Bachelorarbeit

Untersuchung der Portierung einer Eclipse RCP Applikation auf Eclipse RAP unter Berücksichtigung von Single Sourcing

zur Erlangung des akademischen Grades

Bachelor of Science Informatik

vorgelegt dem
Fachbereich Mathematik, Naturwissenschaften und Informatik der
Fachhochschule Gießen-Friedberg

Eugen Labun
im April 2009

Referent:	Sebastian Süß M. Sc.
Korreferent:	Professor Dr. Thomas Karl Letschert

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Gießen, 20. April 2009

Eugen Labun

Inhaltsverzeichnis

1 Einleitung

1.1 Über das Projekt

Technische Produkte und Anlagen werden immer komplizierter. Die Wartung und Fehlersuche wird immer schwieriger. Um eine effektive Analyse zu ermöglichen, werden die relevanten Daten innerhalb des ganzen Produktlebenszyklus gesammelt: Eine Lokomotive des Herstellers *Bombardier Transportation* verfügt zum Beispiel über rund 6000 Sensoren, die jede Sekunde Temperaturen, Spannungen, Stromstärken, Helligkeit, Feuchtigkeit, Druck, Position, Geschwindigkeit und andere Daten liefern. Die Auswertung der Daten wird durch die speziellen Computersysteme ermöglicht, die für den effektiven Umgang mit extragroßen Datenmengen ausgelegt und an die Anforderungen der Ingenieure angepasst sind.

Das Projekt "Parasuite" (*Product Analysis and Reporting Application Suite*) ist ein solches System, das von der Firma *Cognidata*[1] in Zusammenarbeit mit der Fachhochschule Gießen-Friedberg und der Philipps-Universität Marburg entwickelt wird. Industriepartner des Projekts sind der Lokomotivenhersteller *Bombardier Transportation*[2] und der Fahrstuhlhersteller *ThyssenKrupp Aufzüge*[3].

Ziel des Parasuite-Projekts ist die Entwicklung eines Entscheidungsunterstützungssystems zur vorausschauenden Produkt- und Anlagenwartung. Hierzu werden die erfassten Rohdaten strukturiert, mit Hilfe von speziellen Algorithmen nach Fehlermustern untersucht und über die mächtigen Filter in verschiedenen Darstellungen und Relationen zueinander den Ingenieuren zur Verfügung gestellt. Die Architektur des gesamten Systems ist flexibel gestaltet und ermöglicht die Anpassung an einen beliebigen Industriezweig.

1.2 Problemstellung

Die Benutzerinteraktion mit dem System erfolgte bislang über eine grafische Desktop-Anwendung, die mit Hilfe des Eclipse RCP Frameworks entwickelt wurde.

Nun sollte zusätzlich zu der vorhandenen RCP-Anwendung eine *Web-Applikation* entwickelt werden. Diese sollte den Benutzern den Zugang zu dem System auch dann ermöglichen, wenn die Desktop-Anwendung auf dem lokalen Computer nicht vorhanden ist. Die einzige benötigte Software wäre ein Standard-Webbrowser.

[1] http://www.cognidata.de/
[2] http://www.transportation.bombardier.com/
[3] http://www.thyssenkrupp-aufzuege.de/

Das Web-Interface würde die Benutzung des Systems flexibler machen und in den Fällen, in denen die lokale Installation der Java-Platform und der Desktop-Anwendung nicht möglich ist, sogar die einzig mögliche Art des Zugangs zum System darstellen.

Es wurden folgende *Hauptanforderungen* an die Web-Anwendung definiert:

- Das Benutzerinterface und Look&Feel der Web-Applikation soll möglichst identisch mit der Desktop-Anwendung sein;
- Es soll clientseitig nur HTML und JavaScript verwendet werden. Der Einsatz von Java-Applets, Adobe Flash oder ähnlichen Browser-Erweiterungen ist nicht erlaubt (dies soll die Benutzung des Systems mit allen modernen Betriebssystemen out-of-the-box ermöglichen).

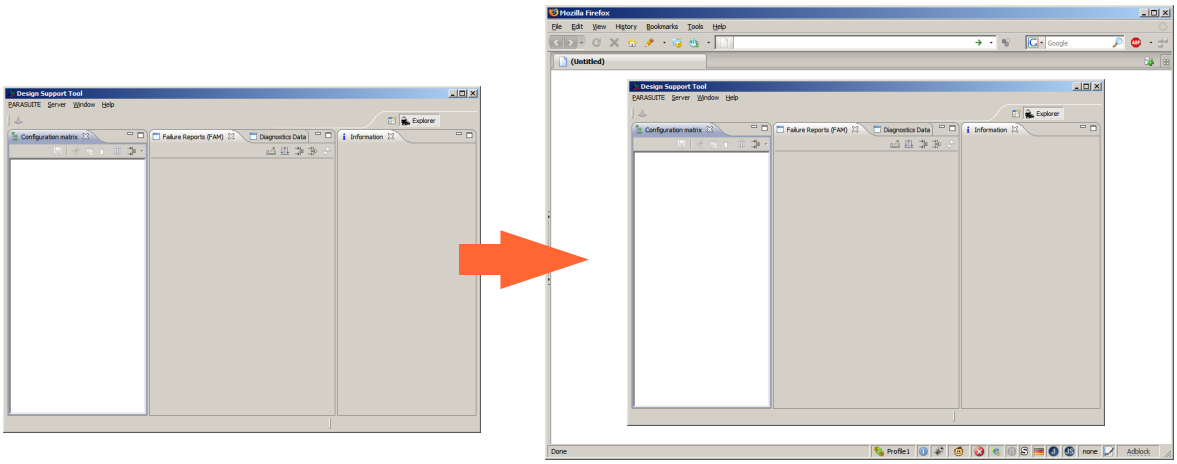


Abbildung 1.1: Problemstellung: RCP to Web

1.3 Aufbau dieser Arbeit

Die Arbeit ist in sechs Kapitel unterteilt.

Kapitel 1

Das Kapitel 1 ist die Einleitung. Hier werden das Projekt, das Ziel der Arbeit und der Aufbau der Arbeit beschrieben.

Kapitel 2

Das Kapitel 2 enthält Überlegungen zu Technologien, die für die Lösung der gestellten Aufgaben in Betracht gezogen wurden, sowie die Begründung der Entscheidung für Eclipse RAP.

Kapitel 3

Im Kapitel 3 erfolgt eine allgemeine Vorstellung der Eclipse RAP Technologie und die Beschreibung der Architektur des RAP Frameworks.

Kapitel 4

Parallele Entwicklung für RCP und RAP erfordert die Einigkeit in Begriffen, eine gute Vorstellung von OSGi und den Einsatz einiger nicht tradizioneller Methoden. Diesen Fragen ist das Kapitel 4 gewidmet.

Kapitel 5

Das Kapitel 5 behandelt die allgemeinen Ansätze der Portierung einer RCP Applikation auf die RAP Platform unter Berücksichtigung von Single Sourcing.

Kapitel 6

Im Kapitel 6 werden zunächst die Zukunft von Eclipse RAP und die Auswirkung dieser Arbeit auf das Projekt besprochen. Das Kapitel endet mit einer Zusammenfassung.

2 Wahl der Technologie

2.1 Eclipse RAP

Bei der Suche nach einer geeigneten Technologie wurde relativ schnell das Eclipse RAP[1] Projekt gefunden. Dieses Framework ist *exakt* für den oben genannten Zweck (Abb. 1.1) konzipiert, nämlich eine vorhandene Eclipse RCP Applikation webfähig zu machen.

Eclipse RAP entsprach vollständig den gestellten Anforderungen (siehe 1.2). Auch die Demo-Applikationen[2] waren vielversprechend und zeigten großes Potential. Die Belastbarkeit und allgemeine Performance sollte für die Projektzwecke ausreichen: ein Testserver mit Core2Duo 2.8 GHz CPU auf Ubuntu Hardy und Maximum 512 MB Heap Speicher für die JVM konnte einen Durchsatz von 100 Requests pro Sekunde erreichen, die durchschnittliche Antwortzeit betrug 14 ms[3].

Aus diesen Gründen galt Eclipse RAP von Anfang an als die favorisierte Technologie.

2.2 Andere Java-basierte Web-Frameworks

Es gab nicht so viele Java-basierte Web-Frameworks, die unter den gegebenen Umständen ernsthaft in Betracht gezogen werden konnten. Konkret wurden *JSF* und *GWT* ausgewählt und mit Eclipse RAP verglichen.

JSF (JavaServer Faces)

Zu Beginn der Arbeit wurde eine Realisierung der Aufgaben mit Hilfe von **JSF**[4] (JavaServer Faces) in Erwägung gezogen.

Im Gegenteil zu RAP unterstützt JSF aber keine Events-Programmierung. Eine MVC[5]-Modellierung ist also extrem schwierig. Praktisch müsste man manuell die

[1] Eclipse RAP Home @ eclipse.org. ⟨URL: `http://www.eclipse.org/rap`⟩ – Letzter Zugriff am 28.02.2009. Eclipse RAP Home @ innoopract.com. ⟨URL: `http://www.innoopract.com/de/eclipse-rap/`⟩ – Letzter Zugriff am 09.03.2009.

[2] `http://www.eclipse.org/rap/demos.php`

[3] MANUEL WOELKER, The new Eclipse download wizard and RAP performance. ⟨URL: `http://eclipsesource.com/blogs/2008/09/17/the-new-eclipse-download-wizard-and-rap-performance/`⟩ – Letzter Zugriff am 16.04.2009.

[4] `http://java.sun.com/javaee/javaserverfaces/`

[5] Model-View-Controller

Requests vorbereiten/auslesen, die Anbindung an eine JavaScript Library selbst programmieren und im Übrigen eine große Anzahl an low-level Kodierung durchzuführen.

Wegen dem zu großen Aufwand wurde diese Alternative zugunsten von Eclipse RAP abgelehnt.

GWT (Google Web Toolkit)

Das **GWT**[1] (Google Web Toolkit) ermöglicht mit der Java-Sprache auch Web-Widgets zu programmieren (Java wird von GWT in JavaScript übersetzt). Die API dieser Widgets unterscheidet sich aber gravierend von den Wokbench/JFace/SWT APIs, die in einer RCP Applikation verwendet werden. Die Wokbench- und JFace-Konzepte sind nicht vorhanden. Die exisierende Eclipse RCP Applikation müsste also im Falle des GWT-Einsatz neu implementiert werden. Das Single Sourcing von Desktop- und Web-Applikationen wäre nicht möglich. Für die Programmierung von stand-alone Web Applikationen, die nicht in ihren Look & Feel Eigenschaften an die bereits bestehenden Desktop Applikationen gebunden sein müssen, wäre dieses Toolkit aber sehr gut geeignet.

