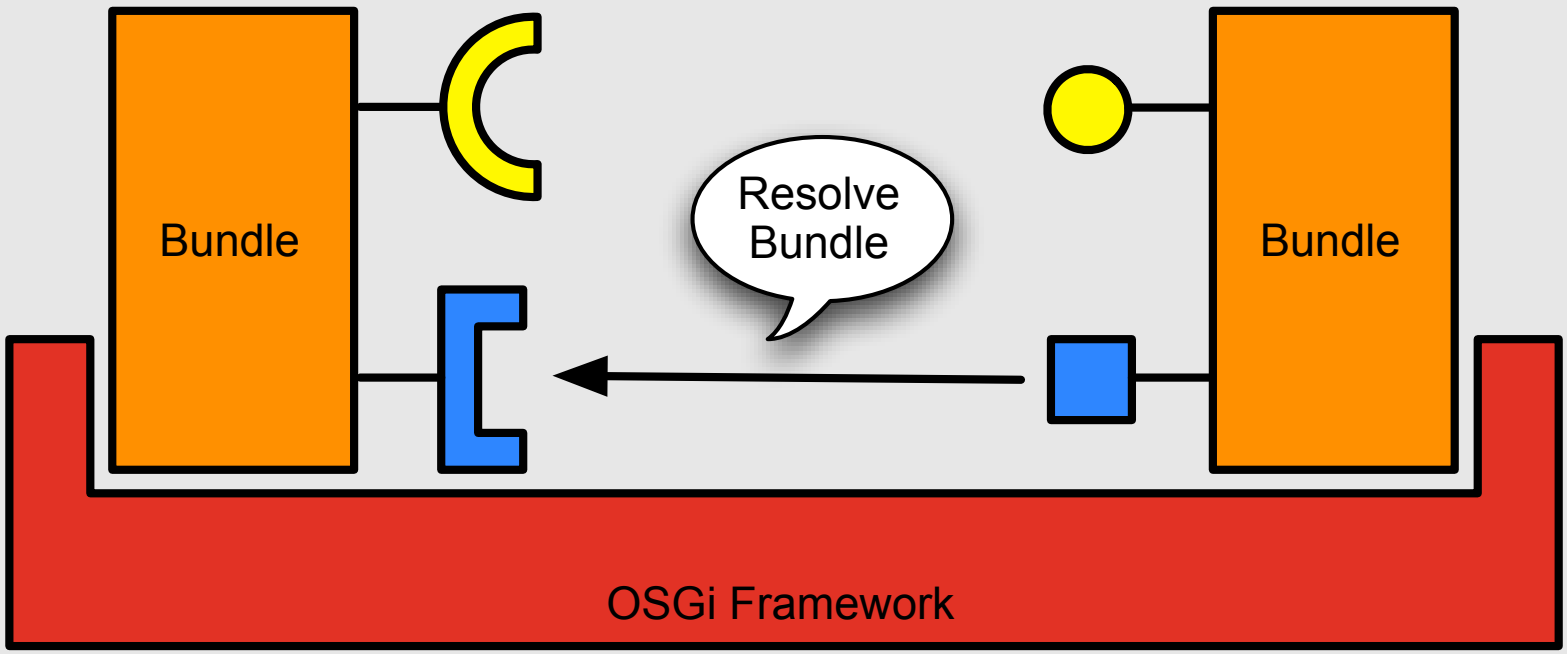
luminis

Service Example



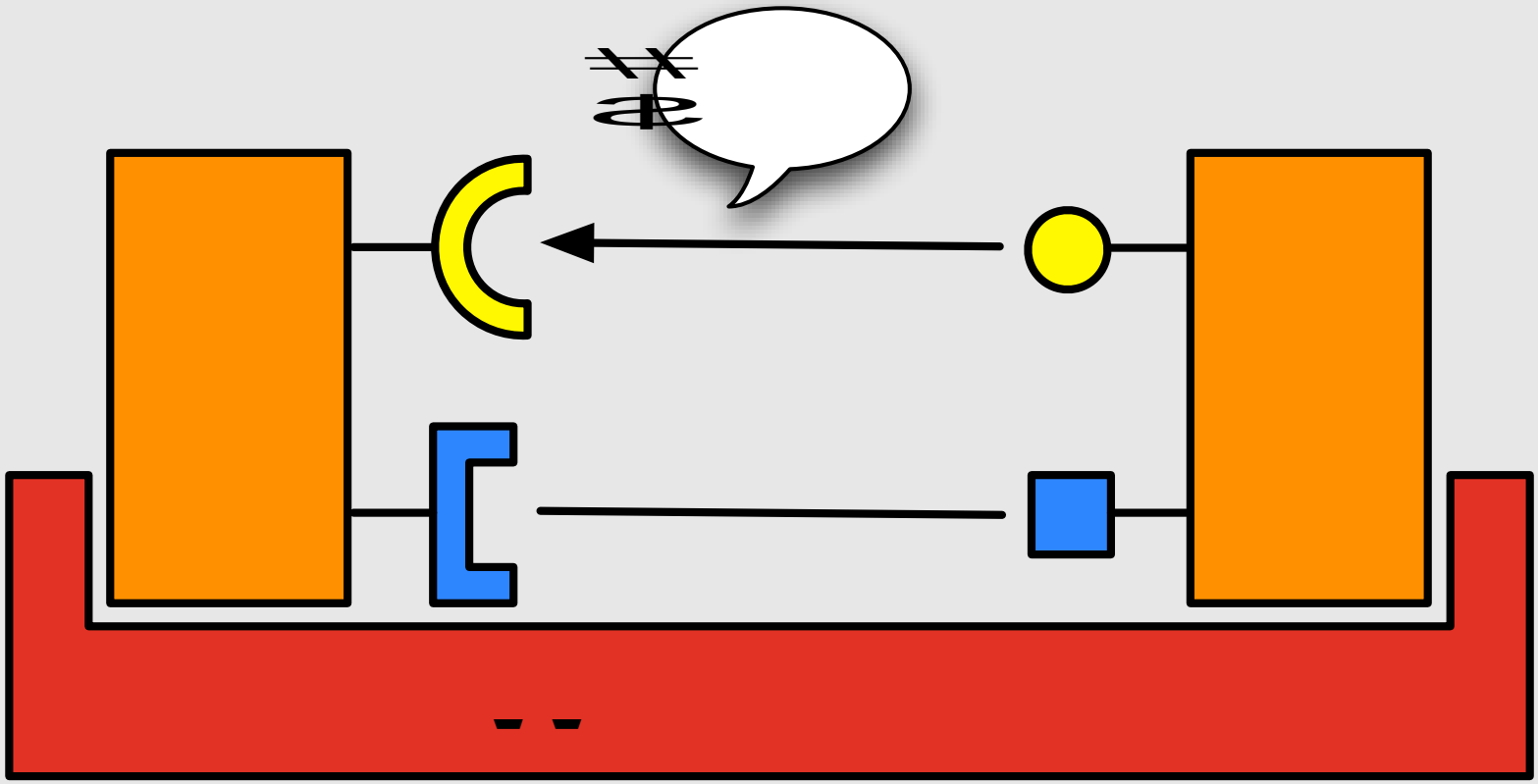
luminis

Service Example



luminis

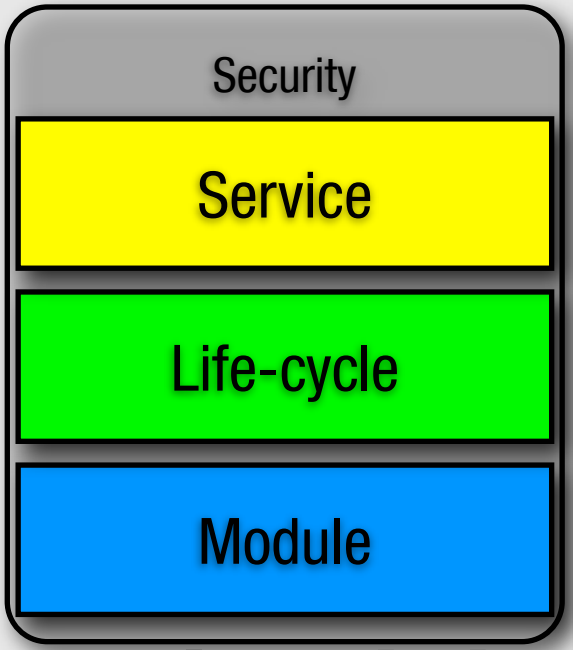
Service Example



luminis

Security Concepts Overview

- Codebased security of the Java Security Model
  - Makes use of Protection Domain
  - The stack walk based Permission Check
  - Signed bundles
- User based security is supported by the UserAdmin service but not integrated in the standard permission check as with JAAS
- PermissionAdmin and ConditionalPermissionAdmin services provide management infrastructure



luminis

Conditional Permission Admin

- New (4.0) way of doing permission management
  - use this exclusively for new implementations
  - interoperability when both PA and CPA are present
- IF all conditions of a set of conditions match THEN apply the supplied permissions
  - More flexible, extensible model
- Conditions evaluation is highly optimized

luminis

Example

```
ConditionalPermissionAdmin condPermAdmin = getConditionalPermissionAdmin();

