



LN8 EE

Aufsetzen einer Lotus Notes 8.5.1 Entwicklungsumgebung

Autor: Mario Gutsche

Veröffentlicht: 25.02.2010

<http://www.mario-gutsche.de>

Kurzfassung

Dieses Dokument beschreibt die Einrichtung einer Arbeitsumgebung zur Entwicklung von Lotus Notes 8.5.1 Anwendungen sowie Plugins.

Als Server wird **Domino 8.5.1** in einer virtuellen **Ubuntu 8.0.4** Umgebung installiert. Die Entwicklung normaler Datenbanken geschieht über den **Domino Designer 8.5.1** und Plugins werden in **Eclipse 3.4.2** mit dem **Expeditor Toolkit 6.2.1** entwickelt.

Inhaltsverzeichnis

Kurzfassung (Abstract)

Abbildungsverzeichnis **II**

Tabellenverzeichnis **III**

1 Ubuntu 8.0.4 Server **1**

1.1 Ubuntu Netzwerkinstallation 2

2 Lotus Domino 8.5.1 **5**

3 Softwareversionen **8**

4 Lotus Notes, Designer, Administrator 8.5.1 **9**

5 Eclipse 3.4.2 **10**

6 Lotus Expeditor Toolkit 6.2.1 **12**

7 Testplugin **14**

7.1 Grundlagen 14

7.2 Implementierung 15

7.3 API-Tools 21

Literaturverzeichnis **22**

Anlagenverzeichnis **i**

Abbildungsverzeichnis

- 5.1 Auswahl des Workspace 10
- 6.1 Expeditor Start-Konfiguration 13
- 7.1 Extension Points der Sidebar 14
- 7.2 Plug-In Wizard 1 16
- 7.3 Plug-In Wizard 2 18
- 7.4 MANIFEST Editor 19



Tabellenverzeichnis

3.1 Versionsmapping 8



1 Ubuntu 8.0.4 Server

Als Betriebssystem für den Domino Server 8.5.1 wird ein Ubuntu Server der Version 8.0.4 gewählt, da dieser bis 2013 weiter gepflegt wird und somit auch zukunftssicher im Live-Betrieb genutzt werden kann. Um keine neue Festplattenpartition für Ubuntu einrichten zu müssen wird Ubuntu in eine Virtuelle Maschine installiert. Dazu laden wir Ubuntu JeOs 8.0.4¹ herunter, welches speziell für den Einsatz in virtuellen Umgebungen optimiert wurde und lediglich 100 MB groß ist. Das Ubuntu Image wird dann, in einer mit folgenden Eigenschaften erstellten virtuellen Umgebung, gestartet:

- Configuration: Custom
- Guest Operating System: Linux, des weiteren wird Ubuntu ausgewählt
- Processors: Two, um den zur Verfügung stehenden Entwicklungsrechner optimal zu nutzen
- Memory: 500 MB
- Network Connection: Use bridged networking
- Disk: Create a new virtual disk, capacity wird auf 4 GB gesetzt, alles wird sofort allokiert um die Zugriffe zu beschleunigen

Sollte lediglich VMWare-Player zur Verfügung stehen, kann die virtuelle Umgebung erstellt werden, in dem man sich an die Anleitung² der Internetseite „Forever For Now“ hält. Im Installationsmenü von Ubuntu wird als Rechnername *ubuntu* und als Partitionsmethode *Geführt - verwende vollständige Festplatte* gewählt und eine neuer Nutzer mit dem Namen *user* und dem Passwort *user* angelegt. Nachdem die Installation fertig gestellt und das System neu gestartet wurde kann man sich als *user* einloggen und

```
sudo apt-get update && sudo apt-get upgrade
```

ausführen um evtl. nötige Updates zu installieren. Daraufhin installieren wir den Openssh Server um über die Windows-Software Putty³ per SSH auf den Server zugreifen und mit Filezilla⁴ Dateien transferieren zu können zu können.

¹<http://cdimage.ubuntu.com/jeos/releases/>

²<http://www.ffnn.nl/pages/articles/linux/vmware-player-image-creation.php>

³<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

⁴<http://filezilla-project.org/>

```
sudo apt-get install openssh-server
```

Daraufhin stehen folgende Kommandos zur Verfügung:

- Den SSH-Server starten:

```
sudo /etc/init.d/ssh start
```
- Den SSH-Server neu starten:

```
sudo /etc/init.d/ssh restart
```
- Den SSH-Server stoppen:

```
sudo /etc/init.d/ssh stop
```
- SSH-Verbindung zum Server herstellen (alternativ zu Putty):

```
ssh user@your-server-ip-address
```

Ein manuelles Starten des SSH-Server sollte nach einem Neustart nicht nötig sein, da die Installation einen Autostart einrichtet.

