Eclipse SmartHome Using Bndtools
About Me

Jochen Hiller
Developer Evangelist
Deutsche Telekom AG
Seasoned Java Developer
OSGi and IoT Evangelist
Eclipse Concierge Committer
Migrating from PDE to Bndtools in Practice

QIVICON is an Eclipse SmartHome based solution from Deutsche Telekom AG. It utilises OSGi to provide a modular Java runtime.

Since the beginning, QIVICON leveraged Eclipse PDE with Maven & Eclipse Tycho as its build technology but over the time, the complexity increased. It became hard to get an overview and manage the runtime and build dependencies. Especially maintaining target configurations for IDE and CI/CD build, having different embedded gateways for installation increased complexity significantly.

Bndtools is the ‘swiss army knife’ in the context of OSGi development since it takes the nitty-gritty pains and loads off the developer’s chest. And that’s why we decided to avail the benefits of Bndtools.

But, many other OSGi-based projects still cannot avail the benefits as they are very tightly coupled with Eclipse PDE. Want to make a switch from your existing PDE source base to Bndtools? This talk would give you an overview to proceed towards this.

We would like to further demonstrate in this talk how to set up a Bndtools workspace from an existing PDE workspace, convert all current projects to Bnd projects and embrace the OSGi-way of developing bundles.

Since QIVICON containing more than 350 projects utilised this solution to move to a higher modularity maturity level, this talk would, therefore, outline the pros, cons and the learnings using Bndtools in such a big OSGi project for embedded development.
Semantic Versioning

Semantic Versioning
Technical Whitepaper

Revision 1.0
May 6, 2010
package org.eclipse.smarthome.io.net.http;
import org.osgi.annotation.versioning.Version;

@Version("1.0.0")
package org.eclipse.smarthome.io.net.http;
import org.osgi.annotation.versioning.Version;
OAuthClientServiceImpl

uses

HttpServletRequest

imports

HttpURLConnection

implements

HttpURLConnection

HttpURLConnection

uses

javax.servlet.http.HttpServletRequest

imports

javax.servlet.http.HttpServletResponse

implements

javax.servlet.http.HttpServletResponse

org.eclipse.smarthome.core.io.net.http

WebClientFactoryImpl

implements

WebClientFactory

imports

java.net.HttpURLConnection

uses

javax.servlet.http.HttpServletResponse

imports

javax.servlet.http.HttpServletRequest

implements

javax.servlet.http.HttpServletResponse

implements

javax.servlet.http.HttpServletRequest

imports

java.net.HttpURLConnection

uses

org.eclipse.smarthome.core.io.net.http

HttpServletRequest

imports

javax.servlet.http.HttpServletRequest

implements

HttpServletRequest

implements

HttpServletRequest

HttpRequest

HttpServletRequest

HttpURLConnection

HttpServletRequest

imports

java.net.HttpURLConnection

uses

javax.servlet.http.HttpServletResponse

imports

javax.servlet.http.HttpServletRequest

implements

HttpServletRequest

imports

java.net.HttpURLConnection

uses

HttpURLConnection

HttpServletRequest

imports

java.net.HttpURLConnection
Export-Package: 
org.eclipse.smarthome.io.net.http
{version=1.0.0, imported-as=[1.0,1.1)}
Import-Package
org.eclipse.smarthome.io.net.http
{version=[1.0,2)}