condPermAdmin.addConditionalPermissionInfo(
  new ConditionInfo[] {
    new ConditionInfo(
      BundleLocationCondition.class.getName(),
      new String[] {"*://www.luminis.nl/*"})
    },
  new PermissionInfo[] {
    new PermissionInfo(AdminPermission.class.getName(),
                       "*", "*"),
    new PermissionInfo(ServicePermission.class.getName(),
                       "*", ServicePermission.GET),
    new PermissionInfo(ServicePermission.class.getName(),
                       "net.luminis.service.*", ServicePermission.REGISTER),
    new PermissionInfo(PackagePermission.class.getName(),
                       "net.luminis.service.*", PackagePermission.EXPORT),
    new PermissionInfo(PropertyPermission.class.getName(),
                       "*", "read")
  });
```

luminis

Signing bundles in Eclipse

luminis

BundleSignerCondition

- Condition to test if the signer of a bundle matches a pattern
- Uses the wildcard matching

```
condPermAdmin.addConditionalPermissionInfo(new ConditionInfo[]{
   new ConditionInfo(BundleSignerCondition.class.getName(),
     new String[]{"*,o=luminis"})
   }, ALLPERMISSION_INFO);
```

Local Permissions

- Defined in a resource inside the bundle
- Defines a set of permissions that are enforced by the framework
- A bundle can get less than these permissions, but never more
- Defaults to All Permissions
- Good way for operators to “audit” the permissions of a bundle

luminis

Local Permissions Example

- OSGI-INF/permissions.perm

```
( ..ServicePermission "..log.LogService" "GET" )
( ..PackagePermission "..log" "IMPORT" )
( ..ServicePermission "..cm.ManagedService" "REGISTER" )
( ..PackagePermission "..cm" "IMPORT" )
( ..ServicePermission "..useradmin.UserAdmin" "GET" )
( ..PackagePermission "..cm" "SET" )
( ..PackagePermission "com.acme.chess" "IMPORT,EXPORT" )
( ..PackagePermission "com.acme.score" "IMPORT" )
```

luminis

Agenda

- History of OSGi
- The Framework
- The Compendium
- Patterns, Models & Embedding
- Open Source Frameworks