1.1 Ubuntu Netzwerkinstallation

Standardmäßig wird das Ubuntu Netzwerk per DHCP konfiguriert, was solange keine Probleme bereitet wie der Entwicklungsrechner Zugriff auf einen DHCP Server hat, der die IP-Adressen automatisch zuweist. Sollte kein DHCP-Server zur Verfügung stehen, muss das Netzwerk manuell konfiguriert werden, damit per SSH und SFTP von Windows aus auf Ubuntu zugegriffen werden kann. Auch wenn ein DHCP-Server erreichbar ist, ist die manuelle Konfiguration praktisch, da so nicht jedes mal die aktuelle IP-Adresse von Ubuntu angepasst werden muss, falls der DHCP-Server eine andere IP zugewiesen hat und man sich per SSH oder SFTP verbinden möchte. Um das Netzwerk zu konfigurieren wird der bereits installierte Editor Vim genutzt:

```
sudo vim /etc/network/interfaces
```

Vim hat 2 Modi, den Editiermodus, um Text einzugeben und den Kommando-modus für die weitere Bedienung. Da die Steuerung von VIM recht ungewöhnlich ist⁵ sind hier wichtigsten Tastenkombinationen für die Bedienung aufgelistet:

⁵http://www.linuxconfig.org/Vim_Tutorial

- Eingabe vor dem aktuellen Zeichen: *i*
- Eingabe nach dem aktuellen Zeichen: *a*
- Zeichen Löschen: *x*
- kommandomodus starten: *Esc*
- Rückgängig machen: *u* oder *:u*
- Wiederherstellen: *:redo*
- Beenden ohne zu Speichern: *:q!*
- Speichern und Beenden: *:wq*

In der geöffneten Datei wird durch voranstellen eines # die folgende Zeile deaktiviert:

```
iface eth0 inet dhcp
```

Darunter kann nun die Konfiguration der Netzwerkverbindung vorgenommen werden:

```
#iface eth0 inet dhcp  
iface eth0 inet static  
address 192.168.0.111  
netmask 255.255.255.0  
gateway 192.168.0.1
```

Für den Fall das die Virtuelle Maschine ihrer Netzwerkkarte nicht eth0 sondern eth1 zuweist wird die Datei noch um folgende Zeilen ergänzt, die sicherstellen das ihr Dominoserver immer unter der IP *192.168.0.111* erreichbar ist:

```
auto eth1  
iface eth1 inet static  
address 192.168.0.111  
netmask 255.255.255.0  
gateway 192.168.0.1
```

Um die Änderungen zu aktivieren wird das Netzwerk neu gestartet.

```
sudo /etc/init.d/networking restart
```

Sollte Ubuntu nun immer noch nicht von Windows aus mit Putty erreichbar sein muss in den Netzwerkeinstellungen der Virtuellen Umgebung folgendes eingestellt werden: *Network Connection = Custom: Specific virtual network, VMnet8(NAT)* und in den Windows-Netzwerkeinstellungen von VMnet8 muss eine IP im selben Bereich zugeteilt werden wie der Ubuntu-Server (z.B. 192.168.0.100), der VMware Bridge Adapter sollte aktiviert werden.

2 Lotus Domino 8.5.1

Nachdem Ubuntu Installiert ist wird das Domino 8.5.1 Installationsarchiv unter Verwendung von Filezilla in das Home-Verzeichnis des Users *user* kopiert, dann verbinden wir uns von Windows aus mit Putty zu dem Server und erzeugen mit

```
sudo mkdir /Dominoinstall
```

ein Verzeichniss, in das das Domino 8.5.1 Installationsarchiv mit folgendem Befehl extrahiert und danach gelöscht wird:

```
sudo tar -xvf lotus_domino851_xlinux_CZ5RWEN.tar -C /Dominoinstall  
rm lotus_domino851_xlinux_CZ5RWEN.tar
```

Vor der Installation muss eine Gruppe *notes* erstellt werden:

```
sudo groupadd notes
```

Das erfolgreiche Erstellen der Gruppe kann überprüft werden, in dem das Ende von */etc/group* überprüft wird:

```
sudo less /etc/group
```

Nun wird ein Benutzer *notes*, der Mitglied dieser Gruppe ist erstellt, das Home Verzeichnis erzeugt und die Shell auf */bin/bash* gesetzt, da es ansonsten zu Problemen zwischen Domino mit der Standard-Shell *Dash* von Ubuntu kommen kann:

```
sudo useradd -g notes -s /bin/bash -d /home/notes -m notes
```

Als Passwort für den Benutzer wird *notes* gewählt:

```
sudo passwd notes
```

Jetzt erstellen wir die Ordnerstruktur für das Notes Data-Verzeichnis:

```
sudo mkdir /local  
sudo mkdir /local/notesdata
```

Nachdem alles vorbereitet ist kann Domino installiert werden:

```
sudo su
cd /Dominoinstall/linux/domino/
./install
```

Die Frage ob im Konsolenmodus installiert werden soll wird mit *Yes* bestätigt, für den Testserver beantworten wir die Frage nach partitionierten Servern mit *0* bzw. später mit *No* und als Installationsverzeichnis und Data-Verzeichnis wird */opt/ibm/lotus* bzw. */local/notesdata* übernommen. Als Benutzer und Gruppe wird *notes* eingetragen und Server Setup wird *Remote* gewählt, dadurch startet Domino im Listen-Modus und kann von Windows aus über die *Domino remote setup console*, die zusammen mit dem Domino Administrator installiert werden kann, konfiguriert werden. Als Installationstyp wird der *Domino Enterprise Server* (988 MB) gewählt. Nachdem die Installation fertig gestellt ist befindet sich Domino im Listenmodus und kann über Port 8585 von außen konfiguriert werden. Sollte die Konfiguration jetzt nicht durchgeführt werden, kann Domino später jederzeit im Listen-Modus gestartet werden in dem man in das *notesdata*-Verzeichnis wechselt und daraufhin den Server mit dem Parameter *-listen* startet.

```
cd /local/notesdata/
/opt/ibm/lotus/bin/server -listen
```