Performance: GWT vs RAP

Unter Berücksichtigung der großen Datenmengen mit denen im Projekt gearbeitet wird und der großen Tabellen die in der GUI-Anwendung schnell darstellbar sein müssen, wurden die Informationen über die Performance des GWT im Vergleich zu Eclipse RAP gesammelt.

Ein guter Vergleich von GWT und RAP ist in diesem Vortrag[2] enthalten. Laut diesem Vergleich schneidet GWT im Allgemeinen zwar etwas performanter als Eclipse RAP ab, jedoch nicht im entscheidenen Punkt – Darstellung großer Tabellen. Während eine RAP-Applikation nur die Daten zwischen Server und Client überträgt, die für die Darstellung der *sichtbaren Zeilen* einer Tabelle notwendig sind, sendet der Serverteil einer GWT-Applikation standardmäßig den *ganzen Inhalt* einer Tabelle an den Client.

Unter diesen Umständen musste die GWT-Alternative auch zugunsten der Eclipse RAP abgelehnt werden.

2.3 Nicht-Java-basierte Web-Frameworks

Die Web-Frameworks, die nicht auf der Java-Sprache basiert sind (was aber die Lauffähigkeit auf der Java-Platform noch nicht beeinträchtigt), wie Ruby / Ruby on Rails, Groovy / Grails u.a., sind natürlich auch eine ausgezeichnete Wahl für die Programmierung von Web-Anwendungen wenn die Voraussetzungen stimmen.

[1] http://code.google.com/webtoolkit/

[2] MARK RUSSELL/DANIEL RUBEL, Getting started with RAP development. ⟨URL: http://www.eclipsecon.org/2008/index.php?page=sub/&id=215⟩ – Letzter Zugriff am 16.04.2009.

Da in allen diesen Fällen zu den oben genannten Hürden noch die Notwendigkeit ein neues Framework/Sprache zu erlernen kommen würde, und Anzeichen einer besseren Anpassung an die gestellten Anforderungen nicht ersichtlich waren, wurden diese Frameworks nicht näher betrachtet.

2.4 Entscheidung

Gewählt wurde Eclipse RAP. Diese Technologie entsprach am meisten den gestellten Anforderungen. Für einen erfahrenen Eclipse RCP Entwickler versprach sie außerdem den geringsten Aufwand.

3 Vorstellung: Eclipse RAP

3.1 Einführung

Die Rich Ajax Platform (RAP) ist "die Eclipse Open Source Plattform für die Entwicklung von Desktop- und Webanwendungen auf Grundlage einer einheitlichen Codebasis".[1]

Das RAP Projekt wurde von der Firma Innoopract[2] initiiert und wird zur Zeit hauptsächlich durch die Mitarbeiter dieser Firma gepflegt.

Seit Sommer 2008 mit Erscheinen der Eclipse-Version "Ganymede" ist die RAP fester Bestandteil der Eclipse Distribution.

Im folgenden werden die wesentlichen Merkmale der RAP Platform betrachtet.

"RCP für's Web"

RAP ist in erster Linie für erfahrene Eclipse RCP Entwickler gedacht, die auf diese Weise ihre Desktop Applikationen webfähig machen können. Die Entwicklung folgt den bekannten Modellen unter Verwendung der gleichen Konzepte und Frameworks, wie bei der RCP Porgrammierung: Workbench, JFace, SWT. Das reduziert die Entwicklungszeit erheblich.

Die ganze Komplexität, verbunden mit Ajax, JavaScript und Client-Server-Kommunikation wird durch das RAP Framework gekapselt. Auch die Fragen der Kompatibilität mit verschiedenen Browsern werden von RAP übernommen. Der Entwickler verwendet ausschließlich die Java Sprache und eine bekannte API, die der Eclipse RCP API nahezu identisch ist.

Die Anlehnung der RAP-API an die RCP-API inklusive der Benennung von Packages, Klassen und Methoden ermöglicht die Verwendung einer einheitlicher Codebasis für die gleichzeitige Entwicklung von zwei Versionen der Applikation: für die Desktop- und für die Web-Anwendung. Der Anteil des gemeinsamen Code kann bis zu 70 – 90 % reichen.

Aus der Sicht des Benutzers ist das Look & Feel einer RAP Applikation und einer RCP Applikation fast identisch. Deshalb ist es zu erwarten, dass die RAP Applikationen eine hohe Akzeptanz finden werden, weil die Benutzer sich nicht an eine

[1] INNOOPRACT – Technologien und Dienstleistungen für Eclipse. ⟨URL: http://www.innoopract.com/de/⟩ – Letzter Zugriff am 08.03.2009.

[2] INNOOPRACT – Technologien und Dienstleistungen für Eclipse.

neue Oberfläche gewöhnen müssen. Die folgende Grafik (Abb 3.1[1]) zeigt eine RAP-Applikation, die äußerlich und von der Funktionalität her kaum von einer Desktop-Anwendung zu unterscheiden ist.

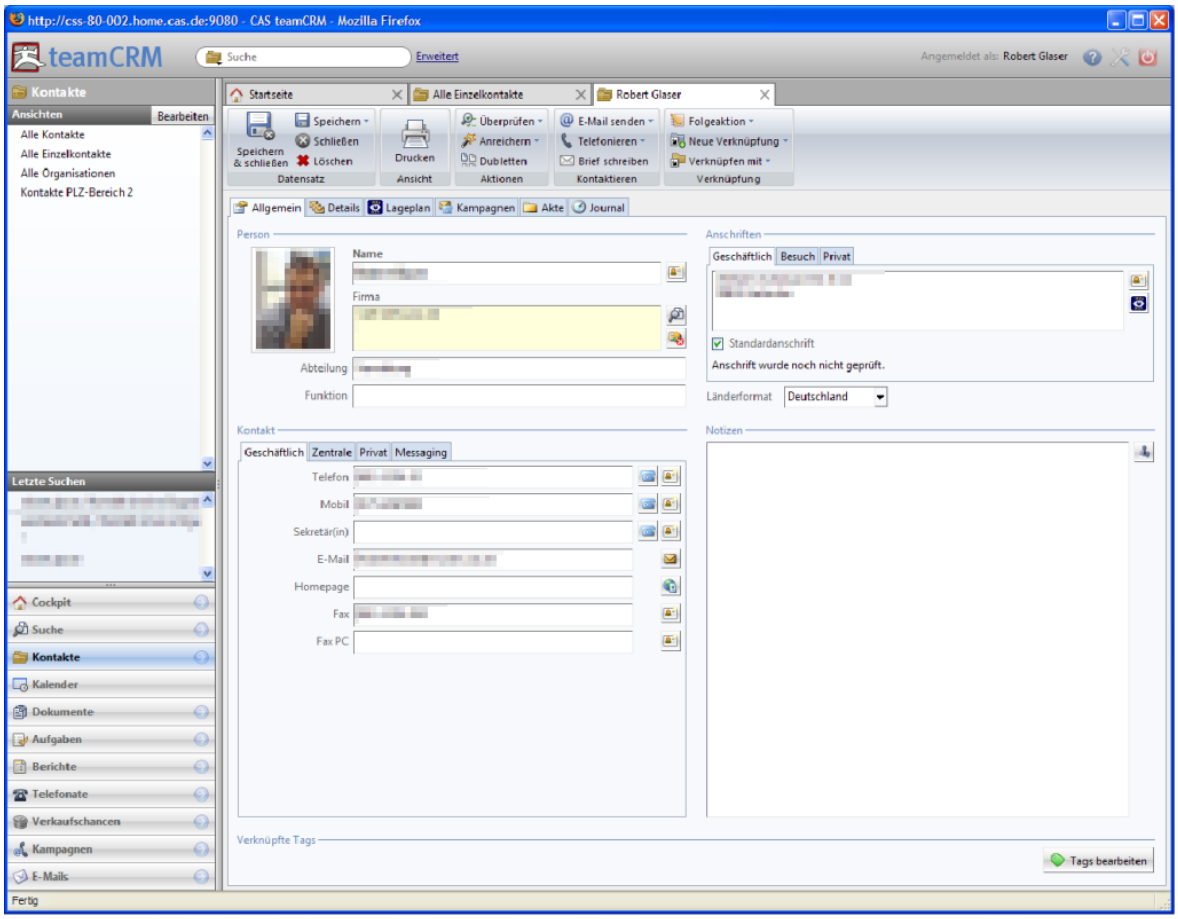


Abbildung 3.1: CRM Lösung von CAS Software

Programmierung mit Eclipse und Java API

Durch die Verwendung des Eclipse Plug-in Mechanismus, Java SWT Widgets und bewerter Techniken für die GUI-Entwicklung[2] kann die übliche Vorgehensweise zur Programmierung fast komplett beibehalten werden. Das Erlernen und die explizite Verwendung anderer Sprachen und Techniken wie JavaScript, ActionScript, HTML, Ajax, Flash usw. entfällt.

Natürlich sind dabei keine strengen Grenzen gesetzt: der Entwickler *kann* auf beliebige Technologien zugreifen um z.B. eine Funktionalität zu erhalten, die (noch) nicht in RAP vorhanden ist, etwa eigene JavaScript Widgets schreiben und anbinden.

[1] Quelle: ELIAS VOLANAKIS, Would you believe this is a browser based Application? ⟨URL: `http://rapblog.innoopract.com/2008/03/would-you-believe-this-is-browser-based.html`⟩ – Letzter Zugriff am 12.03.2009.

[2] wie WindowBuilder, SWT Designer oder Visual Editor

"Mit RAP können Entwickler auf preisgekrönte Java Entwicklungstools und auf die Plug-Ins von `eclipse.org` zurückgreifen. Da Applikationen als Bundles (Plug-Ins) und vollständig in Java entwickelt werden können sämtliche Arbeitsschritte vom Launching, über Debugging bis hin zum Exportieren von Standard `.war` Dateien direkt aus der Eclipse-IDE ausgeführt werden".[1]

Wiederverwendung von Eclipse Workbench, JFace, SWT

Auf die gleiche Weise, wie eine RCP Applikation gebaut wird, wird auch eine RAP Applikation gebaut. Dem Entwickler stehen dieselben Konzepte zur Verfügung, vor allem

- **SWT** als Sammlung von Widgets und Low-Level Graphics,
- **JFace** als bequeme Kombinationen von SWT Widgets und Modellen,
- **Workbench** als Integration von Views und Perspektiven.

