



Single-Sourcing Desktop, Web und Mobile-Applications mit Eclipse

Rabea Gransberger

Di. 24.09.2013

Desktop, Web und Mobile

- Eclipse bietet OSGi Framework
- SWT als OS natives UI Framework
- JFace als Container über SWT
- Single-Sourcing:
 - Eclipse RCP -> Desktop
 - Eclipse RAP -> Web
 - Tabris -> Mobile

OSGi



- OSGi Alliance, non-profit, Mär. 99
 - IBM, Oracle, Adobe, Telekom u.a.
- Spezifikation für modulare Java Applikationen
- Aktuell: OSGi Service Platform Release 5, 2012
- Implementierungen u.a.:
 - **Equinox** – getrieben von Eclipse (Enterprise)
 - Apache Felix – Apache Software Foundation
 - Knopflerfish – Gatespace
 - Concierge – für mobile und eingebettete Systeme

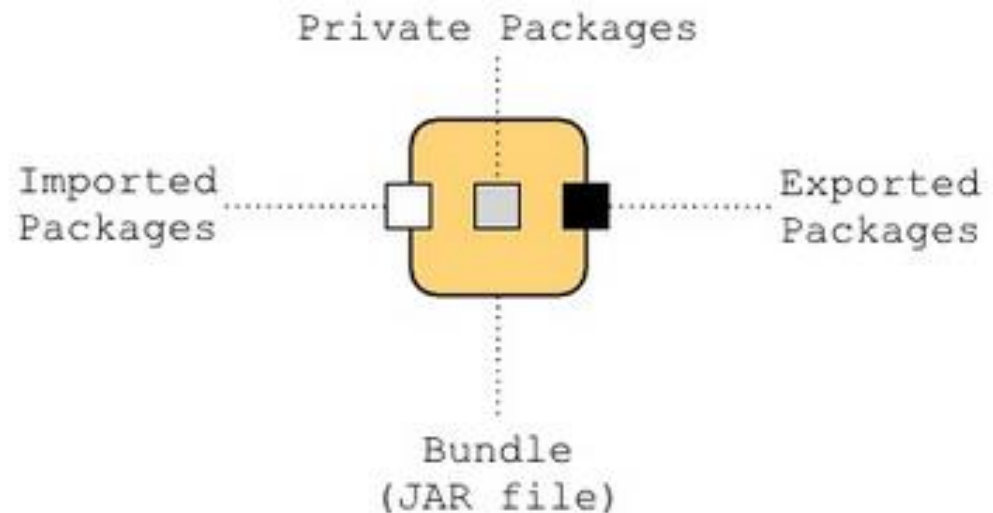
OSGi II



- Application Server Unterstützung:
 - Gemini Blueprint - Eclipse
 - WebSphere Application Server - IBM
 - Glassfish - Oracle
 - WildFly (ehem. JBoss AS) - JBoss

OSGi Bundle

- Modul-Basis sind OSGi Bundles
- Export von Packages für andere Plugins
- Import von Packages von anderen Plugins
- Definition in Manifest.MF
- Viele Frameworks bieten/sind OSGi Bundles



Manifest.MF Beispiel

Manifest-Version: 1.0

Bundle-ManifestVersion: 2

Bundle-Name: RCP App

Bundle-SymbolicName: de.jugbremen.app

Bundle-Version: 1.0.0.qualifier

Bundle-Activator: de.jugbremen.app.Activator

Bundle-Vendor: JUG Bremen

Require-Bundle: org.eclipse.ui

org.eclipse.core.runtime

Export-Package: de.jugbremen.app

Import-Package: org.eclipse.core.runtime

Bundle-RequiredExecutionEnvironment: JavaSE-1.7

Eclipse Equinox

- Equinox ist Eclipse OSGi Implementierung
- Eclipse Plugin == OSGi Bundle
- OSGi Service für Extension Points (plugin.xml)
 - Ermöglicht anderen Bundles an definierten Punkten die Funktionalität des eigenen Bundles zu erweitern oder zu verändern
- Eclipse selbst besteht aus Plugins

Eclipse RCP

- Eclipse Rich Client Platform
- Vordefinierte Bundles und Extension Points für eigene Desktop Applikation
- Features u.a.
 - Native Betriebssystem-Oberfläche
 - Jobs für Backgroundverarbeitung
 - Drag&Drop, Mehrmonitorunterstützung
 - Update-Mechanismus + Versionierung

SWT und JFace

- SWT ist ein UI Framework welches die nativen Komponenten des Betriebssystems zur Darstellung verwendet
- JFace bietet Standardkomponenten über SWT wie Wizards, Preferences, Progress Monitors
- SWT Programmiermodell ähnlich AWT: Komponenten bekommen Parent über Konstruktor

```
new Button(composite, SWT.PUSH)
```

ECLIPSE RCP DEMO

RCP Applikation erstellen

- Eclipse für RCP/RAP Entwickler runterladen
- File->New->Project
- Plugin Development->Plugin Project
- *Project Name*: de.jugbremen.rcp
- *Eclipse Version*: auswählen
- ***Next***

New Plug-in Project

Plug-in Project
Create a new plug-in project

Project name:

☒ Use default location
Location:

Project Settings
☒ Create a Java project
Source folder:
Output folder:

Target Platform
This plug-in is targeted to run with:
☒ Eclipse version:
☐ an OSGi framework:

Working sets
☐ Add project to working sets
Working sets:

- *Name und Vendor* eintragen
- `.qualifier` in Version wird bei deployment mit Timestamp ersetzt
- *Options: This plugin will make contributions to the UI*
- *Rich Client Application? Yes*
- ***Next***

New Plug-in Project

Content

Enter the data required to generate the plug-in.