OSGi compendium

User Admin

Initial Provisioning

Wire Admin

Log

Device Access

XML Parser

Measurement and State

Preferences

UPnP™ Device

Configuration Admin

Position

Metatype

Service Tracker

Event Admin

HTTP

IO Connector

Execution Environment Spec

Declarative Services

luminis

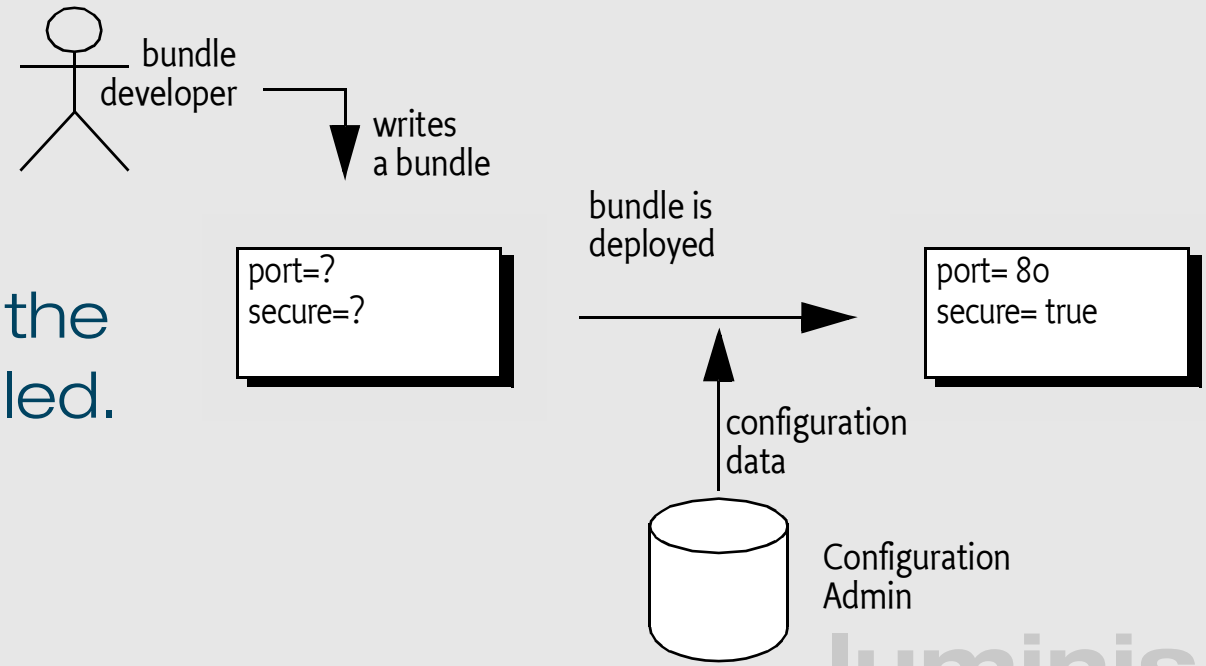
User Admin

- Used in any application that needs role based access control
- Provides: users, roles and groups
- Can authenticate users
- Can determine autorization for authenticated users
- Fairly easy to plug-in to HTTP, SOAP, RMI, JMX or anything else

luminis

Config Admin

- Configuration Admin:
  - contains externally configurable settings for a service;
  - allows management systems to configure all settings;
  - settings can be created even before the actual bundle is installed.



luminis

Event Admin

Channel Pattern

- Publish subscribe
- Asynchronous and synchronous
- Hierarchical topics
- Used within OSGi too

luminis

Event Admin Example

```
class Subscriber implements BundleActivator, EventHandler {
    final static String[] topics = new String[] {
        "org/osgi/service/log/LogEntry/LOG_WARNING",
        "org/osgi/service/log/LogEntry/LOG_ERROR" };

    public void start(BundleContext context) {
        Dictionary d = new Hashtable();
        d.put(EventConstants.EVENT_TOPIC, topics);
        d.put(EventConstants.EVENT_FILTER, "(bundle.symbolicName=com.acme.*)");
        context.registerService(EventHandler.class.getName(), this, d);
    }
    public void stop(BundleContext context) {
    }

    public void handleEvent(Event event) {
        //...
    }
}
```