Um Domino nun von Windows aus einrichten zu können muss bei der Installation vom Domino Administrator explizit das *Remote Server Setup* ausgewählt werden, da nur dann die *serversetup.exe* im Notes-Verzeichnis installiert wird, welche mit dem Parameter *-remote* gestartet werden muss:

```
xxx/IBM/Lotus/Notes/serversetup.exe -remote
```

Im Setup-Wizard werden folgende Einstellungen vorgenommen:

1. First server or standalone server
2. Server Name: Ubuntu; Server Title: Domino 8.5.1 Development Server
1
3. Organisation Name: Dev; Password: notes
4. Domain Name: Dev
5. Admin Lastname: Admin; Password: notes; Also save a local copy of the
id

6. Setup Internet services for: Web Browser, Internet Mail Clients, Directory services
7. Enable Port Drivers: TCP/IP; Host Name: ubuntu
8. Prohibit Anonymous access, Add LocalDomainAdmins
9. I want to make additional copies of the ID files

Daraufhin kann der Server gestartet werden und ist unter dem Namen ubuntu/Dev über Notes erreichbar:

```
cd /local/notesdata/  
/opt/ibm/lotus/bin/server
```

Achtung: Bei einem Live-Betrieb sollten natürlich nicht die hier verwendeten Passwörter genutzt werden.

3 Softwareversionen

Bei der Entwicklung von Erweiterungen¹ für Lotus Notes 8.5.1 muss darauf geachtet werden die passenden Versionen von Eclipse und dem Lotus Expeditor Toolkit zu verwenden, da Lotus Notes auf Expeditor und somit der Eclipse RCP aufbaut. Auf Grundlage von Tabelle 3.1, die sich auch mit den Aussagen von IBM² deckt, wird im Folgenden Eclipse 3.4.2 und das Lotus Expeditor Toolkit 6.2.1 verwendet.

Notes	Eclipse	Expeditor
8.5.1	3.4.2	6.2.1
8.5.0	3.4.0	6.2.0
8.0.2	3.2.0	6.1.2
8.0.1	3.2.0	6.1.2
8.0.0	3.2.0	6.1.1

Tabelle 3.1: Versionsmapping.³

¹Vgl. Wütherich u. a., 2008, S. 1 ff..

²http://www-10.lotus.com/ldd/lewiki.nsf/dx/Using_IBM_Lotus_Expeditor_Toolkit_6.2.1_with_IBM_Lotus_Notes_8.5.1

³Heisterberg, 2009, S. 1.

4 Lotus Notes, Designer, Administrator 8.5.1

Die Installation von Lotus Notes, Domino Designer und Domino Administrator geht Problemlos von statten. Die Installationsdatei

`notes_designer_admin_851_w32_CZ5S0EN.exe`

ermöglicht die Installation aller benötigten Anwendungen auf einmal. Es muss jedoch darauf geachtet werden, dass die Installation des *Remote Server Setup* manuell aktiviert wird, um später den Domino Server konfigurieren zu können. Es ist ratsam nach der Installation noch keine der Anwendungen zu starten sondern erst den Domino Server zu installieren und zu konfigurieren, damit die Ersteinrichtung gleich richtig durchgeführt werden kann.

Nachdem der Domino Server wie in Kapitel 2 eingerichtet wurde, sollte als erstes der Domino Administrator gestartet werden, um den Administratorzugang einzurichten.

- Benutzer-Id:
 - Ubuntu: `/local/notesdata/admin.id`
 - Windows: `xxx/IBM/Lotus/Notes/admin.id`
- Domino Server: `ubuntu/Dev`
- Connection Type: LAN
- Network Type: TCP/IP
- Server Address: `192.168.0.111`

Nach diesen Einstellungen sollte eine Verbindung zum Server hergestellt werden können und es empfiehlt sich eine neue „Location“ *Admin* anzulegen um schnell zum Administrator-Account wechseln zu können.

- home/mail server, Passthru server, Domino directory server: `ubuntu/Dev`
- Mail file location: on Server
- Mail file: `mail/admin.nsf`
- User ID to switch to: `xxx/admin.id`

Nun wird über *Register* der erste Benutzer registriert und im Folgenden anstatt dem Administrator-Account genutzt. Dies ist ratsam um frühzeitig Berechtigungsprobleme zu erkennen, wie sie evtl. bei XPages auftauchen können.

5 Eclipse 3.4.2

Für die Entwicklung der Plugins wird Eclipse 3.4.2 (Ganymede) benötigt, welches von der Internetseite der Eclipse Foundation ¹ kostenlos heruntergeladen werden kann.

Für standard Java Plug-ins, Eclipse RCP und eRCP Anwendungen empfiehlt IBM² die Eclipse-Version „*Eclipse for RCP/Plug-In Developers*“, daher wird diese heruntergeladen und entpackt. Eine Installation von Eclipse ist nicht nötig, stattdessen starten sie Eclipse durch einen Doppelklick auf die *eclipse.exe*, die sich in dem Ordner befindet, in den sie Eclipse entpackt haben.