Properties

ID: de.jugbremen.app

Version: 1.0.0.qualifier

Name: App

Vendor: JUG Bremen

Execution Environment: JavaSE-1.7 [Environments...](#)

Options

☒ Generate an activator, a Java class that controls the plug-in's life cycle

Activator: de.jugbremen.app.Activator

☒ This plug-in will make contributions to the UI

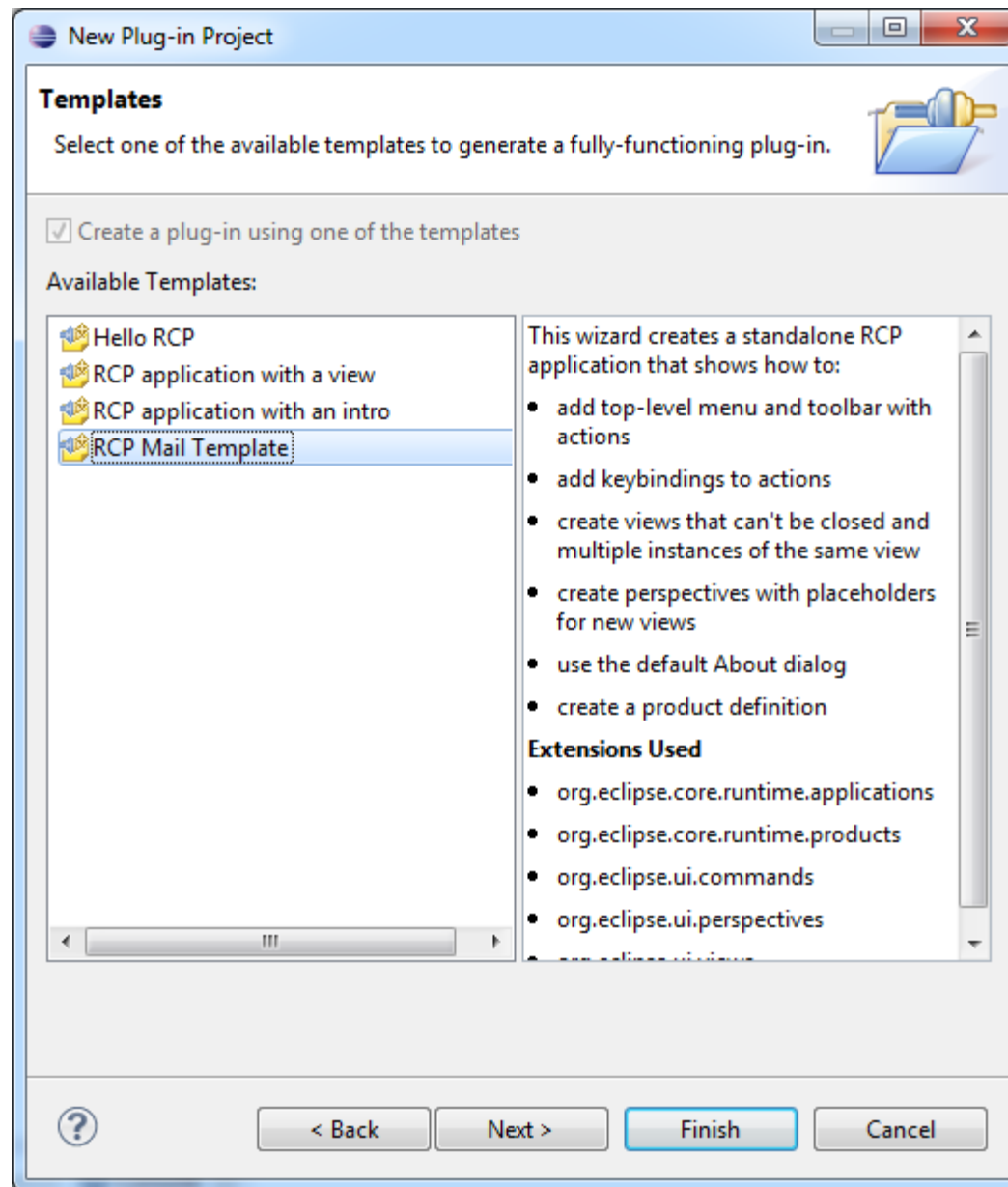
☐ Enable API analysis

Rich Client Application

Would you like to create a 3.x rich client application? ☒ Yes ☐ No

? < Back Next > Finish Cancel

- *RCP Mail Template* auswählen
- **Next**
- Auf der nächsten Seite Defaults beibehalten
- ***Finish***



New plug-in project with an RCP mail template


RCP Mail Template

This template generates a standalone RCP application, complete with views, menu and toolbar actions, keybindings and a product definition

Product name:

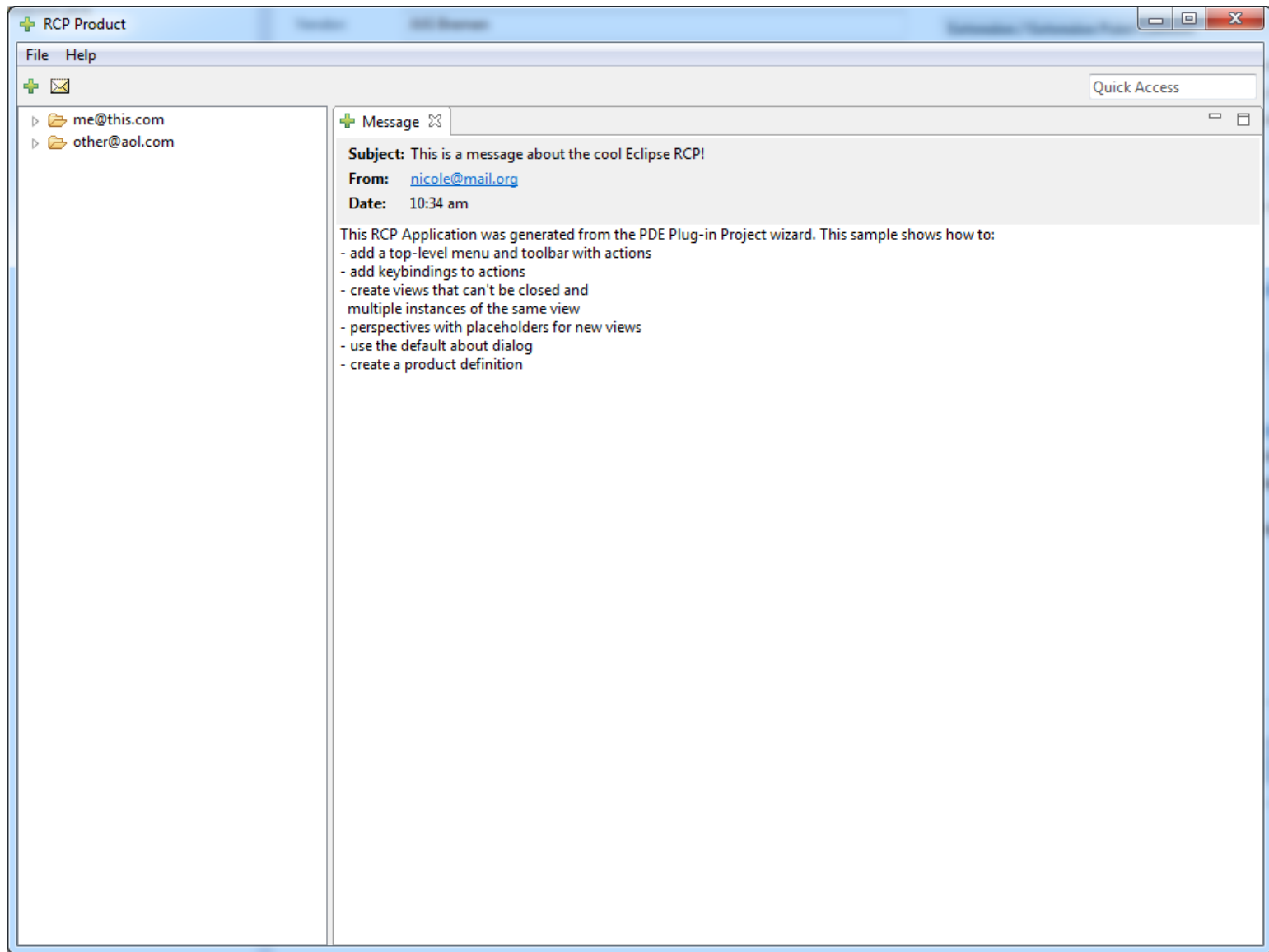
Package name:

Application class:



Applikation Starten

- Die `MANIFEST.MF` öffnen
 - Im Tab Overview im Bereich Testing
 - Launch An Eclipse Application
-
- Nun kann man sich die genrierte Sourcen anschauen, wie z.B. `plugin.xml` und `ApplicationWorkbenchWindowAdvisor`



SINGLE SOURCING?!

