OSGi Service Platform

What is OSGi?

- The dynamic module system for Java
- A specification issued by the OSGi Alliance
- A universal runtime for embedded, client, server and cloud computing

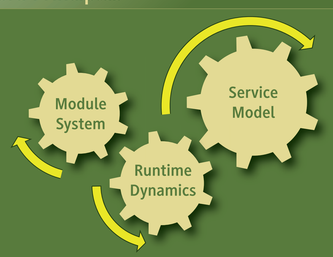
Popular OSGi Implementations

Apache Felix
Eclipse Equinox
Knopflerfish OSGi

Popular Development Tooling

Eclipse PDE
Bnd Tool

OSGi Core Principles



Module System

- Modularization reduces complexity and increases productivity
- Separating the API from the implementation lets you gain flexibility
- Modules provide a sound foundation for effective code reuse

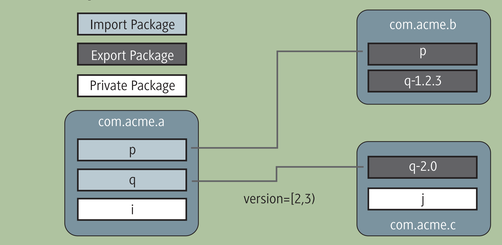
Runtime Dynamics

- OSGi brings modularization from development to run-time
- Installing, updating and uninstalling enables standardized hot deployment
- The same bundle can be deployed in multiple versions

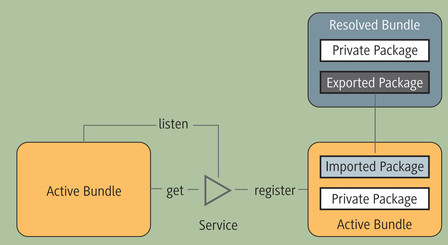
Service Model

- OSGi offers a publish-find-bind service model
- Service-orientation lets you gain flexibility through loose coupling
- Existing assets can be integrated transparently

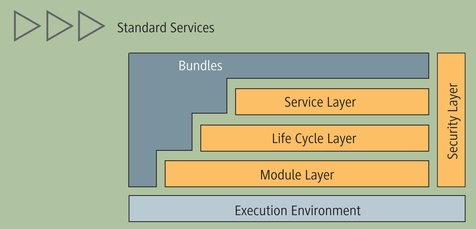
OSGi Dependencies



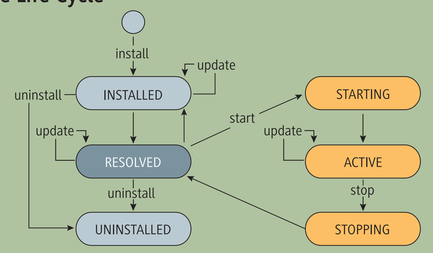
OSGi Framework



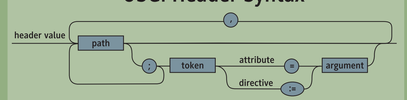
OSGi Architecture



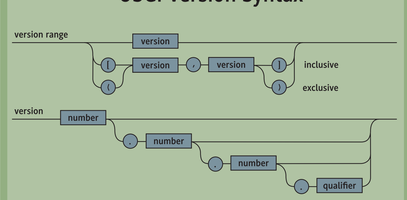
Bundle Life Cycle



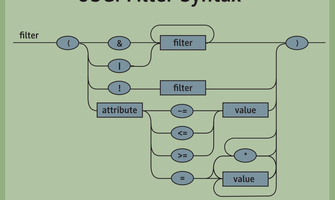
OSGi Header Syntax



OSGi Version Syntax



OSGi Filter Syntax



What is an OSGi Service?

- A POJO instance created by a Bundle and registered under a Service Interface
- Optionally described by Service Properties
- Can be looked-up and used by other Bundles

OSGi Standard Services

- Package Admin Service
- Start Level Service
- Conditional Permission Admin
- URL Handlers Service
- Log Service
- Http Service
- Device Access
- Configuration Admin Service
- Metatype Service
- Preferences Service
- User Admin Service
- Wire Admin Service
- IO Connector Service
- Initial Provisioning
- UPnP™ Device Service
- Declarative Services
- Event Admin Service
- Deployment Admin
- Auto Configuration
- Application Admin Service
- DMT Admin Service
- Monitor Admin Service
- Foreign Application Access

What is an OSGi Bundle?

- Contains coherent classes and resources
- Declares its public API
- Declares its dependencies
- Hides its internal implementation
- JAR deployment format

Important Bundle Manifest Headers

Manifest header	Meaning
Bundle-SymbolicName	Unique within OSGi Framework; use reserve domain name convention!
Bundle-Version	»major.minor.micro.qualifier«, e.g. »1.2.3.test«; default »0.0.0«
Bundle-Classpath	Lists all directories and/or JARs; default ».«
Bundle-Activator	A *BundleActivator* will be called for (de)activation
Import-Package	Lists all dependent packages
Import-Package	Lists all public API packages

Best Practices

- Make your Bundles cohesive: Use different Bundles for different concerns
- Use minimum Execution Environment in order to enable maximum reuse
- Prefer *Import-Package* over *Require-Bundle* in order to decrease coupling
- Apply versions to imports and exports in order to allow for fine-grained matching during resolving
- Make *Activator.start()* return fast in order to improve startup time
- Prefer *ServiceTracker* or a declarative service model over *ServiceListeners* in order to simplify programming
- Apply typical OSGi patterns like Whiteboard Pattern or Extender Model to leverage OSGi core functionality

Content: Weigle Wilczek GmbH, **Heiko Seeberger**