Dabei muss beachtet werden, dass RAP die RCP-API nicht zu 100% implementiert. Dies hat zwei Hauptgründe:

- Die Implementierung ist *noch nicht* erfolgt (mangels zeitlicher oder menschlicher Ressourcen),
- Die Implementierung *wird nie erfolgen* wegen der Besonderheiten der Web-basierten Programme: manche Funktionalität ist dort nicht möglich oder nicht sinnvoll (z.B. eine zeitnahe Reaktion auf MouseMove Events wegen der Netzwerklatenz oder der direkte Zugriff auf das lokale Dateisystem).

RCP/RAP Single Sourcing

Parallele Entwicklung von Applikationen für Desktop und Web-Anwendung wird revolutionär vereinfacht. Wie bereits erwähnt wurde, ist sogar die Benennung der API-Elemente wie Packages, Klassen und Methoden identisch mit der RCP-API gehalten.[2]

Das Single Sourcing ist nicht nur auf RAP und RCP beschränkt. Auch die parallele Entwicklung für alle drei Eclipse Runtimes, also inklusive eRCP[3] (embedded RCP), ist möglich. Dies zeigt das Projekt EBERT der Eclipse Examples Seite.[4]

Dabei muss der Entwickler sich nicht auf den kleinsten gemeinsamer Nenner in Bezug auf die unterstützen APIs begrenzen. Es ist möglich die Applikationen so

[1] INNOOPRACT – Technologien und Dienstleistungen für Eclipse.

[2] Die Benennung der RCP und RAP-Plug-ins unterscheidet sich aus historischen Gründen. Vielleicht würde man aus heutiger Sicht betrachtet auch die Benennung von Plug-ins identisch mit ihren RCP-Gegenteilen machen. Dies würde das Single Sourcing noch weiter vereinfachen, aber auch die gleichzeitige Verwendung von RCP und RAP Plug-ins in einer Applikation unmöglich machen (für einen solchen hypotetischen Fall). Quelle: Rüdiger Herrmann, newsgroup eclipse.technology.rap, Oct 17 2008

[3] `http://www.eclipse.org/ercp/`

[4] WAYNE BEATON, Eclipse Business Expenses Reporting Tool (EBERT). ⟨URL: `http://www.eclipse.org/examples/example.php?id=expenses`⟩ – Letzter Zugriff am 10.03.2009.

zu programmieren, dass die RCP Applikation z.B. alle Möglichkeiten der RCP-Platform benutzt und die RAP und eRCP Anwendungen auf die etwas begrenzten Möglichkeiten ihrer jeweiligen Platform zurückgreifen.

Natürlich funktioniert in der Praxis nicht alles reibungslos. Die Probleme sind jedoch überwindbar und die Gesamteffektivität übersteigt andere Formen der parallelen Entwicklung für Desktop und Web.

Theming und Branding

Als Nachteil von RAP Applikationen *kann*[1] gesehen werden, dass sie einer Desktop/RCP Applikation zu ähnlich sind, was im Web eher ungewöhnlich ist.

Um das Aussehen von RAP Applikationen der Mehrheit von modernen Web Applikationen ähnlicher zu machen und die Anwendung entsprechend dem Corporate Design zu gestalten können Anpassungen am Look & Feel durch Theming[2] und Branding[3] vorgenommen werden. Einen weiteren Fortschritt bietet das RAP Extended Presentation API[4] (momentan noch im beta-Zustand).

3.2 RAP Architektur

Aus Sicht des Programmierers sehen die RCP und RAP Applikationen zunächst beinahe identisch aus (Abb. 3.2). Beide stützen sich auf die APIs von Workbench, JFace und SWT/RWT (RWT[5] ist das Gegenstück von SWT in RAP).

Die Unterschiede sind jedoch gravierend.

Eine *RCP Applikation* ist eine Desktop Applikation, sie läuft lokal auf dem Computer, unmittelbar auf dem darunterliegenden Betriebssystem (die Applikation wird zusammen mit der JVM betrachtet).

Eine *RAP Applikation* ist verteilt:

- ein Teil läuft auf dem Server. In der Abbildung ist gezeigt, dass die Applikation innerhalb der Equinox (OSGi Runtime Implementierung), und die letzte in einem Servlet Container (z.B. Tomcat) läuft.
- ein anderer Teil läuft auf dem lokalen Computer des Benutzers. Der Browser des Benutzers stellt dabei das unterste sichtbare Abstraktionslevel dar, das im Falle der RCP Applikation dem Betriebssystem bzw. der JVM entspricht. Als clientseitiges JavaScript Framework wird die `qooxdoo`[6] Library verwendet.

[1] "kann", weil es auch als Vorteil im Sinne der einheitlichen Benutzererfahrungen gesehen wird
[2] `http://help.eclipse.org/ganymede/topic/org.eclipse.rap.help/help/html/advanced/theming.html`
[3] `http://help.eclipse.org/ganymede/topic/org.eclipse.rap.help/help/html/advanced/branding.html`
[4] HOLGER STAUDACHER, Extended Presentation API. ⟨URL: `http://eclipsesource.com/blogs/2008/12/18/extended-presentation-api/`⟩ – Letzter Zugriff am 11.03.2009.
[5] RAP Widget Toolkit – general introduction: `http://wiki.eclipse.org/WidgetToolkit`
[6] `http://qooxdoo.org/`

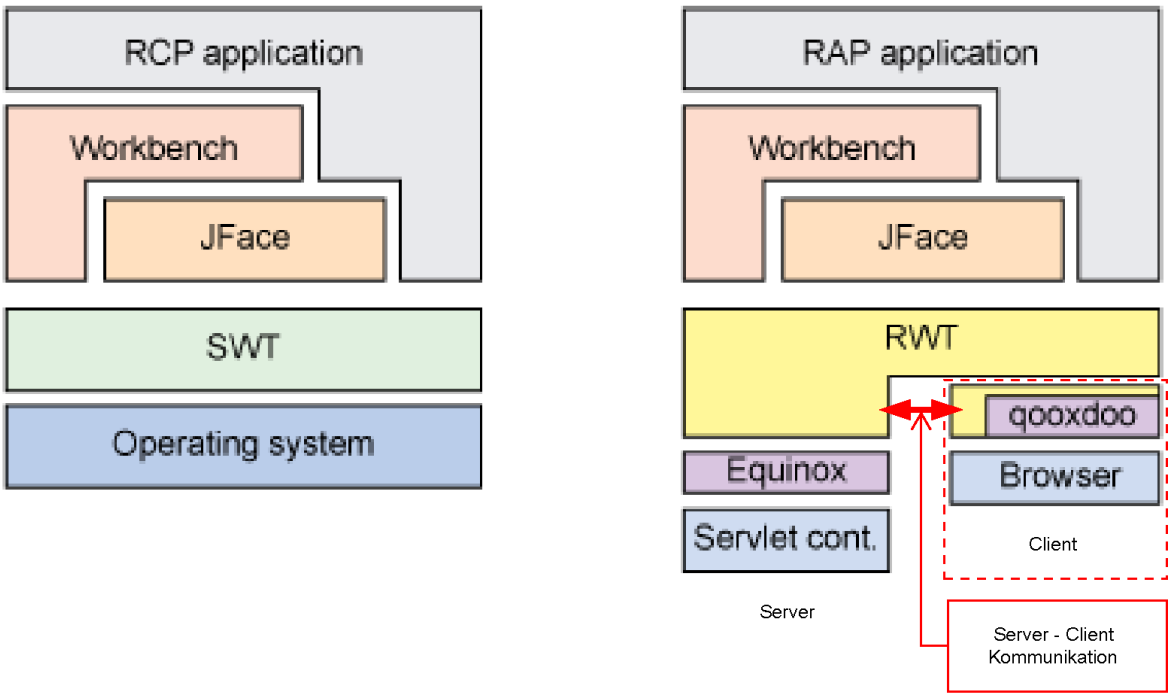


Abbildung 3.2: RCP vs RAP

Diese Bibliothek haben die RAP Entwickler wegen der Ähnlichkeit ihrer API zu API von SWT-Widgets gewählt.

Client - Server Kommunikation

Die Client - Server Kommunikation findet in RAP mittels HTTP-Protokoll auf folgende Weise statt (Abb. 3.3):

1. Der Benutzer ruft mit dem Browser eine bestimmte Adresse (URL) auf.
2. Der HTML- und JavaScript Code wird in den Browser geladen und ausgeführt. Der Benutzer sieht das initiale GUI.
3. Nach jeder Benutzer-Interaktion (Maus, Tastatur) wird ein Ajax-Request (XMLHttpRequest) mit der Information über dieses Ereignis an den Server gesendet.
4. Jeder dieser Requests löst beim Server die sog. **RWT life cycle**[1] aus:
   - *Prepare UI Root*: verantwortlich für den Aufruf von *entry points* (Implementierungen von `IEntryPoint`).
   - *Read Data*: Auslesen von Request-Parameter und Synchronisierung des Zustandes von JavaScript-Widgets im Browser und RWT-Widget-Instanzen serverseitig (z.B. Aktualisierung der `text`-Eigenschaft eines TextInput-Controls).