Single Sourcing

Aufbauend auf Eclipse RCP Framework

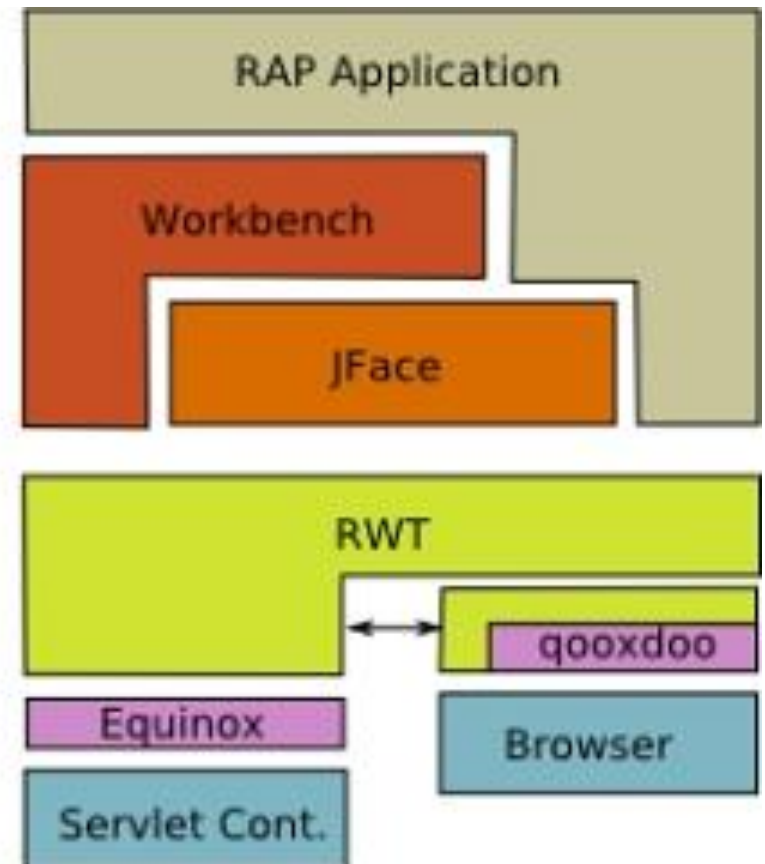
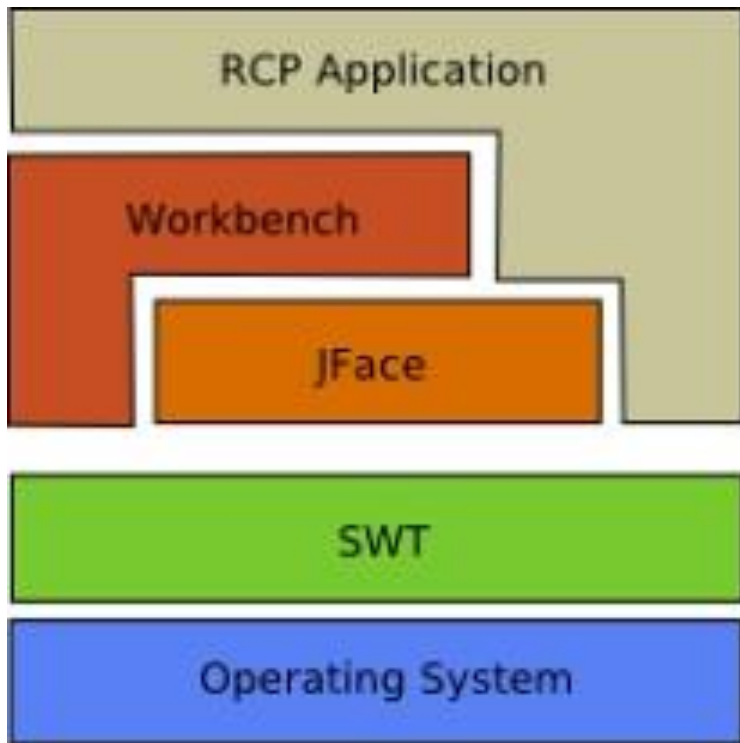
- Eclipse Remote Application Platform (RAP) für Web-Applikation
- Tabris für Web-Applikationen (iPhone, Android, HTML5)

Der selbe Code lässt sich mit den 3 Frameworks verwenden, optimal für kleine Entwicklerteams

Wie geht das?

- Code wird wie für Desktop mit SWT/JFace entwickelt
- RAP/Tabris verwenden andere Basis-Bundles
- RWT (RAP Widget Toolkit) nutzt HTML5 Rendering/JSON Kommunikation
- Dank OSGi sind Bundles äquivalent einsetzbar
 - `org.eclipse.rap.ui` statt `org.eclipse.ui`
 - Exportierte Packages /Klassen sind die selben

RCP + RAP Architektur



<http://www.infoq.com/articles/eclipse-rap-casestudy>

Aber....

- Oberflächengestaltung Desktop/Web/Mobile unterscheiden sich
- RAP/Tabris = Server / Multi-User Environment
- RAP/Tabris stellen Unterstützung für Browser-Navigation, GPS, Telefonbuch etc. bereit
- Nicht alle Features werden unterstützt

..... dazu später mehr

ECLIPSE RAP

Eclipse RAP



- Seit ca. 2007, damals von Innopract
- Volle Unterstützung in Eclipse
- Deployment als .war
- Theming über CSS
- Einfache Internationalisierung
- [RAP Widget Demos](#)
- Mittlerweile sehr gute Dokumentation

RAP Extensions

- Client / Session Handling
- HTTP File Upload
- HTML Markup für Widgets
- Browser und Navigations-Historie
- Auto Suggest

MIGRATION RCP ZU RAP

Migration RCP zu RAP

- Für RAP wird eine andere Target Platform (Gruppe von Plugins für die Kompilierung) benötigt:
- File->New->Other
- Plugin Development ->Target Definition
 - *Folder*: de.jugbremen.app
 - *File Name*: rap.target
 - *Initialize*: Nothing

New Target Definition

Target Definition

Create a new target definition.