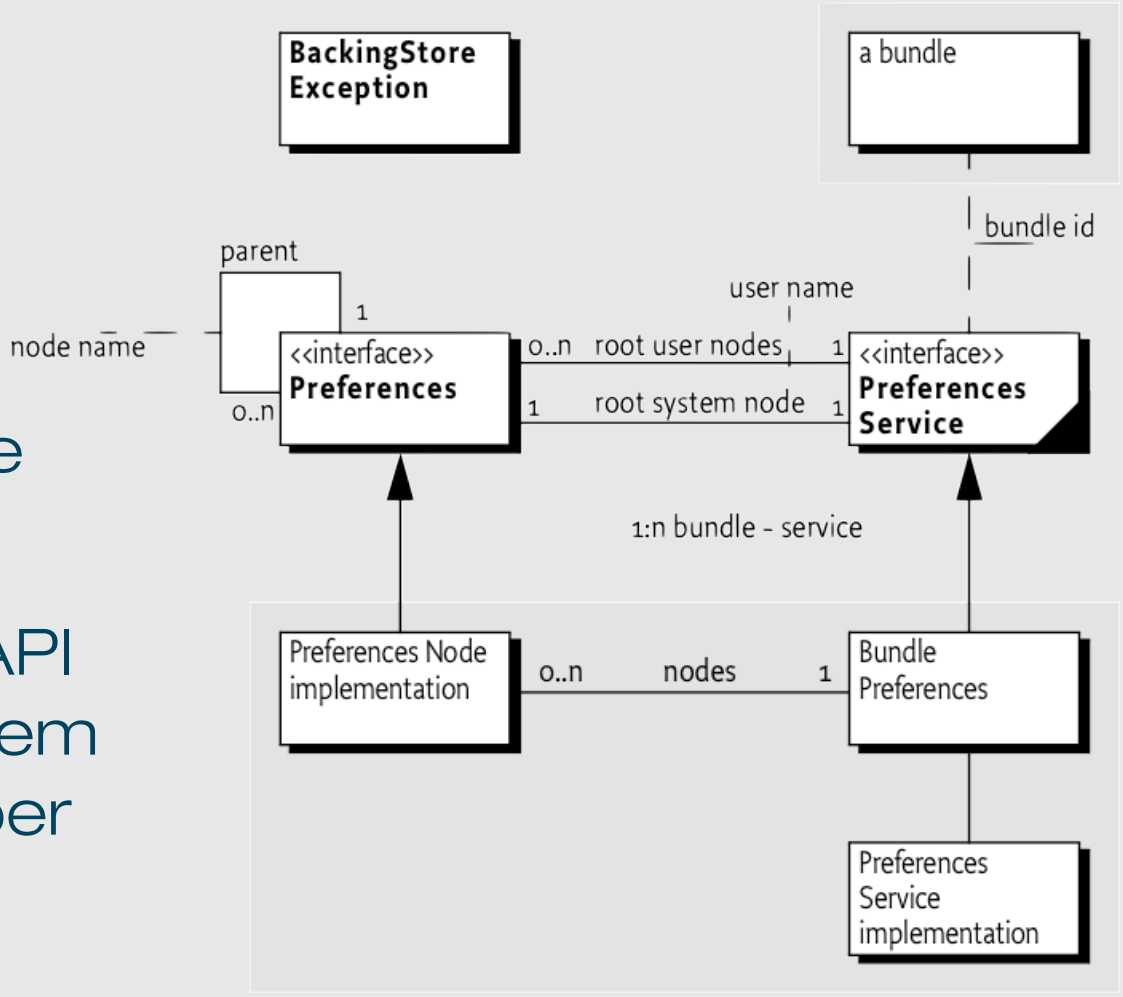
```
class Publisher {
    EventAdmin m_eventAdmin;

    public void send() {
        if (m_eventAdmin != null) {
            Dictionary properties = new Hashtable();
            properties.put("timestamp", new Date());
            m_eventAdmin.sendEvent(new Event("com/acme/timer", properties));
        }
    }
}
```

luminis

Preferences

- Preferences:
  - contains bundle private settings;
  - is coupled to the bundle life-cycle;
  - like the standard Java API there is a notion of system and user preferences per bundle.