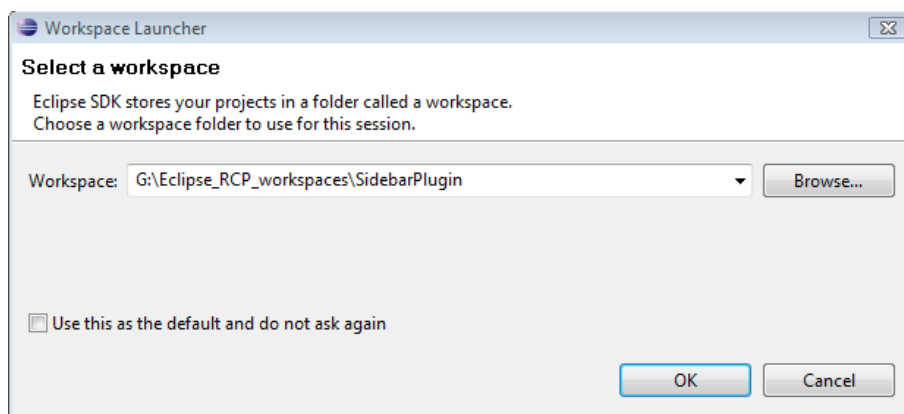


Abbildung 5.1: Auswahl des Workspace

Nach dem Start erscheint das in Abbildung 5.1 dargestellte Dialogfenster zur Auswahl des „*Workspace*“.

Der „*Workspace*“ bezeichnet den Ort auf der Festplatte, wo standardmäßig alle angelegten Projekte, sowie im Unterverzeichnis *.metadata* sämtliche Einstellungen, die in Eclipse vorgenommen werden, gespeichert werden. Dabei wird die Eclipse-Hierarchie eines Projektes auf das Dateisystem abgebildet, so dass sich zum Beispiel die Java Klasse *meineKlasse.java*, des Paketes *de.meinName.MeinProjekt*, in dem Ordner *Aktueller-Workspace/Projekt-Name/src/de/meinName/MeinProjekt* befindet. Da-

¹<http://www.eclipse.org/downloads/packages/release/ganymede/sr2>

²http://www-10.lotus.com/ldd/lewiki.nsf/dx/Using_IBM_Lotus_Expeditior_Toolkit_6.2.1_with_IBM_Lotus_Notes_8.5.1

durch können Sicherheitskopien der Projekte problemlos erstellt werden, da einfach nur der Workspace kopiert werden muss.³

Wenn mehrere unabhängige Plugins oder Anwendungen entwickelt werden besteht die Möglichkeit sich für jedes einen neuen Workspace anzulegen, um die Projekte sauber von einander zu trennen. Damit bereits vorgenommene Eclipse Konfigurationen nicht bei jedem neuen Workspace erneut durchgeführt werden müssen können diese über das Menü *File -> Export -> General -> Preferences* als einzelne Datei exportiert und in den neuen Workspace importiert werden.

Um die einzelnen Projekte mit ihren jeweiligen Workspaces schnell öffnen zu können, ohne sich durch Menüs hangeln zu müssen, kann für jeden Workspace eine Verknüpfung nach folgendem Muster angelegt werden:

```
"c:/eclipse/eclipse.exe" -data c:\workspace-projekt-x
```

Nachdem ein Workspace ausgewählt wurde erscheint der Startbildschirm von Eclipse, der einen Zugriff auf Beispielprojekte, Tutorials, die Hilfe u.A. bietet. Von hier aus wird auch die „*Workbench*“, die eigentliche Programmieroberfläche, gestartet, in dem auf den gebogenen Pfeil geklickt wird. Nach dem Start der Workbench kann das Expeditor Toolkit installiert werden.

Workingsets:

Eine praktikable Alternative, zu der Verwendung unterschiedlicher Workspaces, sind Workingsets. Bei diesen werden alle Projekte innerhalb eines Workspaces belassen, sie werden jedoch nicht gleichzeitig innerhalb der Eclipse Navigator-Ansicht angezeigt.

Dies erreicht man in dem innerhalb der Navigator-Ansicht per Rechts-klick neue Workingsets erstellt und diesen die anzuzeigenden Projekte zugewiesen werden.

Eine gute Anleitung hierfür befindet sich im Blog von Doug Hughes.⁴ Des Weiteren hat Peter Friese ein sehr gutes Video⁵ über Workingsets erstellt, das ich jedem ans Herz legen kann.

³Vgl. Cuber, 2005, S. 38, 84.

⁴<http://www.alagad.com/blog/post.cfm/using-eclipse-working-sets-instead-of-workspaces>

⁵<http://www.peterfriese.de/eclipse-working-sets-part-i/>

6 Lotus Expeditor Toolkit 6.2.1

Das Expeditor Toolkit ist ein Plugin für Eclipse, das die Entwicklung von Expeditor Anwendungen und Plugins vereinfacht und den Konfigurationsaufwand reduziert. Da Lotus Notes 8.5.1 auf Expeditor aufbaut ist das Toolkit auch für die Entwicklung von Lotus Notes Plugins einsetzbar.

Das Expeditor Toolkit kann von der Internetseite der IBM heruntergeladen werden.¹ Die für das Herunterladen nötige Registrierung ist dabei völlig kostenfrei und steht jedem offen.