[1] `http://wiki.eclipse.org/WidgetToolkit#LifeCycle`, `http://help.eclipse.org/ganymede/topic/org.eclipse.rap.help/help/html/reference/api/org/eclipse/rwt/lifecycle/ILifeCycle.html`

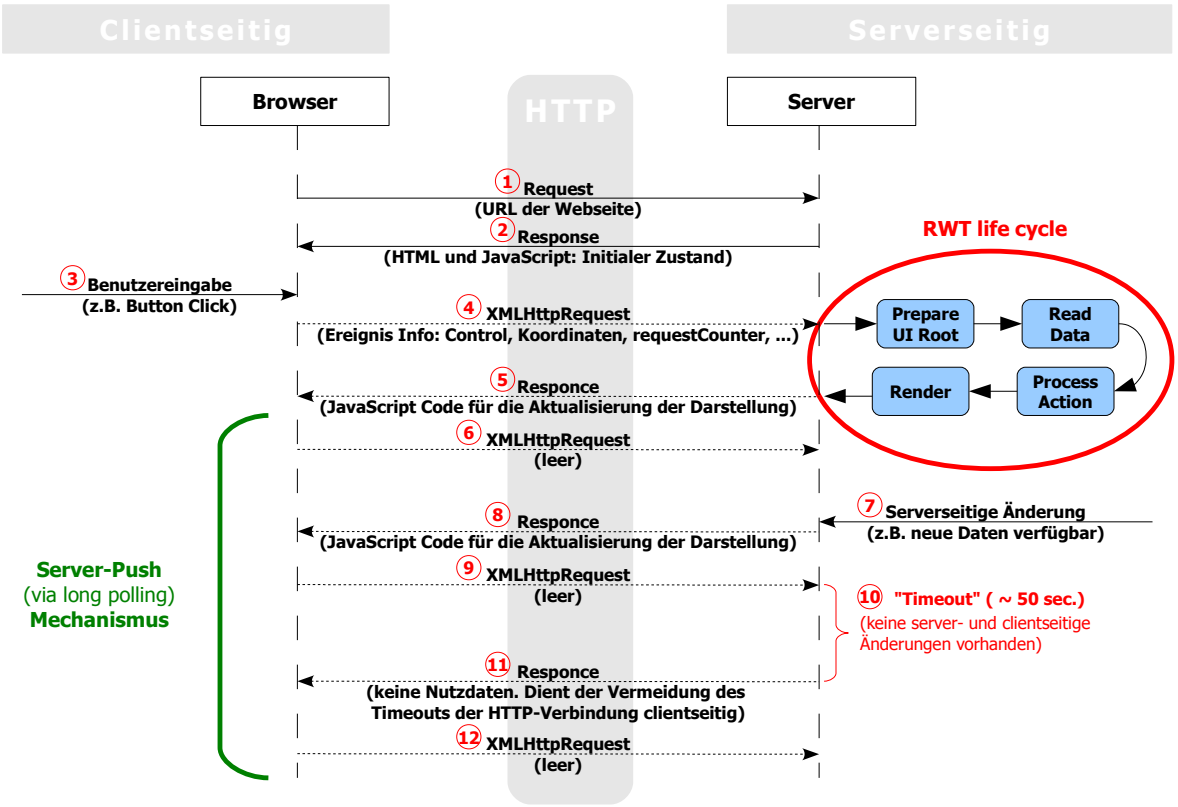


Abbildung 3.3: RAP Client-Server Kommunikation

- *Process Action*: Unmittelbare Verarbeitung der Benutzeraktion durch Aufruf von Listeners, die für die entsprechenden Events registriert wurden.
- *Render*: Generierung des JavaScript Code für die Aktualisierung der Darstellung clientseitig. Nur die veränderten Daten werden gesendet. Das resultiert in einer sehr kleinen Datenmenge, oft nur ein paar Hundert Bytes.

5. Der generierte JavaScript Code wird an den Client geschickt. Der Browser führt den Code aus und der Benutzer sieht das aktualisierte GUI.

   Um die Aktualisierung des clientseitigen Zustandes bei serverseitigen Änderungen zu ermöglichen wird der **Server Push**[1] Mechanismus mittels sog. *long polling* verwendet. Dieser Mechanismus ist in RAP standardmäßig deaktiviert. Die Aktivierung erfolgt durch den Aufruf von `UICallBack.activate`[2]. Dadurch wird erzwungen, dass der Client ständig einen (leeren) Request offen hält.

6. Da eine HTTP-Verbindung[3] technisch nur von dem Client initiiert werden kann, sendet der Client einen *leeren* (d.h. ohne Nutzdaten) Request an den

[1] ServerPushFAQ. ⟨URL: `http://code.google.com/p/google-web-toolkit-incubator/wiki/ServerPushFAQ`⟩ – Letzter Zugriff am 16.04.2009.
[2] `http://help.eclipse.org/ganymede/topic/org.eclipse.rap.help/help/html/reference/api/org/eclipse/rwt/lifecycle/UICallBack.html#activate(java.lang.String)`
[3] Im Sinne Request/Response Paar.

Server, allein mit dem Zweck, dass dieser Request vom Server beantwortet werden *kann*. Auf solche Weise kann der Client von den Änderungen auf der Serverseite erfahren.

7. Falls serverseitig irgendwelche Änderungen stattfinden, z.B. wenn neue Daten verfügbar sind, kann der Server dem Client dies sofort mitteilen (nächster Schritt).
8. Der Server antwortet auf den (leeren) Request des Clients mit den neuen Daten. Der Client aktualisiert die Darstellung.
9. Der Client sendet erneut einen leeren Request, um auf die nächste serverseitige Veränderung sofort reagieren zu können.
10. Die Antwortzeit auf den Request des Clients kann nicht beliebig lange sein. Nach einem Timeout – meistens ca. 1 Minute – kann der Browser entscheiden, dass der Server nicht mehr antworten wird und schließt die Verbindung. Der eventuell später eintreffende Response des Servers wird in diesem Fall vom Browser ignoriert. Um eine solche Situation zu vermeiden sollte ein serverseitiger "Timeout" eingerichtet werden (momentan nicht standardmäßig in RAP vorhanden), der ca. alle 50 Sekunden einen Response auslöst, allein mit dem Zweck einen neuen Request vom Client zu erhalten.
11. Leerer Response vom Server nach "Timeout".
12. Leerer Request vom Client (wie 9).

Dies ist eine vereinfachte Darstellung, in Wirklichkeit ist der Prozess etwas komplizierter. Zum einen wird nicht jedes Event sofort an den Server geschickt. Manche werden erst mit einer gewissen Verzögerung (z.B. die Mausposition nach 200 ms und nicht alle 5 ms), manche gar nicht geschickt. Wenn möglich, wird ein Event direkt im Browser bearbeitet, ohne ein Request an den Server zu senden (z.B. eine ToolTip-Anzeige). Solche Besonderheiten der Implementierung sind durch die *Netzwerklatenz* bedingt: der Event-Austausch kann dadurch nicht so schnell wie bei einer Desktop Applikation erfolgen.

Zum Glück muss man nicht unbedingt in Details der Client - Server Kommunikation einsteigen. Das RAP Framework wurde letztendlich gerade mit dem Gedanken erfunden, den Entwickler von dieser Last zu befreien.

4 OSGi und Entwicklungsumgebung

Bei der Vertiefung in Eclipse RAP stellte sich heraus, dass die richtige Verwendung dieses Frameworks ein hohes Maß an Verständnis der zugrunde liegenden Technologien – OSGi, Eclipse Workbench/JFace/SWT, Eclipse RCP – und sehr gute Java-Kentnisse erfordert.

Die Fragen der Einrichtung der Entwicklungsumgebung und *best practices* der Entwicklung erwiesen sich auch als sehr wichtig, denn dabei werden – bedingt durch den RAP-Einsatz – einige nicht offensichtliche Methoden verwendet.

Weiterhin sind bei der Zusammenarbeit im Team die Richtlinien für die Projektführung und Ansätze zur Automatisierung der Einstellungen an den einzelnen Arbeitsplätzen wichtig.

4.1 Begriffe und Konzepte

Bei der Software Entwicklung unter Einsatz von Eclipse verwendet man oft einige Begriffe, wie *Project*, *Plug-in*, *Workspace*. Die Bedeutung dieser Begriffe im softwaretechnischen Sinn unterscheidet sich aber von ihrer buchstäblichen oder angenommenen Bedeutung. Das einheitliche Verständnis ist aber insbesondere bei der Arbeit im Team an einem großen Projekt notwendig. Deswegen wird hier kurz darauf eingegangen.

Ein ***Project*** *innerhalb der Eclipse IDE* ist meistens als ***Modul*** *im software-technischen Sinn* zu verstehen. Mehrere solche Module, zusammengefasst im *Eclipse* ***Workspace***, bilden ein ***Projekt*** *im software-technischen Sinn*. Die Zusammenfassung von Modulen in Subsysteme kann in Eclipse durch den Einsatz von **Working Sets**[1] erfolgen.

Der Eclipse **Workspace** soll in erster Linie nicht als der Speicherort der Module verstanden werden (obwohl es dazu verwendet werden kann und oft verwendet wird), sondern als der Speicherort für projektbezogene *Einstellungen*. Es ist nicht selbstverständlich, dass die Quellcodedateien sich physikalisch im Workspace-Ordner befinden. Sie können auch an einem beliebigen anderen Ort liegen. Der Workspace enthält *Links* auf alle Projektdateien. Gerade im Zusammenhang mit RCP/RAP Single Sourcing wird dieses Konzept konsequent durchgesetzt:

- Aus der *Sicht des Dateisystems* gibt es zwei Workspace-Ordner, die nur ihre Einstellungen und Links auf die Projektdateien (aber keine Projektdateien selber) enthalten, und einen gemeinsamen Projekt-Ordner (wo sich alle Projektdateien befinden).

[1] `http://help.eclipse.org/ganymede/topic/org.eclipse.platform.doc.user/concepts/cworkset.htm`

- Die *logische Sicht* der Eclipse IDE präsentiert die Dateien so als würden diese unmittelbar im Workspace-Ordner liegen.

4.2 OSGi

Jeder Entwickler muss realisieren, dass die gesamte Entwicklung in der **OSGi**[1] Umgebung[2] stattfindet.

Was bedeutet das konkret? Jedes Eclipse *Project* (Modul) ist ein Eclipse **Plug-in** und dieses stellt automatisch ein **OSGi Bundle** dar. Die Projects, die noch nicht als OSGi Bundles angelegt sind (das erkennt man an der fehlenden `MANIFEST.MF` Datei), sollten als solche angelegt werden.

Dies vereinheitlicht und vereinfacht die Steuerung von Abhängigkeiten im gesamten Projekt. Es entfällt die Notwendigkeit, den Bezug auf andere Projekte über JAR `.classpath` herzustellen (was in Eclipse IDE meistens über den Dialog "Java Build Path" passiert). Sämtliche Abhängigkeiten werden einheitlich über das OSGi-Manifest (`MANIFEST.MF` Datei) geregelt, konkret durch die Header `Require-Bundle` und `Import-Package`.

Die Module in OSGi-Bundles umzuwandeln ist insbesondere für *Third Party Komponenten*[3] / JARs, wie JBoss oder Apache Client Libraries, wichtig. Diese Bundles können dann auch unter Versionskontrolle gestellt werden. Dies vereinfacht die Einstellungen an den Arbeitsplätzen und stellt sicher, dass die gleichen Bibliotheken von allen Entwicklern verwendet werden. Es gibt öffentliche Repositorien, die die oft verwendeten Bibliotheken in Form von OSGi Bundles zur Verfügung stellen, z.B.