Enter or select the parent folder:

de.jugbremen.app

de.jugbremen.app

File name: rap.target

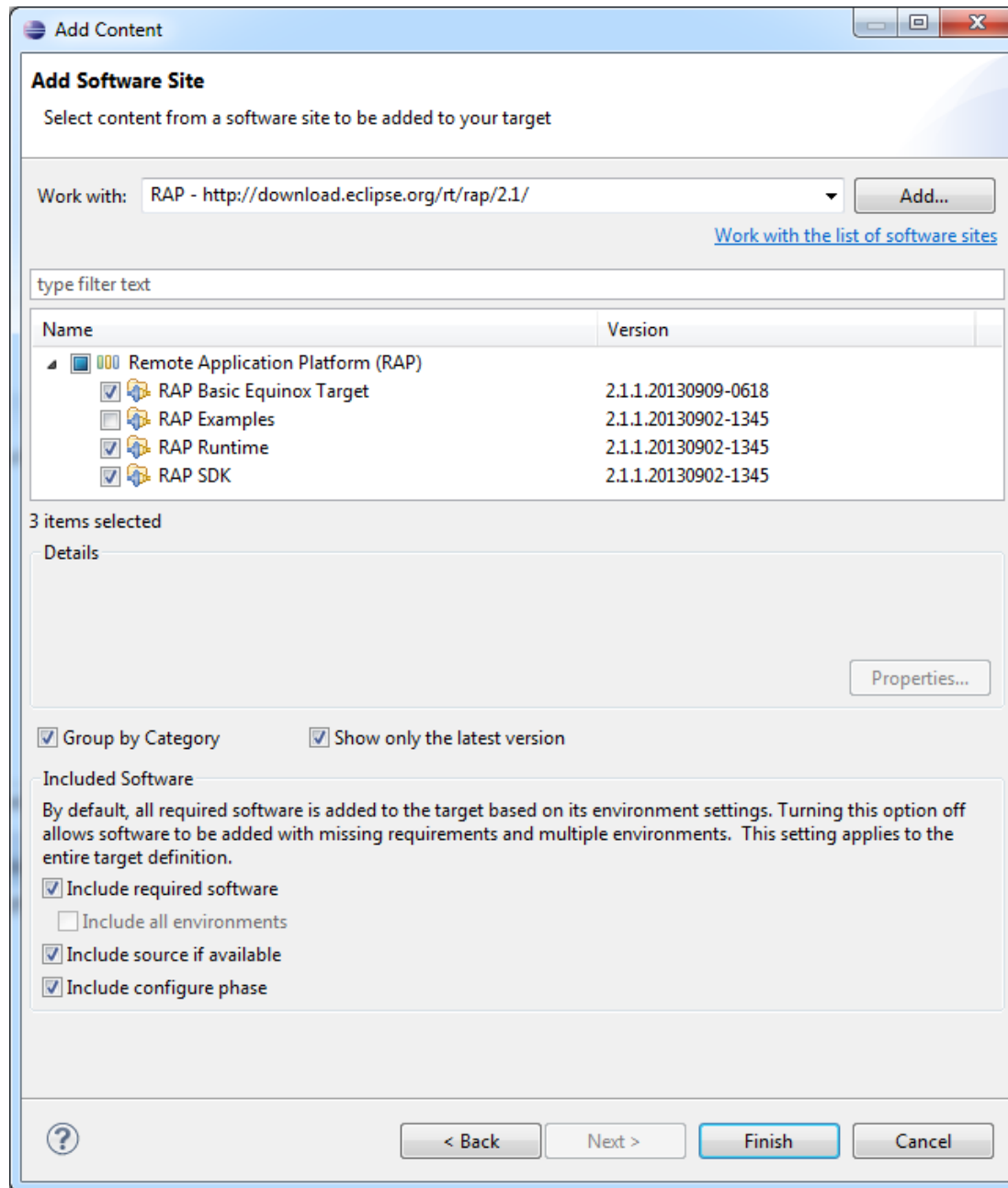
Initialize the target definition with:

- ☒ Nothing: Start with an empty target definition
- ☐ Default: Default target for the running platform
- ☐ Current Target: Copy settings from the current target platform
- ☐ Template: Base RCP (Binary Only)

? < Back Next > Finish Cancel

Target Platform

- *Locations: Add...*
- *Software Site*
- Work With:
<http://download.eclipse.org/rt/rap/2.1/>
- ***Enter***
- Alles außer *RAP Examples* auswählen
- ***Finish*** und *Speichern*
- Oben Links: *Set as Target Platform*



Fehler beheben

- Durch die Änderung der Target Platform sind Fehler entstanden
- MANIFEST.MF
 - Reiter *dependencies*
 - `org.eclipse.ui` auswählen
 - *Properties*: Auf *Optional* setzen
 - *Add* auswählen
 - `org.eclipse.rap.ui` eintippen
 - *Properties*: Auf *Optional* setzen