Nachdem sich die Zip-Datei mit dem Code **CZ66FML** auf der Festplatte befindet, muss diese in ein beliebiges Verzeichniss entpackt werden. Daraufhin wechselt man zu Eclipse und öffnet über das Menü den Update-Manager: **Help -> Software Updates....**

Im Update Manager wird auf die Register-Karte **Available Software** gewechselt und die Schaltfläche **Add Site** betätigt. Nun wird über die Schaltfläche **Local** das Verzeichniss **Expeditor_Toolkit_Install** geöffnet, welches sich in dem zuvor erzeugten Expeditor Toolkit Ordner befindet. Daraufhin erscheinen im Update-Manager sämtliche Funktionen die das Toolkit zur Verfügung stellt und können einzeln installiert werden. Wir wählen alle Funktionen und betätigen danach die Schaltfläche **Install**. Nachdem im folgenden Fenster die Nutzungsbedingungen akzeptiert wurden beginnt die Installation von Expeditor, die mit einem Neustart von Eclipse abgeschlossen wird.

Nach dem Neustart von Eclipse erscheint das Konfigurationsfenster aus Abbildung 6.1. Bei der Entwicklung von Plugins für Lotus Notes muss die „Testumgebung“ auf Lotus Notes gestellt werden und der Pfad dem Verzeichnis **framework\rpc\ eclipse** innerhalb der Lotus Notes Installation entsprechen. Daraufhin sollten die verbleibenden Felder automatisch mit Daten über die von Lotus Notes verwendete Java-Version gefüllt werden. Wenn ausschließlich Plugins für Notes 8.5.1 entwickelt werden, kann als „Konformitätsstufe des Compilers“ **1.6** gewählt werden, wodurch neuere Sprach-Elemente von **Java 6** verwendet werden können. Soll das Plugin auch in älteren Notes-Versionen (8.x) funktio-

¹<http://www14.software.ibm.com/webapp/download/nochargesearch.jsp?q=Lotus+Expeditor+Toolkit+6.2>

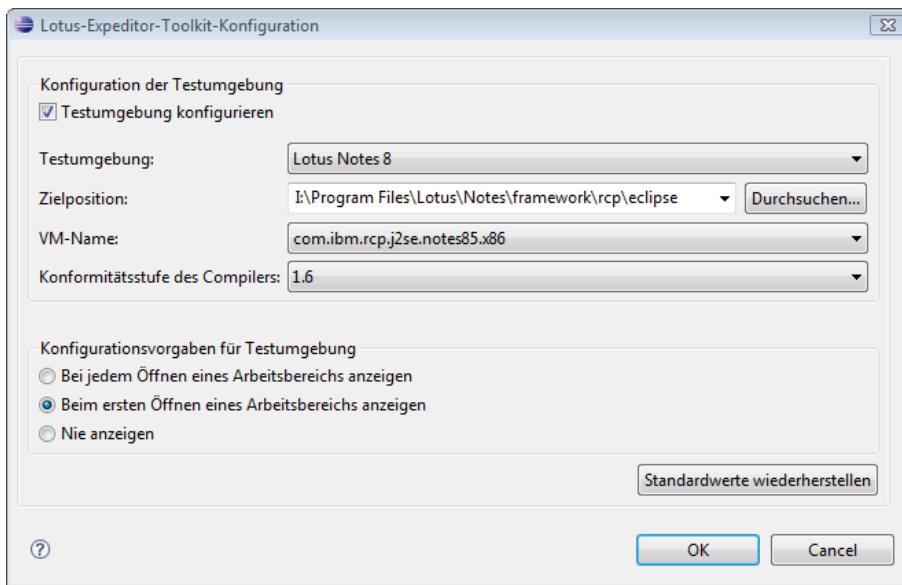


Abbildung 6.1: Expeditor Start-Konfiguration

nieren, so sollte als Konformitätsstufe **1.5** gewählt werden. Nachdem die Einstellungen vorgenommen wurden erscheint wieder die Eclipse Startseite, die um zusätzliche Schaltflächen für Expeditor Beispiele und Tutorials erweitert wurde.

Um Plugins in Lotus Notes testen zu können wählen wir im Menü **Run -> Run Configurations**, woraufhin sich ein Konfigurationsfenster öffnet. In der Liste links wird „Client Services“ markiert, welches von Expeditor hinzugefügt wurde und dann der darüber liegende Knopf zur Erstellung einer neuen Konfiguration betätigt. Die daraufhin erzeugte Konfiguration ist bereits fertig eingestellt, es muss ihr nur noch ein aussagekräftiger Name wie z.B. **Notes 8.5.1** gegeben werden und im Abschnitt **Workspace Data** muss die **Location** angepasst werden. Ich verwende `${workspace_loc}/../runtime-Notes8.5.1`.

Um das Plug-in schneller in Notes testen zu können, kann unter dem Reiter „Common“ noch ein Häkchen vor **RUN** im Bereich **Display in favorites menu** gesetzt werden.

Das Expeditor Toolkit ist nun fertig konfiguriert und erstellte Plugins können getestet werden indem die Run-Schaltfläche für die zuvor erstellte Konfiguration betätigt wird.²

²http://www-10.lotus.com/ldd/lewiki.nsf/dx/Using_IBM_Lotus_Expeditor_Toolkit_6.2.1_with_IBM_Lotus_Notes_8.5.1

7 Testplugin

Zum Testen der Installation wird ein einfaches Plugin erstellt, das der Lotus Notes Sidebar ein leeres Element hinzuzufügen. Im Anhang befindet sich der komplette benötigte Code.