- Eclipse Orbit Bundle Repository: `http://www.eclipse.org/orbit/`
- SpringSource Enterprise Bundle Repository: `http://www.springsource.com/repository/`

4.3 Ordnerstruktur

Für die single-sourced RCP/RAP Entwicklung wird die folgende physikalische Ordnerstruktur vorgeschlagen: Abb. 4.1.

Der ganze Development-Bereich ist unter dem Ordner `Dev` gekapselt. In diesem Ordner liegen auch Startup-Skripte für Eclipse und andere Tools (MySQL, JBoss, etc.)

[1] `http://www.osgi.org/`
[2] `http://www.eclipse.org/equinox/`
[3] CHRIS ANISZCZYK, Tip: Third Party Bundles. ⟨URL: `http://eclipsesource.com/blogs/2008/09/02/tip-third-party-bundles/`⟩ – Letzter Zugriff am 11.03.2009.

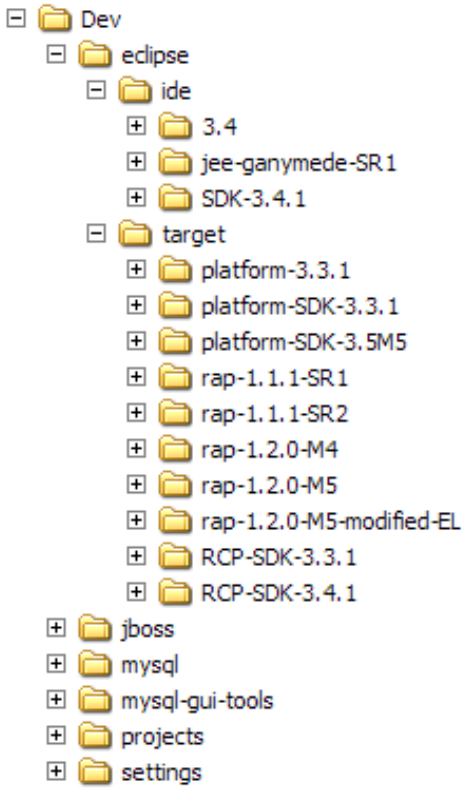


Abbildung 4.1: Gesamte Ordnerstruktur

Sämtliche Pfade werden relativ zum Ordner `Dev` angegeben. Dies ermöglicht ein unkompliziertes Verschieben des Ordners im Dateisystem oder ausrollen an einem anderen Computer.[1]

Die Struktur eines Ordners, der ein Projekt (im software-technischen Sinn) enthält, ist in Abb. 4.2 gezeigt.

Im Projekt-Ordner (`Cognidata` in der Abbildung) befinden sich folgende Unterordner:

- `rap_ws`: Workspace für RAP Entwicklung,
- `rcp_ws`: Workspace für RCP Entwicklung,
- `prj`: enthält die Eclipse Projects (== Eclipse Plug-ins == OSGi Bundles).

Bei dieser Einstellungen werden die *runtime-Workspaces* der laufenden RCP-Applikationen (`runtime-bombardier.product` in der Abbildung) bei standardmäßigen Einstellungen auch unterhalb des Projekt-Ordners erscheinen und sich nicht mit anderen Projekten vermischen.

[1] Leider sind, bedingt durch aktuelle Eclipse-Einschränkungen, nicht alle Pfade relativ einstellbar. Z.B. die Links zu Projekten aus dem Workspace werden aus unverständlichen Gründen als absolute Pfade gespeichert. Der Einsatz von *path variablen* in Eclipse lässt dies nicht umgehen. Zum einen können solche Pfadvariablen nicht für alle Pfade in Eclipse verwendet werden, zum anderen fehlt sogar eine einfache Möglichkeit, eine solche Variable per Kommandozeilen-Parameter beim Start von Eclipse zu initialisieren. Es gibt mehrere Bug-Threads zu diesem Thema im Bugtracker von Eclipse. Hoffentlich wird sich die Situation mit der Zeit verbessern, sodass wirklich *sharable* Projekte möglich sein werden.

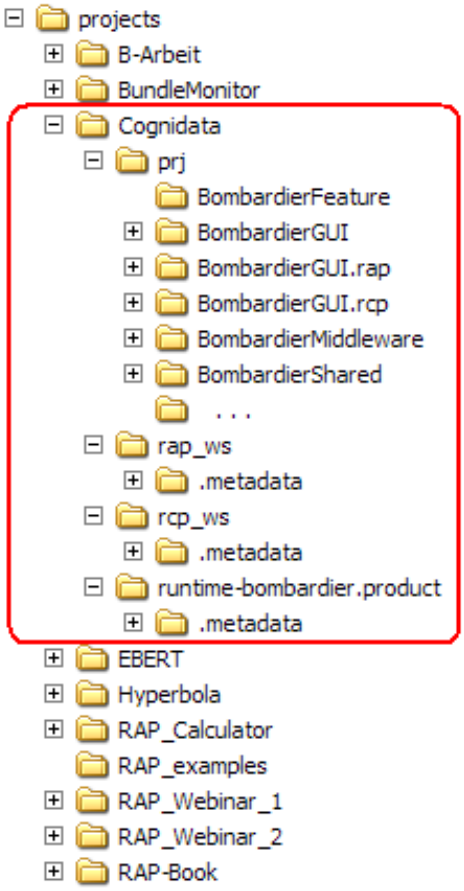


Abbildung 4.2: Struktur der Projektordner

Modifikation des alten Ordner-System

Um die Projekt-Dateien von dem Workspace-Ordner in einen anderen Ordner zu *verschieben* kann der *Move Project* Dialog (Abb. 4.3) aufgerufen werden: `File > Move`.

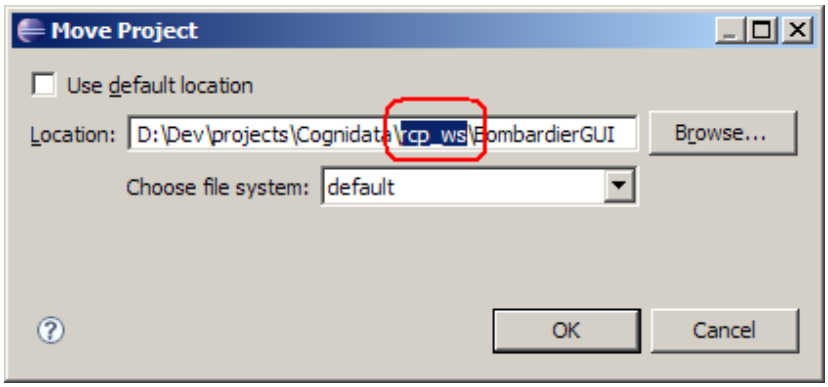


Abbildung 4.3: Move Project Dialog

Um die Projekte aus einem bestimmten Ordner mit dem Workspace zu *assoziieren* kann die Prozedur "Import existing Projects into Workspace" (`File > Import > General > Existing Projects into Workspace`) benutzt werden, wobei das Häckchen "Copy projects into workspace" leer bleiben muss.

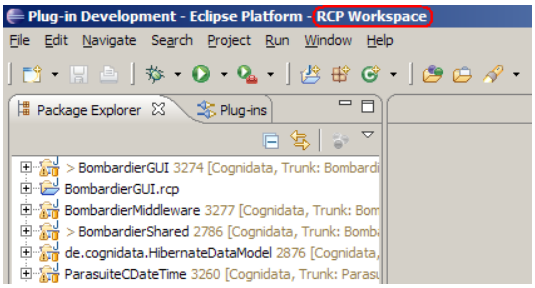
4.4 Allgemeines Konzept der Entwicklung

Es wird eine und die selbe Eclipse-Distribution zweimal gestartet – jeweils mit verschiedenen Workspace-Ordners als Parameter.

Um diesen Prozess zu automatisieren und visuell schnell unterscheiden zu können, in welchem Workspace man sich gerade befindet, werden folgende *Startup-Skripte* für Eclipse vorgeschlagen: Abb. 4.4.

Zwei Workspaces sind notwendig, da wir gegen zwei verschidene, miteinander nicht kompatible Target Platforms – RCP und RAP – entwickeln und die Einstellung der Target Platform ist in Eclipse als Workspace-bezogen realisiert.[1]

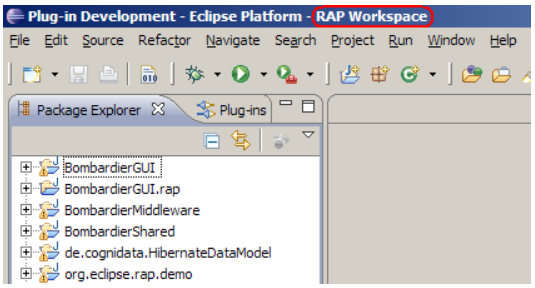
Es wird emfohlen, in den Workspace-Einstellungen das standardmäßig aktivierte "Build automatically" abzuschalten und das "Refresh automatically" einzuschalten (`Window > Preferences > General > Workspace`).




```
REM # Go to the folder containing this
    script:
cd /d "%~dp0"

set RCP_WORKSPACE=projects\Cognidata\
    rcp_ws

start eclipse\ide\jee-ganymede-SR1\
    eclipse.exe -data %RCP_WORKSPACE% -
    showlocation "RCP Workspace"
```




```
REM # Go to the folder containing this
    script:
cd /d "%~dp0"

set RAP_WORKSPACE=projects\Cognidata\
    rap_ws

start eclipse\ide\jee-ganymede-SR1\
    eclipse.exe -data %RAP_WORKSPACE% -
    showlocation "RAP Workspace"
```

Abbildung 4.4: Startup Skripte für **RCP** und **RAP** Workspaces

Wie man an den Pfaden erkennt, sind diese Skripte nicht an absolute Pfade gebunden. Lediglich die oben vorgestellte Ordnerstruktur wird vorausgesetzt.

Nach dem Starten der Skripte ist im Eclipse Fenstertitel stets zu erkennen (in der Abbildung rot umgekreist), um welchen Workspace es sich aktuell handelt.

Dieser Einsatz erwies sich als sehr hilfreich, da wegen der Ähnlichkeit der Projektliste in beiden Workspaces der aktuelle Workspace nicht schnell erkennbar war, was zu fehlerhaften Editierungen führte.

[1] Eventuell wird sich diese Situation in den zukunftigen Eclipse-Versionen verändern, so dass es möglich sein wird, die Target Platform *Project*- oder *Working Set*-bezogen und nicht Workspace-bezogen einzustellen.