Fehler beheben

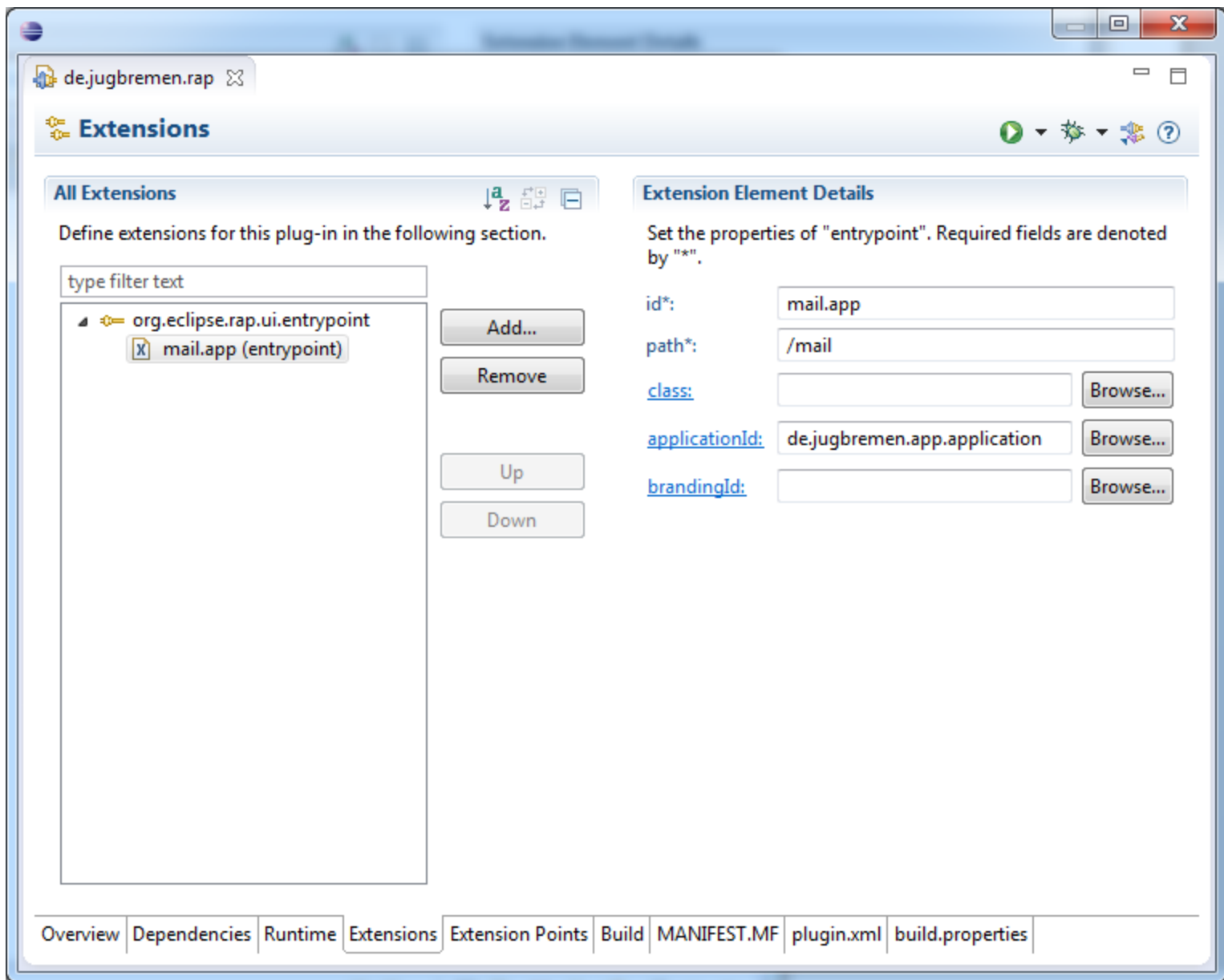
- `de.jugbremen.app.ApplicationActionBarAdvisor`
 - Feld `aboutAction` und alle Referenzen zunächst auskommentieren
 - In RAP ist die Implementierung eines About Dialoges nicht verfügbar, da es nicht wie in RCP ein Product mit den benötigten Infos gibt

Entrypoint definieren

- Die Applikation kompiliert nun
- Aber: Start als RAP noch nicht möglich
- Es muss erst ein *Entry-Point* definiert werden:
- File->New->Plugin Project
- *Name*: `de.jugbremen.rap`
- Alles so lassen nur:
 - Rich Client Application: No
 - kein Template

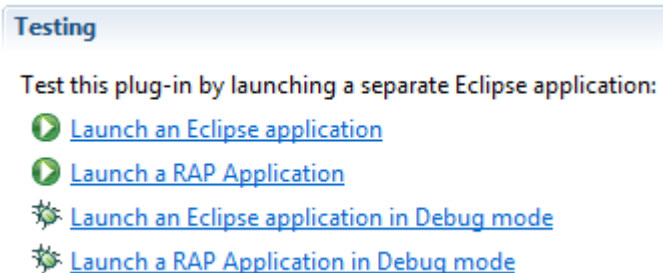
Entry Point

- `Manifest.MF->Dependencies`
- `org.eclipse.ui` **durch** `org.eclipse.rap.ui` **ersetzen**
- `de.jugbremen.app` **hinzufügen**
- **Reiter: Extensions**
- **Add:** `org.eclipse.rap.ui.entrypoint`
 - `Id: mail.app`
 - `Path: /mail`
 - `applicationId: de.jugbremen.app.application`