7.1 Grundlagen

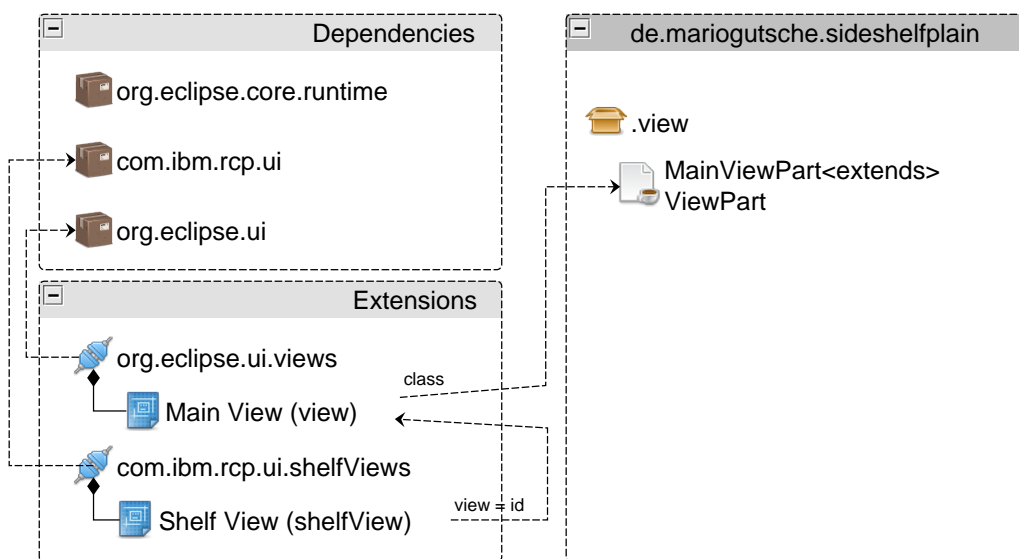


Abbildung 7.1: Extension Points der Sidebar

Um der Lotus Notes Sidebar neue Elemente hinzuzufügen, ist es nötig zwei vorhandenen Extension Points durch eigene Extensions zu erweitern. Der Erste ist **org.eclipse.ui.view**, dieser muss immer erweitert werden, wenn der Eclipse RCP neue Anzeigeelemente hinzugefügt werden sollen, und dazu zählen auch die Inhalte der Sidebar. Die eigene Erweiterung besteht aus einem Verweis auf eine selbst angelegte Java Klasse, die entweder das Interface `org.eclipse.ui.IViewPart` implementieren muss, oder der Einfachheit halber von der Klasse `org.eclipse.ui.part.ViewPart` erbt.

Klassen, die von `ViewPart` erben, müssen die Methode `createPartControl(Composite parent)` implementieren. Diese muss den Quellcode zur Steuerung des View-Inhaltes enthalten. Wird die Methode leer gelassen, entsteht eine leere View, was für Testzwecke jedoch ausreichend ist. Zusätzlich muss noch

die Methode `setFocus()` implementiert werden, diese kann jedoch inhaltslos bleiben, hauptsächlich sie existiert.

Der zweite Extension Point wurde von IBM der Eclipse RCP hinzugefügt und lautet **com.ibm.rcp.ui.shelfViews**. Die Extension, die für diesen angelegt werden muss, verweist nicht etwa auf eigenen Quellcode, sondern über das Attribut „view“ lediglich auf die „id“ der zuvor angelegten View-Extension, um diese View nicht irgendwo zu erzeugen, sondern in der Sidebar.

Das neu zu erstellende Sidebar Plug-in ist somit eine ganz normale Eclipse RCP View, die wie jede andere Eclipse View programmiert werden kann. IBM's Erweiterung ist erst von Belang, wenn es um die Positionierung des Plug-ins innerhalb der Lotus Notes Benutzeroberfläche geht. Abbildung 7.1 veranschaulicht das Zusammenspiel zwischen Extension-Point, Extensions und selbst zu erstellendem Code.

7.2 Implementierung

Wir wechseln in die Eclipse Workbench und erzeugen links im „**Package Explorer**“ per Rechtsklick **New -> Other** ein „**Plug-in Project**“ aus der Kategorie „**Plug-in Development**“. Das Projekt bekommt den Namen **“de.mariogutsche.sideshelfplain”**, da es üblich ist hier die Wurzel der geplanten Java-Paketstruktur zu verwenden und so die Aktivator-Klasse, die für das Lifecycle-Management des Plugin verantwortlich ist, gleich im richtigen Paket erzeugt wird.

Unter „Target Platform“ wird festgelegt in welcher Umgebung unser Plugin laufen soll. Da wir es ausschließlich für Notes/Eclipse nutzen, wird Eclipse 3.4 gewählt, wodurch wir sämtliche Eclipse-Features nutzen können. Soll das Plugin nur an das OSGI-Framework gebunden sein und nicht an Eclipse, wodurch einige Eclipse-Features nicht mehr ohne Mehraufwand verwendbar sind, so muss „OSGI-Equinox“ als Target Platform gewählt werden. Durch eine weitere Einschränkung auf „OSGI-Standard“ wird dafür gesorgt, dass das Plugin auch unter anderen OSGI Implementierungen lauffähig ist. Dann klicken wir **„Next“**.

