Dependency Manager - Singleton Component

Contents

Components are the main building blocks for OSGi applications. They can publish themselves as a service, and they can have dependencies. These dependencies will influence their life cycle as component will only be activated when all required dependencies are available.

Example usage

To define a singleton component, you can use the DependencyActivatorBase.createComponent() or the DependencyManager.createComponent() method, like in the following example which defines a "TranslationService" osgi service having one required dependency on the "LocalizationService" and one optional dependency on a "LogService". Dependencies are optional by default, unless you invoke the ServiceDependency.setRequired(boolean) method:

```java
public class GoogleBasedTranslationService implements TranslationService {
    volatile LocalizationService m_localizationService; // injected by reflection
    volatile LogService m_log;

    ...
}
```

```
public class Activator extends DependencyActivatorBase {
    public void init(BundleContext ctx, DependencyManager dm) throws Exception {
        Component c = createComponent()
           .setInterface(TranslationService.class.getName(), null)
           .setImplementation(GoogleBasedTranslationService.class)
           .add(createServiceDependency()
              .setService(LocalizationService.class, "(language=en)")
              .setRequired(true))
           .add(createServiceDependency()
              .setService(LogService.class)
              .setRequired(false)));
        dm.add(c);
    }
}
```

You can also inject dependencies using callbacks:

```java
public class GoogleBasedTranslationService implements TranslationService {
    volatile LocalizationService m_localizationService; // injected by reflection
    void bind(LogService log {...}
    ...
}

public class Activator extends DependencyActivatorBase {
    public void init(BundleContext ctx, DependencyManager dm) throws Exception {
        Component c = createComponent()
           .setInterface(TranslationService.class.getName(), null)
           .setImplementation(GoogleBasedTranslationService.class)
           .add(createServiceDependency()
              .setService(LocalizationService.class, "(language=en)")
              .setRequired(true))
           .add(createServiceDependency()
              .setService(LogService.class)
              .setCallbacks("bind", null /* no unbind method */)
              .setRequired(false)));
        dm.add(c);
   }
}
```

Notice that when you define an optional dependency without using callbacks, then a "NullObject" method is injected in the class field (by reflection) when the actual optional service is not available. In this case any invocation on the optional service won't do anything.

Contents