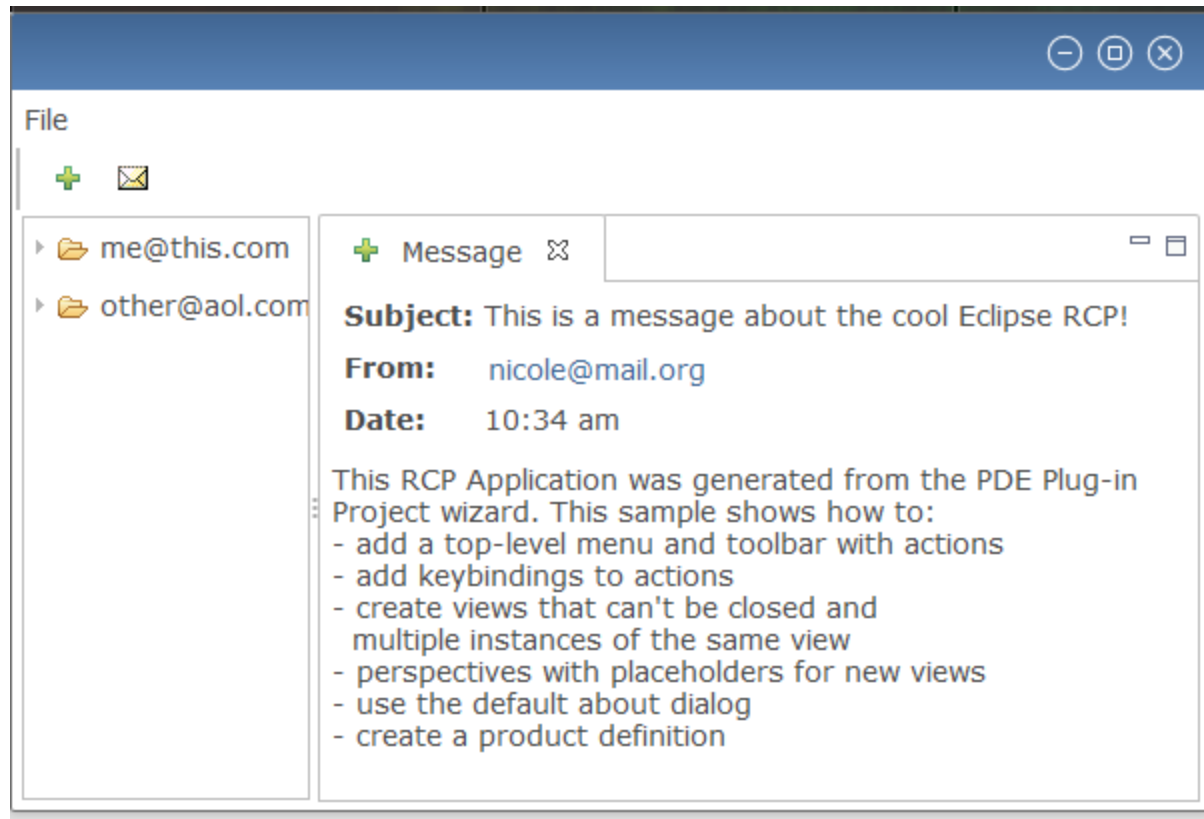


RAP Applikation starten

- In `de.jugbremen.rap`
- `Manifest.MF` öffnen
- Reiter: Overview -> Testing
- Launch a RAP application



RAP Mail Applikation



Migration RCP zu RAP (Kurzform)

- Target Platform (Gruppe von Plugins) ändern
- Require-Bundle durch Import-Package ersetzen oder `require org.eclipse.ui` durch `org.eclipse.rap.ui` ersetzen
- `org.eclipse.rap.ui.entrypoint` für RAP Applikation definieren
- Code der nicht kompiliert erst mal auskommentieren

Weitere Migrationsschritte

Weitere Schritte für eigene Applikationen

- rap.ui und eclipse.ui aus einem Plugin als optional exportieren
- Nicht kompilierende Teile in andere Plugins verschieben
- Singletons auf Multi-User Fähigkeit testen
 - SingletonUtil.getSessionInstance
- Hintergrund Threads in Jobs umwandeln/oder Server Push verwenden
- Drag & Drop überprüfen
- Keybindings und Mouse-Events ans Web anpassen
- Dialog blockieren nicht -> DialogUtil verwenden
- Bilder registrieren um Traffic zu reduzieren
- Oberfläche für Web optimieren

About Dialog anzeigen

- About Dialog ist noch auskommentiert, da er in RAP nicht funktioniert
 - RAP hat kein Produkt, woraus der Dialog in RCP erstellt wird
- **Ziel:** Zwei Implementierungen für RCP und RAP mit gleicher API die im Austausch verwendet werden können

About: Fehler beheben

- File->New->Plugin Project
- `de.jugbremen.rap.ui` (Rich Client:No, Template:No)
- **Neue Klasse:** `de.jugbremen.app>AboutFactory`
- **Neue Methode:** `public static IAction
create(IWorkbenchWindow window)`
- `Manifest.MF`, **Reiter: Runtime**
- **Export Packages: Add:** `de.jugbremen.app`
- (Package wird für andere Plugins freigegeben)

AboutFactory

```
package de.jugbremen.app;
import org.eclipse.jface.action.Action;
import org.eclipse.jface.action.IAction;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.ui.IWorkbenchWindow;

public class AboutFactory {

    public static IAction create(IWorkbenchWindow window) {
        return new Action("About") {
            @Override
            public void run() {
                MessageDialog.openInformation(null, "About", "My RAP About");
            }
            @Override
            public String getId() { return "testid"; }
        };
    }

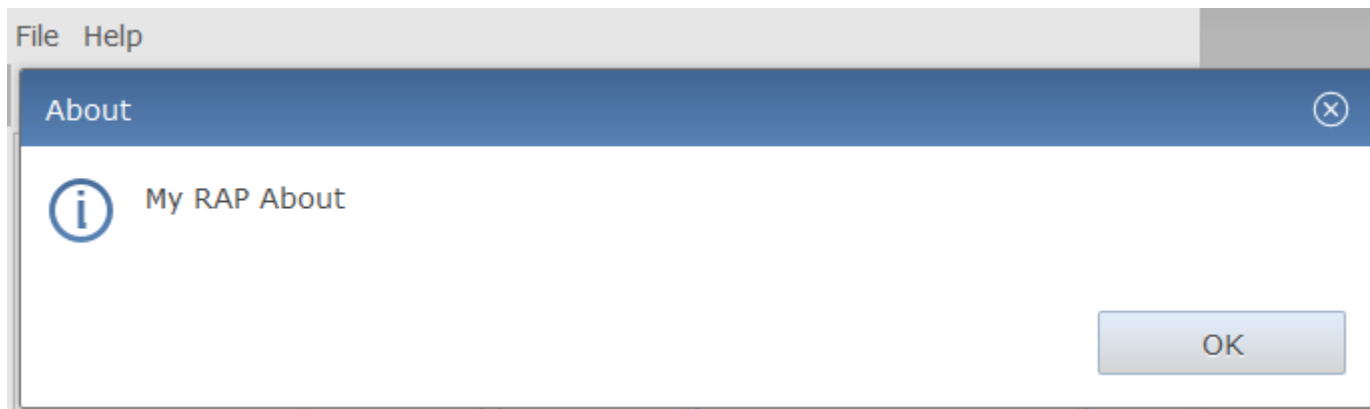
}
```