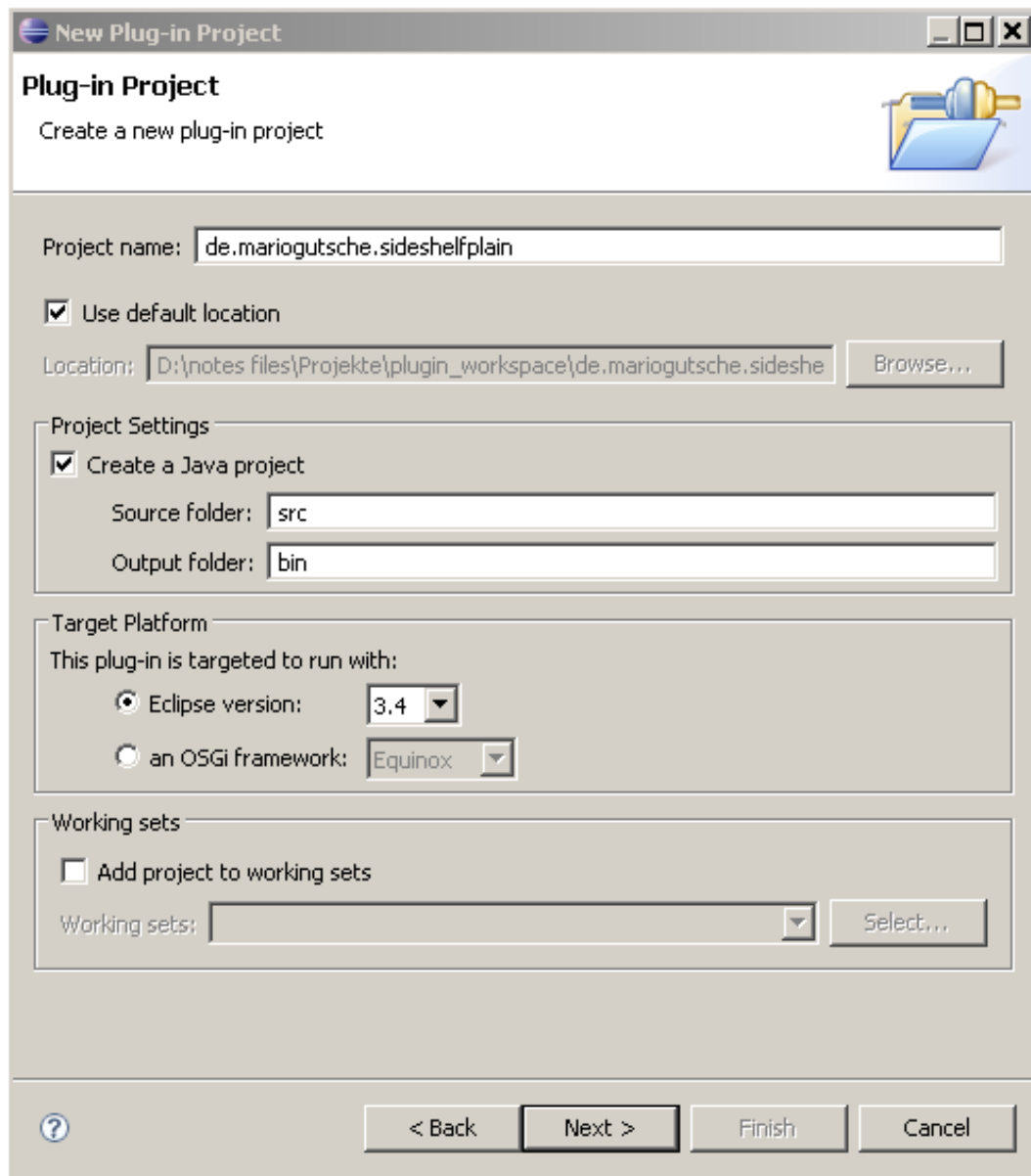


Abbildung 7.2: Plug-In Wizard 1

In dieser, in Abbildung 7.3 dargestellten Eingabemaske, wird als ID "de.mariogutsche.sideshelfplain" eingetragen, der Plug-in Name kann beliebig gesetzt werden, ich wähle „Sideshelf Plug-in“. Als „Plugin Provider“ wird der Urheber/Hersteller o.Ä. eingetragen und das Execution Environment an die älteste Notes Version angepasst, die unterstützt werden soll. Bei Notes 8.5.1 entspricht dies Java 1.6. „Generate an activator...“ muss für die Steuerung des Plug-In Lebenszyklus aktiviert werden, ebenso wie „make contributions to the ui“, da unser Plug-In die Benutzeroberfläche nutzt. Als Klassenname des Activators wird üblicherweise ein Name verwendet, der für das Plug-in spricht, wir wählen „DemoPlugin“, die Paktestruktur wird übernommen.