4.5 *Best practices*

Execution Environment vs spezifische JRE

Es wird empfohlen ein OSGi-standardisiertes Execution Environment (wie "J2SE-1.5" oder "JavaSE-1.6") statt der spezifischen JRE (wie "Sun JDK 1.5") anzugeben.

Dies ist OSGi-konform und erleichtert sowohl die Verwendung der Applikation mit verschiedenen Java-Platform-Implementierungen, als auch die Portierung der Applikation für andere Runtimes (wie eRCP).

Um diesen Einsatz konsequent durchzusetzten muss man sämtliche Packages, die nicht standardmäßig zu einem bestimmten Execution Environment gehören sondern Java-Provider spezifisch sind (das sind z.B. alle `sun.*` Packages), dem Projekt explizit hinzufügen: als Workspace-Bundle[1] oder als zusätzliches Package in der Target Platform.

Team Project Set Dateien

Team Project Set Dateien bilden die Projects in einem Repositorium auf die Projects im Workspace ab.

Es sollen Team Project Set Dateien für beide verwendeten Workspaces zur Verfügung gestellt werden.

Eventuell auch mit weiterer Konkretisierung nach dem

- Subsystem des Gesamtprojekts, d.h. Developer-Group spezifisch (GUI, Middleware, WebServices, etc.), und nach dem
- verwendeten Betriebssystem, da dies die Betriebssystem-spezifische Bundles, wie z.B. JBoss Client/Server Bibliotheken, beeinflusst.

Target Platform

Die RCP und RAP Target Platforms sollten unter *Versionskontrolle* gestellt werden, was deren zentralisierte Anpassung – etwa das Hinzufügen von Plug-ins – ermöglicht.

Außerdem ist dies noch ein Schritt auf dem Weg zur maximalen Automatisierung der Einstellungen auf dem Arbeitsplatz des Entwicklers und zur Vorbeugung von Konfigurationsfehlern.

[1] Siehe *Third Party Komponente* im Abschnitt 4.1

5 RCP to RAP: Portierung und Single Sourcing

In diesem Kapitel wird die Reihenfolge und die allgemeinen Ansätze der Portierung einer Eclipse RCP Applikation auf die Eclipse RAP Platform unter Berücksichtigung von Single Sourcing beschrieben. Eine detaillierte Beschreibung des Portierungs-Prozesses würde den Rahmen dieser Arbeit sprengen.

5.1 Code-Analyse

Am Anfang ist eine Code-Analyse durchzuführen. Dabei müssen folgende Fragen beantwortet werden:

- Welche Teile der vorhandenen Funktionalität sind für den Web-Einsatz nicht sinnvoll oder nicht geeignet? (Das können z.B. Update-Funktionen oder die Zugriffe auf lokale Dateien sein)
- Welche zusätzliche Funktionalität soll in der Web-Applikation vorhanden sein? (z.B. Login/Passwort Authentifizierung)
- Welche Funktionalität soll in beiden Applikation vorhanden sein, aber auf verschiedene Weise gestaltet werden? (z.B. FileUpload statt FileOpen)
- Eine Web Anwendung muss Multiuser-fähig sein. Wie wird dieses Konzept im konkreten Projekt realisiert? (Sollen etwa Benutzer-Einstellungen – falls vorhanden – zwischen den Sitzungen serverseitig gespeichert werden?)

5.2 Arbeitsplatzeinrichtung

Der Arbeitsplatz des Entwicklers soll entsprechend der Empfehlungen aus Kapitel 4 eingerichtet werden.

Die zu portierenden Bundles werden in den RAP Workspace importiert. Dann wird die Target Platform in dieser Eclipse IDE Instanz auf Eclipse RAP umgeschaltet (`Window > Preferences > Plug-in Development > Target Platform`).

Die folgenden Abschnitte beschreiben die allgemeine Reihenfolge der Portierung.

5.3 UI Plug-ins Abhängigkeiten

Wie oben erwähnt, hat RAP zwar in den meisten Fällen dieselben Klassen und Packages, die in einer RCP Applikation verwendet werden. Diese befinden sich aber in Plug-ins (Bundles) mit anderen Namen. Dies führt dazu, dass die Abhängigkeiten zwischen Bundles im allgemeinen Fall nicht erfolgreich aufgelöst werden können. Das Problem tritt nicht ein, wenn in der RCP Applikation die Abhängigkeiten ausschlißliech durch `Import-Package` Manifest-Header (und nicht durch `Require-Bundle`) realisiert werden.

Typischerweise trifft man aber die Abhängigkeiten, die über den `Require-Bundle` Header konfiguriert wurden. In diesen Fällen können die vorhandene Abhängigkeiten zum `org.eclipse.ui` Bundle als *optional* eingestellt und die zusätzlichen, ebenfalls optionalen Abhängigkeiten zu `org.eclipse.rap.ui` hinzugefügt werden.

Diese Operation sollte die meisten Kompilierungsfehler beseitigen.

5.4 Extension Points

Die in RAP nicht implementierten *Extension Points*, wie `org.eclipse.ui.bindings`, sollen für den RAP Workspace unsichtbar gemacht werden.

Der aktuelle Ansatz zur Lösung dieses Problems liegt in der Verwendung von *Plug-in Fragments*[1]. Ein soches Fragment bezieht sich auf sein Host-Plugin und fügt ihm zusätzliche Ressourcen zu: Klassen, Packages, Manifest-Einträge.

In jedem Workspace (RCP und RAP) kann jeweils ein Plugin-Fragment definiert werden, das die Extension Points samt implementierender Klassen aufnimmt, die nicht in dem anderen Workspace vorhanden sind. Das Übertragen der Extension Points erfolgt durch das einfache Ausschneiden der entsprechenden Textfragmente aus der Datei `plugin.xml` und das Einfügen in die Datei `fragment.xml`.

5.5 API Unterschiede

Manche Klassen, Methoden oder Attribute sind (noch) nicht in RAP implementiert. Ein typisches Beispiel ist die folgende Fehlermeldung:

```
ActionFactory.ABOUT cannot be resolved in ApplicationAction-
BarAdvisor.java
```

Der allgemeine Ansatz in diesem und ähnlichen Fällen ist das *Kapseln der inkompatiblen Funktionalität in einem Facade-Package/Plugin*, das die allgemeinen Methodenaufrufe enthält (z.B. `createAboutAction` für den Fall oben) und dann diese Aufrufe an die plattformspezifischen[2] Bundles weiterleitet.

[1] `http://wiki.eclipse.org/FAQ_What_is_a_plug-in_fragment%3F`
[2] D.h. nur im RAP- oder im RCP-Workspace vorhandenen.

Das Auffinden des Bundles, das die Aufrufe plattformspezifisch implementiert, erfolgt dann per *Reflection*, wozu eine bestimmte Namenskonvention notwendig ist.

5.6 Entrypoint

Um eine lauffähige RAP Applikation zu erhalten ist noch das Hinzufügen einer *Entrypoint Extension* notwendig, welche dem `IApplication`-Interface einer RCP Applikation entspricht.

6 Ausblick und Zusammenfassung

6.1 Zukunft von RAP

Die Investitionen eines Unternehmens müssen zukunftsicher sein. Bei jeder verwendeten Technologie stellt sich deshalb die Frage, wie zukunftsicher sie ist? Inwieweit es sich lohnt, in diese Technologie zu investieren? Wie groß ist die Wahrscheinlichkeit, dass die Technologie nicht langfristig unterstützt wird? Antworten auf diese Fragen gibt die nachfolgende Übersicht, ausgehend von zwei Standpunkten: dem organisatorischen und dem technischen Standpunkt .

Organisatorische Sicht

Die Firma-Innovator von RAP – Innoopract GmbH[1], Karlsruhe / Portland – ist ein *strategisches Mitglied*[2] der *Eclipse Foundation*[3].

Der Geschäftsführer von Innoopract Jochen Krause gehört zum *Board of Directors*[4] der Eclipse Foundation.

Am 01.12.2008 haben die Innoopract GmbH und die kanadische Firma *Code 9 Corporation*[5] die Gründung von *EclipseSource*[6] bekannt gegeben.[7] Jeff McAffer von Code 9, der die technische Leitung bei EclipseSource übernimmt, ist der Autor[8] des *RCP-Referenzbuchs*[9].

Die EclipseSource stellt somit einen Zusammenschluss der Leiter und anerkannten Experten von mehr als zehn Schlüsselprojekten der Eclipse Foundation dar, darunter RCP, Equinox und RAP.

[1] `http://www.innoopract.com/`
[2] `http://www.eclipse.org/membership/become_a_member/membershipTypes.php#strategic`
[3] `http://www.eclipse.org/org/foundation/`
[4] `http://www.eclipse.org/org/foundation/directors.php`
[5] `http://code9.com/`
[6] `http://eclipsesource.com/`
[7] Pressemitteilungen bei Innoopract: `http://www.innoopract.com/de/unternehmen/pressemeldungen/de/code-9-und-innoopract-gruenden-eclipsesource/` und bei Heise.de: `http://www.heise.de/developer/news/meldung/print/119775`.
[8] zusammen mit Jean-Michel Lemieux
[9] JEFF MCAFFER/JEAN-MICHEL LEMIEUX, Eclipse Rich Client Platform: Designing, Coding, and Packaging Java™ Applications. Addison-Wesley Professional, Oct 2005, ISBN 0–321–33461–2.

Die gleichen Personen und Organisationen spielen auch die Schlüsselrollen bei dem jungen Projekt *Eclipse RT*[1] (Eclipse Runtime), das die Runtime-bezogenen Projekte (wie RCP und RAP) koordinieren wird, und bei der Entwicklung der nächsten Version von Eclipse – *Eclipse 4* (Codename *e4*).

Bei dieser Konstellation ist die Zukunft von RAP organisatorisch und so zu sagen, politisch, mehr als gesichert.

Technische Sicht

Das RAP API-Set wird sich erweitern (soweit das für die Benutzung im Browser notwendig und angemessen ist) und sich an den Umfang des RCP API-Sets annähern. Sogar die API Teile, die zunächst nicht als webfähig galten wie z.B. der `Graphics Context`, können realisiert werden.