luminis

Agenda

- History of OSGi
- The Framework
- The Compendium
- Patterns, Models & Embedding
- Open Source Frameworks

Service Whiteboard Pattern

- Instead of having clients look up and use a service interface, have clients register a service interface to express their interest
- The service tracks the registered client interfaces and calls them when appropriate
- This is called the **Whiteboard** pattern
  - It can be considered an Inversion of Control pattern

luminis

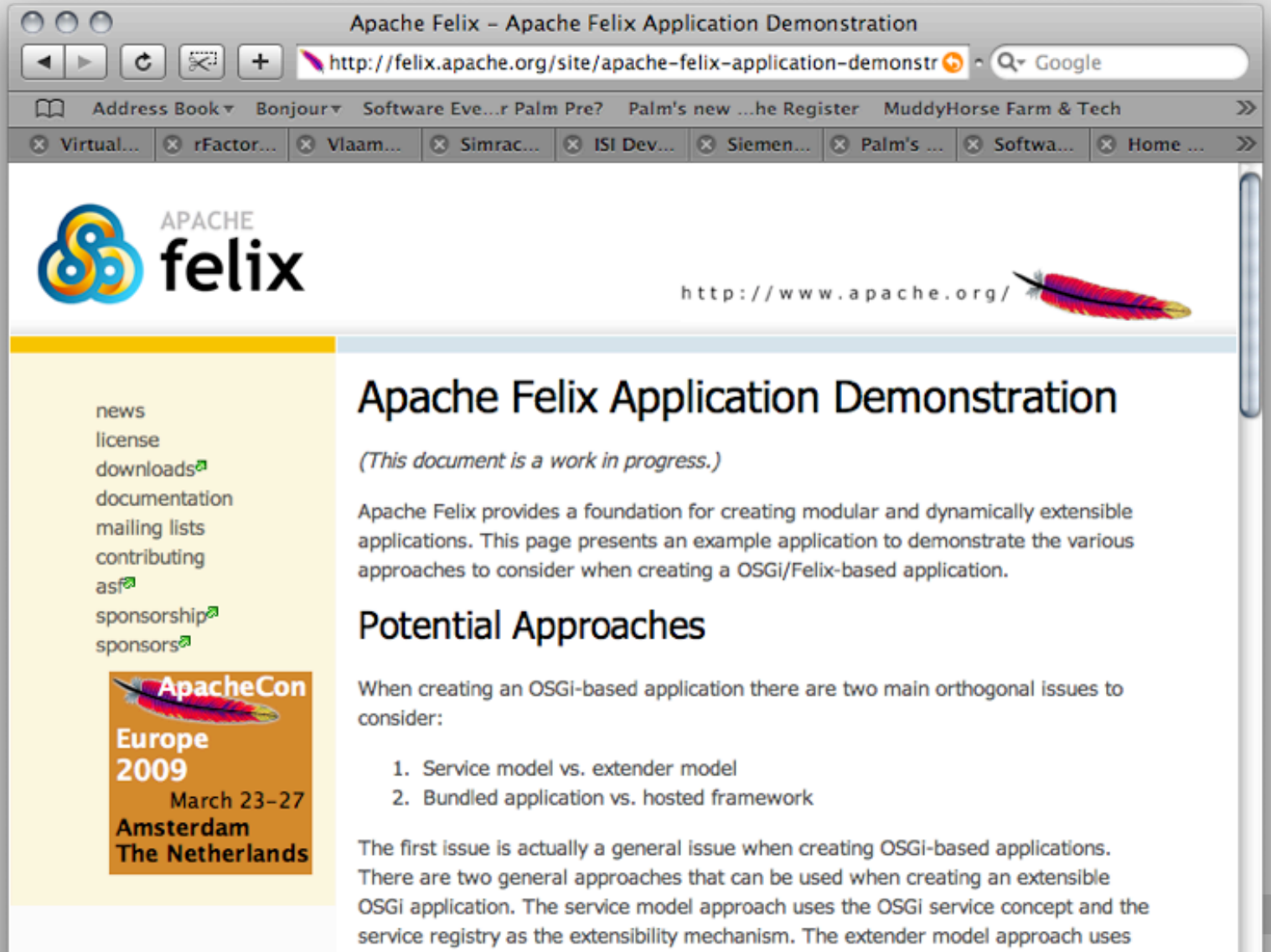
Service vs. Extender Models

- Two different approaches for adding extensibility to an OSGi-based application
  - The **service-based** approach uses the OSGi service concept and the service registry as the extensibility mechanism
  - The **extender-based** approach uses the OSGi installed bundle set as the extensibility mechanism
- Advantages and disadvantages for each
- Can be used independently or together

luminis

Example code

http://felix.apache.org/site/apache-felix-application-demonstration.html



Bundled vs. Hosted

- Applications can leverage OSGi functionality in two ways
  - **Bundled application**
    - Build entire application as a set of bundles that will run on top of a framework instance
  - **Hosted framework**
    - Host a framework instance inside the application and externally interact with bundles/services

luminis

Hosted Framework

- More complicated due to external/internal gap between application and framework
  - e.g., unlike bundles the host application does not have a bundle context by which it can access framework services
- Requires host/framework interactions
  - Accessing framework
  - Providing services to bundles
  - Using services from bundles
- Standardized API is in progress for R4.2

luminis

Hosted Framework

- Felix tries to simplify hosted framework scenarios
- Configuration data is passed into framework constructor
- Felix framework is the System Bundle
  - Gives the host application an intuitive way to access framework functionality
- Felix constructor also accepts „constructor activators“ to extend system bundle
- Felix tries to multiplex singleton resources to allow for multiple framework instances

luminis

Hosted Framework

```
public class Main {