About: Fehler beheben

- In `de.jugbremen.app / Manifest.MF`
- **Dependencies->Add:** `de.jugbremen.rap.ui` (Optional)
- In `de.jugbremen.app / ApplicationActionBarAdvisor`
- **Code für Feld `aboutAction` einkommentieren**
- **Typ ändern auf `IAction`**
`aboutAction =`
`ActionFactory.ABOUT.create(window) ;`
- **Ersetzen durch**
`aboutAction = AboutFactory.create(window) ;`

About Dialog Testen

- Die RAP Applikation erneut starten
 - Ggf. Menü->Run->Run Configurations
 - Reiter: Bundles prüfen ob alle markiert
- Es gibt nun einen Menü-Punkt Help mit About



Single Sourcing?

- Wie kriegen wird nun Single Sourcing und unseren About Dialog in der RCP Applikation wieder angezeigt?
- Die Projekte `de.jugbremen.rap` und `de.jugbremen.rap.ui` **schließen**
- **Neues Plugin:** `de.jugbremen.rcp`
- **Hier:** `de.jugbremen.app>AboutFactory` implementieren
 - so dass sie das gleiche wie am Anfang tut

Single Sourcing!

- In `de.jugbremen.app` Dependency hinzufügen
- Target Platform auf RCP ändern
 - Preferences->Target Platform->Running Platform
- Als RCP Applikation starten!
- Nun kann man beliebig zwischen RCP und RAP „umschalten“
- Besser: 2 Workspaces mit den jeweiligen Targets und speziellen Plugins verwenden

TABRIS

Tabris



- Seit 2011, von EclipseSource
- Kostenpflichtiges Framework (ab 780€/Entwickler), 30 Tage Testversion
- Native Applikationen für iOS und Android
- Baut auf RAP auf
 - Nicht alle Widgets und geringere Zahl Plugins
- Interpretiert JSON vom Server als native Komponenten auf dem Device
- Eclipse Unterstützung für Android, Xcode für iOS

Tabris Extensions

- Branding/Export der Geräte App
- Spezielles UI Framework welches native Navigation der Betriebssysteme generiert
 - U.a. Swiping
- ClientDevice Objekt für Infos zum Gerät
- Reaktion auf Background/Foreground der App
- Geolocation API
- Sperrung App bei Inaktivität
- Speicherung kleiner Datenmengen auf dem Gerät
- AppLauncher: SMS, Mail, Browser, Telefonanruf etc. aus App starten

Migration Tabris

- Workbench ist nicht verfügbar
 - Keine Commands, Views, Menüs
 - Plugins aufteilen: nur SWT/JFace benutzen
- Navigations-Framework verwenden
- Langsame Kommunikation bedenken
- Bilder für versch. Auflösungen optimieren
- Erweiterungen für Geolocation etc. einbauen

FAZIT

Testen

- Plugin Spy mit Infos zu Widgets in RCP/RAP verfügbar
- RCP
 - UI Tests mit Jubula
 - Auch in CI Container lauffähig
- RAP
 - Spezielle Testfixture für Unit-Tests ohne UI
 - UI Test über Selenium

Vorteile Single Sourcing

- 3 Frameworks die aufeinander aufbauen: Einmal gelernt, 3 mal verwendet
- Moderne Weboberfläche ohne HTML/Ajax Wissen
- iOS und Android Clients ohne Objective-C und Android SDK
- Selbe Codebasis: Kein wiederholtes Re-implementieren
- Kunde erhält die Lösungen schneller
- Einfache Entwicklung durch schnellen Jetty und Debugging Tools
- RAP + Tabris werden immer mehr Features unterstützen, die dann aus RCP einfach integriert werden können

Links

- **RCP:**
http://wiki.eclipse.org/index.php/Rich_Client_Platform
- **RAP:** <http://eclipse.org/rap/>
- **Tabris:** <http://developer.eclipsesource.com/tabris/>
- **Single Sourcing Guide:**
<http://eclipsesource.com/en/about/contact-us/single-sourcing-guide-pdf-download/>
- **CAS PIA RAP Studie:**
<http://www.infoq.com/articles/eclipse-rap-casestudy>
- **EclipseCon Europe Vortrag:**
<http://www.youtube.com/watch?v=nbRKE1b8KyU&list>