In den künftigen Eclipse-Versionen werden Veränderungen an der RCP-API stattfinden, die den Einsatz der Eclipse Platform für Webanwendungen einfacher machen werden. So ist für Eclipse 4 als wichtige Neuerung geplant, die *Multi-User-Fähigkeit* der Eclipse-Platform einzuführen, was für den Webeinsatz absolut notwendig ist. Weiterhin wird *CSS-Styling* von Oberflächen sowie für Web (RAP) als auch für Desktop (RCP) Applikationen möglich sein (Abb. 6.1[2]). Das SWT-Team experimentiert auch mit einem weiteren Ansatz: *Cross-Compiling*. Dabei kann der Java-Code der Widgets bzw. der Applikation in ActionScript übersetzt werden, wodurch mehr Funktionalitäten direkt im Browser laufen können. Auch die Übersetzung in JavaScript wie bei GWT soll möglich werden.[3]

Auf diese Weise soll es erleichtert werden, auch komplexere Applikationen so zu entwickeln, dass sie ohne Veränderungen sofort sowohl auf der RCP- als auch auf der RAP-Platform einsatzfähig sind.

Und die Zukunft heißt: OSGi. Das Eclipse Runtime Project wird dafür sorgen, dass alle Eclipse Runtime Technologien koordiniert und den OSGi Grundsätzen entsprechend entwickelt werden.

Was die Akzeptanz und praktischen Erfolge der RAP Technologie betrifft ist es interessant zu wissen, dass zwei der insgesamt drei Finalisten der *Eclipse Community Awards 2009*[4] in der Kategorie "Best Commercial Equinox Applications" auf Eclipse RAP basieren.[5]

Die besten Spezialisten der Eclipse Welt arbeiten auf die eine oder andere Weise an RAP mit. Dadurch sind die besten Voraussetzungen für die technische Entwicklung der RAP Technologie gegeben.

---

[1] `http://www.eclipse.org/rt/`

[2] Quelle: CHRIS ANISZCZYK, The first e4 milestone. ⟨URL: `http://eclipsesource.com/blogs/2009/02/10/the-first-e4-milestone/`⟩ – Letzter Zugriff am 11.03.2009

[3] Eine Preview auf Eclipse 4.0. ⟨URL: `http://entwickler.com/itr/news/psecom,id,42320,nodeid,82.html`⟩ – Letzter Zugriff am 06.03.2009.

[4] `http://feeds.feedburner.com/~r/eclipse/fnews/~3/548955748/20090304_AwardsFinalists.php`

[5] RAP in the finals. ⟨URL: `http://eclipsesource.com/blogs/2009/03/10/rap-in-the-finals/`⟩ – Letzter Zugriff am 11.03.2009.

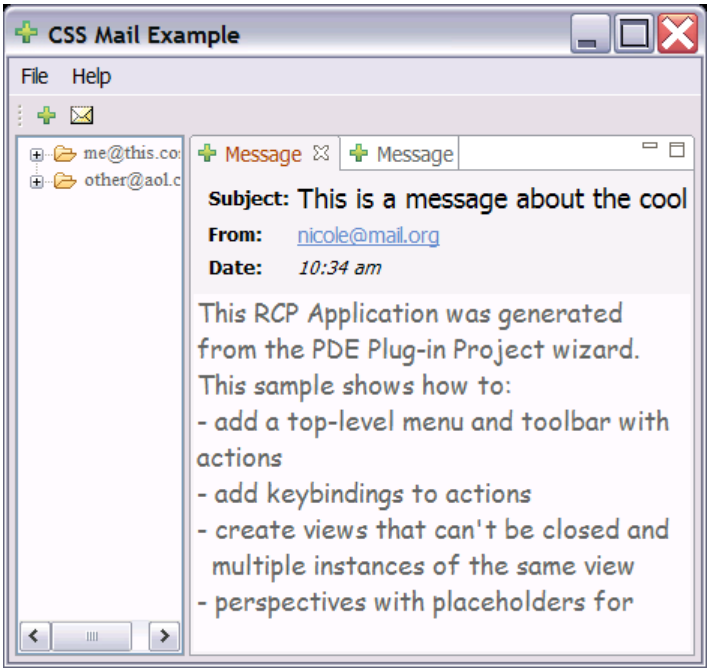


Abbildung 6.1: RCP Mail CSS Example

6.2 Auswirkung dieser Arbeit auf das Projekt

Die durchgeführten Untersuchungen wurden von Projektleitern als vielversprechend bezeichnet.

Die Entscheidung, das Web-Interface der Anwendung mit Hilfe von Eclipse RAP zu implementieren, wurde für richtig befunden.

Die weitere Entwicklung der Web- und Desktop-Interfaces im Projekt wird im Kontext der durch diese Arbeit gezeigten Richtung durchgeführt.

6.3 Zusammenfassung

Diese Arbeit war eine Herausforderung für den Autor.

Zum einen waren die Kentnisse und Erfahrungen in Bezug auf Eclipse RCP zu Beginn der Arbeit nicht ausreichend. Die gründliche Einarbeitung in diese Bereiche bedarf einer soliden Zeitinvestition. Die Menge der Fragen umfasste die Eclipse Workbench / JFace / SWT Konzepte sowie die OSGi Spezifikation und Eqinox als OSGi-Implementierung.

Zum anderen – bedingt durch das relativ geringe "Alter" der Eclipse RAP Technologie — waren nicht viele Erfahrungen in der Entwickler-Community vorhanden. Die *meisten* für diese Arbeit benutzten Quellen wurden erst nach dem Beginn der Mitarbeit am Projekt veröffentlicht. Und auch innerhalb dieser Zeit entwickelte sich die RAP Technologie sehr dynamisch, sodass die Konzepte und best practices des

Single Sourcing sich in dieser Zeit geändert haben[1]. Mehrere Webinare[2] zum Thema RAP-RCP Single Sourcing und die Analyse der Fragestellungen im RAP-Forum[3] zeigen, dass noch vieles in diesem Bereich nicht ganz klar ist, sogar für erfahrene Entwickler.

Ebenso relativ neu ist auch die OSGi Thematik. Wenn es z.B. um praktische Anwendung der Ansätze zur Steuerung der Bundle-Abhängigkeiten[4] (`Require-Bundele` vs `Import-Package` Header), Plug-in Fragments oder *split packages*[5] geht, ist es manchmal schwer, sachlich richtige Empfehlungen im Internet zu finden.

Unabhängig von (oder dank?) dieser Schwierigkeiten war die Mitarbeit am Projekt sehr interessant und die gestellten Ziele wurden erreicht. Die erworbenen Kentnisse in den Eclipse RCP/RAP- und OSGi-Bereichen sind hochaktuell und lassen sich sehr gut für jede Anwändungsdomäne verwenden.

[1] Z.B. die Benutzung von *Plug-in Fragments* wurde erst in der letzten Zeit von den RAP Entwicklern als best practice empfohlen (statt der "selbständigen" platformspezifischen Bundles).

[2] RÜDIGER HERRMANN, RAP Webinar: Single-Sourcing Techniken für RAP und RCP. ⟨URL: `http://eclipsesource.com/de/resources/webcasts/`⟩ – Letzter Zugriff am 09.03.2009, FRANK APPEL, RAP Webinar: Single-Sourcing Techniques for RAP and RCP. ⟨URL: `http://eclipsesource.com/en/resources/webcasts/`⟩ – Letzter Zugriff am 09.03.2009, JOCHEN KRAUSE, RAP Webinar: Single-Sourcing Techniques for RAP and RCP. ⟨URL: `http://eclipsesource.com/en/resources/webcasts/`⟩ – Letzter Zugriff am 09.03.2009

[3] Eclipse NewsPortal - eclipse.technology.rap. ⟨URL: `http://www.eclipse.org/newsportal/thread.php?group=eclipse.technology.rap`⟩ – Letzter Zugriff am 17.04.2009.

[4] [require-bundle vs import-package]: `http://mail-archives.apache.org/mod_mbox/felix-dev/200610.mbox/%3C4523BBED.4040408@ungoverned.org%3E`, Best practices for dependency management: `http://www.eclipsezone.com/eclipse/forums/t98465.html`

[5] CHRIS ANISZCZYK, Tip: Split Packages and Visibility. ⟨URL: `http://eclipsesource.com/blogs/2008/08/22/tip-split-packages-and-visibility/`⟩ – Letzter Zugriff am 17.04.2009.

Abbildungsverzeichnis

Literaturverzeichnis

Appendix E. Useful OSGi tips > E.1. *OSGi Fragments*. ⟨URL: http://static.springframework.org/osgi/docs/1.1.2/reference/html/appendix-tips.html⟩ – Letzter Zugriff am 10.03.2009.

Eclipse NewsPortal - eclipse.technology.rap. ⟨URL: http://www.eclipse.org/newsportal/thread.php?group=eclipse.technology.rap⟩ – Letzter Zugriff am 17.04.2009.

Eclipse RAP Home @ eclipse.org. ⟨URL: http://www.eclipse.org/rap⟩ – Letzter Zugriff am 28.02.2009.

Eclipse RAP Home @ innoopract.com. ⟨URL: http://www.innoopract.com/de/eclipse-rap/⟩ – Letzter Zugriff am 09.03.2009.

Eclipse RCP Home @ eclipse.org. ⟨URL: http://www.eclipse.org/rcp⟩ – Letzter Zugriff am 28.02.2009.

EngineeringAdvice. ⟨URL: http://code.google.com/p/google-depan/wiki/EngineeringAdvice⟩ – Letzter Zugriff am 10.03.2009.

INNOOPRACT – Technologien und Dienstleistungen für Eclipse. ⟨URL: http://www.innoopract.com/de/⟩ – Letzter Zugriff am 08.03.2009.

Managing Plugins in Eclipse. ⟨URL: http://blog.exis.com/colin/archives/2004/12/23/managing-plugins-in-eclipse/⟩ – Letzter Zugriff am 10.03.2009.

RAP Developer Guide. ⟨URL: http://help.eclipse.org/ganymede/nav/33⟩ – Letzter Zugriff am 04.03.2009.

RAP FAQ. ⟨URL: http://wiki.eclipse.org/RAP/FAQ⟩ – Letzter Zugriff am 04.03.2009.

ServerPushFAQ. ⟨URL: http://code.google.com/p/google-web-toolkit-incubator/wiki/ServerPushFAQ⟩ – Letzter Zugriff am 16.04.2009.

Eine Preview auf Eclipse 4.0. ⟨URL: http://entwickler.com/itr/news/psecom,id,42320,nodeid,82.html⟩ – Letzter Zugriff am 06.03.2009.

Im Gespräch: Jochen Krause, Geschäftsführer der Innoopract Informationssysteme GmbH

RAP in the finals. ⟨URL: http://eclipsesource.com/blogs/2009/03/10/rap-in-the-finals/⟩ – Letzter Zugriff am 11.03.2009.