    private static Felix m_felix = null;
    public static void main(String[] args) throws Exception {
        Map configMap = new HashMap();
        configMap.put(AutoActivator.AUTO_START_PROP + ".1",
            "file:bundle/org.apache.felix.shell-1.0.2.jar " +
            "file:bundle/org.apache.felix.shell.tui-1.0.2.jar");
        List list = new ArrayList();
        list.add(new AutoActivator(configMap));
        configMap.put(FelixConstants.SYSTEMBUNDLE_ACTIVATORS_PROP, list);
        try {
            m_felix = new Felix(configMap);
            m_felix.start();
            m_felix.waitForStop();
            System.exit(0);
        }
        catch (Exception ex) {
            System.err.println("Could not create framework: " + ex);
            ex.printStackTrace();
            System.exit(-1);
        }
    }
}
```

luminis

Hosted Framework

- Providing a host application service:

```
BundleContext bc = felix.getBundleContext();
bc.registerService(Service.class, svcObj, null);
```

- Accessing internal bundle services:

```
BundleContext bc = felix.getBundleContext();
ServiceReference ref =
    bc.getServiceReference(Service.class);
Service svcObj = (Service) bc.getService(ref);
```

luminis

Hosted Framework

- Classes shared among host application and bundles must be on the application class path
  - Disadvantage of hosted framework approach, which limits dynamics
  - Use of reflection by host to access bundle services can eliminate this issue, but it is still not an optimal solution
- In summary, better to completely bundle your application if possible

luminis

Open Source Implementations

- Apache Felix: http://felix.apache.org/
  - R4, originally called Oscar
- Knopflerfish 2: http://www.knopflerfish.org/
  - R4, open source version of UbiServ by Makewave
- Equinox: http://www.eclipse.org/equinox/
  - R4, initially developed for Eclipse and the RCP
- Concierge: http://concierge.sourceforge.net/
  - R3, optimized for resource constrained environments

luminis

Agenda

- History of OSGi
- The Framework
- The Compendium
- Patterns, Models & Embedding
- Open Source Frameworks

Coming soon:

- software distribution
- configuration management
- just started in the Apache Incubator

luminis

Any questions?

? & !