Aniszczyk, Chris: The first e4 milestone. ⟨URL: http://eclipsesource.com/blogs/2009/02/10/the-first-e4-milestone/⟩ – Letzter Zugriff am 11.03.2009.

Aniszczyk, Chris: Tip: Split Packages and Visibility. ⟨URL: http://eclipsesource.com/blogs/2008/08/22/tip-split-packages-and-visibility/⟩ – Letzter Zugriff am 17.04.2009.

Aniszczyk, Chris: Tip: Third Party Bundles. ⟨URL: http://eclipsesource.com/blogs/2008/09/02/tip-third-party-bundles/⟩ – Letzter Zugriff am 11.03.2009.

Aniszczyk, Chris/Crosby, Phil: Manage your Eclipse environment. ⟨URL: http://www.ibm.com/developerworks/library/os-ecl-manage/⟩ – Letzter Zugriff am 10.03.2009.

Aniszczyk, Chris/Muskalla, Benjamin: Rich Ajax Platform, Part 1: An introduction. ⟨URL: http://www.ibm.com/developerworks/opensource/library/os-eclipse-richajax1/⟩ – Letzter Zugriff am 09.03.2009.

Aniszczyk, Chris/Muskalla, Benjamin: Rich Ajax Platform, Part 2: Developing applications. ⟨URL: http://www.ibm.com/developerworks/opensource/library/os-eclipse-richajax2/⟩ – Letzter Zugriff am 09.03.2009.

Appel, Frank: eclipse rich ajax platform (RAP). ⟨URL: http://www.nabble.com/attachment/13278068/0/RAP_Longtalk.pdf⟩ – Letzter Zugriff am 16.04.2009.

Appel, Frank: RAP Webinar: Single-Sourcing Techniques for RAP and RCP. ⟨URL: http://eclipsesource.com/en/resources/webcasts/⟩ – Letzter Zugriff am 09.03.2009.

Presentation Slides .pdf
RAP code examples .zip
Webinar audio with presentation slides – .wmv, .mp4

Appel, Frank/Krause, Jochen: RAP the Web. ⟨URL: http://www.eclipsecon.org/2008/index.php?page=sub/&id=199⟩ – Letzter Zugriff am 09.03.2009.

Präsentation im PDF-Format.

Beaton, Wayne: Eclipse Business Expenses Reporting Tool (EBERT). ⟨URL: http://www.eclipse.org/examples/example.php?id=expenses⟩ – Letzter Zugriff am 10.03.2009.

Brüstel, Jonas: Evaluierung der Rich Ajax Platform (RAP) anhand einer Beispielanwendung. ⟨URL: http://opus.kobv.de/fhbrb/volltexte/2008/51/pdf/jonas_bruestel_diplomarbeit_compression.pdf⟩ – Letzter Zugriff am 26.03.2009.

Diplomarbeit

Herrmann, Rüdiger: RAP Webinar: Single-Sourcing Techniken für RAP und RCP. ⟨URL: http://eclipsesource.com/de/resources/webcasts/⟩ – Letzter Zugriff am 09.03.2009.

Presentation Slides .pdf
RAP code examples .zip
Webinar audio with presentation slides – .wmv, .mp4

Krause, Jochen: Rich Ajax Platform (RAP) - Web 2.0 the Eclipse Way. ⟨URL: http://www.java-forum-stuttgart.de/jfs/2006/abstracts.html⟩ – Letzter Zugriff am 09.03.2009.

Präsentation im PDF-Format.

Krause, Jochen: RAP Webinar: Single-Sourcing Techniques for RAP and RCP. ⟨URL: http://eclipsesource.com/en/resources/webcasts/⟩ – Letzter Zugriff am 09.03.2009.

Krishna, Suresh/Fenstermaker, Trebor: Using Eclipse Ganymede to develop for the desktop, Web and mobile devices, Part 1: Developing for the Rich Client Platform, the Ganymede way. ⟨URL: http://www.ibm.com/developerworks/edu/os-dw-os-eclipse-ganymede-pt1.html⟩ – Letzter Zugriff am 03.03.2009.

Untertitel: *Desktop applications with RCP, Subversion and p2.*
Eine kostenlose Registrierung auf der Website notwendig.

Krishna, Suresh/Fenstermaker, Trebor: Using Eclipse Ganymede to develop for the desktop, Web and mobile devices, Part 2: Developing for the Rich Ajax Platform, the Ganymede way. ⟨URL: http://www.ibm.com/developerworks/edu/os-dw-os-eclipse-ganymede-pt2.html⟩ – Letzter Zugriff am 03.03.2009.

Untertitel: *RAP: The new way to the Web.* (Aktuell ist ein Fehler auf der Website in der Überschrift vorhanden. Hier wurde die gedachte Bezeichnung angegeben.)
Eine kostenlose Registrierung auf der Website notwendig.

Krishna, Suresh/Fenstermaker, Trebor/Nehrer, Peter: Using Eclipse Ganymede to develop for the desktop, Web and mobile devices, Part 3: Developing for the Embedded Rich Client Platform, the Ganymede way. ⟨URL: http://www.ibm.com/developerworks/edu/os-dw-os-eclipse-ganymede-pt3.html⟩ – Letzter Zugriff am 03.03.2009.

Untertitel: *eRCP: Going mobile.*
Eine kostenlose Registrierung auf der Website notwendig.

Lange, Fabian: Eclipse Rich Ajax Platform. Bringing Rich Client into the Web. Apress, 31 Dec 2008.

Companion Website: http://www.rap-book.com

McAffer, Jeff: e4 and RAP. ⟨URL: http://eclipsesource.com/blogs/2009/02/20/e4-and-rap/⟩ – Letzter Zugriff am 11.03.2009.

McAffer, Jeff/Lemieux, Jean-Michel: Eclipse Rich Client Platform: Designing, Coding, and Packaging Java™ Applications. Addison-Wesley Professional, Oct 2005, ISBN 0–321–33461–2.

Companion Website: http://eclipsercp.org

Muskalla, Benjamin/Sternberg, Ralf: RCP goes Web 2.0. ⟨URL: http://www.innoopract.com/fileadmin/user_upload/Dokumente/Web-enabled_RCP_Applications_with_the_Rich_Ajax_Platform_pdf.pdf⟩ – Letzter Zugriff am 03.04.2009.

Based on an article by Benjamin Muskalla and Ralf Sternberg in Eclipse-Magazin Vol. 12; translation by Innoopract Inc

OSGi Alliance: OSGi Service Platform. Core Specification. Release 4, Version 4.1. April 2007. ⟨URL: http://www.osgi.org/Download/Release4V41⟩ – Letzter Zugriff am 04.03.2009.

Datei r4.core.pdf (2.063.064 bytes)
Eine kostenlose Registrierung auf der Website notwendig.

Russell, Mark/Rubel, Daniel: Getting started with RAP development. ⟨URL: http://www.eclipsecon.org/2008/index.php?page=sub/&id=215⟩ – Letzter Zugriff am 16.04.2009.

Präsentation im PDF-Format.

Schafer, Steffen: Einsatzmoglichkeiten der Eclipse RAP. ⟨URL: http://www.oio.de/public/opensource/eclipse-rap/tutorial-eclipse-rich-application-plattform-portierung.htm⟩ – Letzter Zugriff am 09.03.2009.

Auch im PDF-Format vorhanden.

Staudacher, Holger: Extended Presentation API. ⟨URL: http://eclipsesource.com/blogs/2008/12/18/extended-presentation-api/⟩ – Letzter Zugriff am 11.03.2009.

Sternberg, Ralf/Herrmann, Rüdiger: Getting started with RAP development. ⟨URL: http://www.eclipsecon.org/2008/index.php?page=sub/&id=199⟩ – Letzter Zugriff am 09.03.2009.

Präsentation im PDF-Format.

Sternberg, Ralf/Muskalla, Benjamin: Single Sourcing mit der RAP. Java-Magazin 2009, Nr. 2.

Volanakis, Elias: Java-based web-apps with the Rich Ajax Platform (RAP). ⟨URL: http://www.pjug.org/docs/RAP.pdf⟩ – Letzter Zugriff am 05.04.2009.

Volanakis, Elias: Would you believe this is a browser based Application? ⟨URL: http://rapblog.innoopract.com/2008/03/would-you-believe-this-is-browser-based.html⟩ – Letzter Zugriff am 12.03.2009.

Volanakis, Elias/Lopez, Jordi Boehme: From RCP to RCP/RAP – Conversion and Single-Sourcing Techniques. ⟨URL: http://www.eclipsecon.org/2008/index.php?page=sub/&id=233⟩ – Letzter Zugriff am 09.03.2009.

Präsentation im PDF-Format.

Wickesser, Craig: Case study: Eclipse Rich Ajax Platform Use at CAS Software AG. ⟨URL: http://www.infoq.com/articles/eclipse-rap-casestudy⟩ – Letzter Zugriff am 08.03.2009.

Wittemann, Martin: Professionelle Entwickler-Tools für das JavaScript-Framework qooxdoo. ⟨URL: http://www.stz-ida.de/download/forschung/ba/mwittemann.pdf⟩ – Letzter Zugriff am 05.04.2009.

Bachelor Thesis

Woelker, Manuel: The new Eclipse download wizard and RAP performance. ⟨URL: http://eclipsesource.com/blogs/2008/09/17/the-new-eclipse-download-wizard-and-rap-performance/⟩ – Letzter Zugriff am 16.04.2009.

Danksagung

In erster Linie danke ich meinem Projektleiter Sebastian Süß und dem Korreferenten Prof. Dr. Th. Letschert für die Möglichkeit an diesem Projekt teilzunehmen und für die Unterstützung in allen Fragen!

Björn Kasteleiner danke ich für die LaTeX-Vorlage, auf deren Basis diese Arbeit erstellt wurde!

Nils Braden danke ich für das Korrekturlesen und für viele wertvolle Anmerkungen!

Der freundlichen Eclipse-Community danke ich für die Hilfe bei der Klärung der Fragen und bei der Suche nach Problemlösungen, insbesondere Elias Volanakis und Wayne Beaton für die produktiven Diskussionen über RAP/RCP Single Sourcing!

Meiner Tochter Dana danke ich für das allererste Korrekturlesen, insbesondere für den Vorschlag, *"Love-Level"* statt "Low-Level" zu schreiben!

Meiner schönen Frau Oksana danke ich nicht nur für die Motivation für diese Arbeit und für das Informatik-Studium, sondern für den Sinn des Lebens überhaupt!