Seit Version 3.4 des Eclipse Plugin Development Environment (PDE) gibt es die Möglichkeit „Enable API Analysis“ zu aktivieren. Die Auswahl ist hier jedem selbst überlassen, ich empfehle es jedoch für ernsthafte Projekte, da dadurch Tools zur Verfügung stehen, die einen dabei unterstützen, wartbare APIs zu entwerfen.¹ Jetzt klicken wir auf „Finish“ und bestätigen die Frage ob in die Plugin-Perspektive gewechselt werden soll.

In der Plugin-Perspektive sollte nun der Editor für die Datei MANIFEST.MF, wie in Abbildung 7.4 dargestellt, sichtbar sein.

Hier wechseln wir auf den „Extensions“-Reiter und fügen die Erweiterung „org.eclipse.ui.views“ hinzu, die Extension Details können optional mit einer ID und einem Namen für diese Extension-Instanz befüllt werden, über die die Extension innerhalb der Laufzeitumgebung identifiziert werden kann, für dieses Beispiel ist es jedoch nicht nötig.

Jetzt wird per Rechts-klick auf „org.eclipse.ui.views“ und **New -> View** eine neue Ansicht mit folgenden Eigenschaften erstellt:

id de.mariogutsche.sideshelfplain.views.MainViewPart

name Demo View

(dieser Wert wird in der Sidebar im Titel angezeigt)

class de.mariogutsche.sideshelfplain.views.MainViewPart

(diese Klasse wird im nächsten Schritt erst erzeugt)

¹Vgl. Aniszczyk, 2008, S. 1 ff.

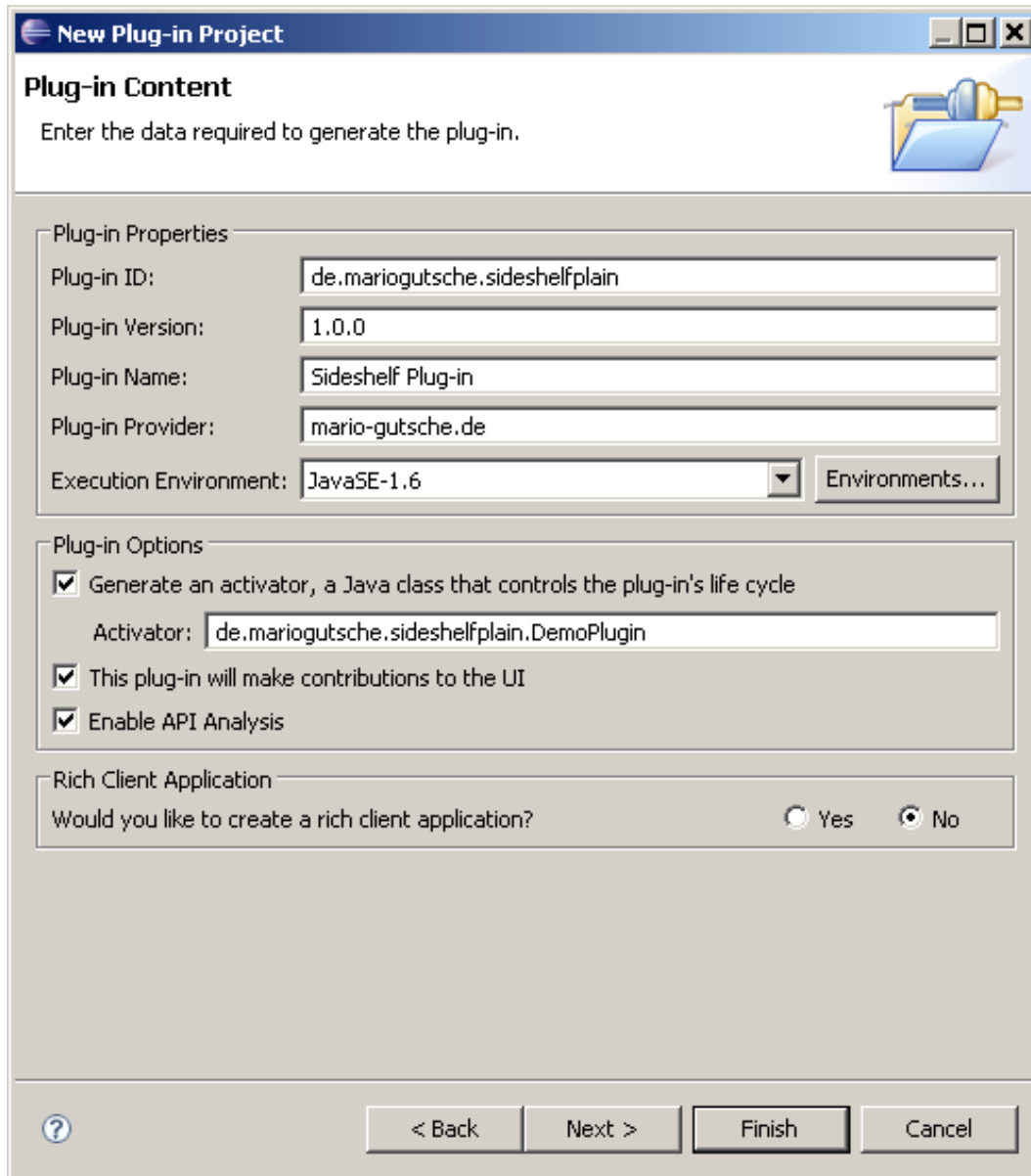


Abbildung 7.3: Plug-In Wizard 2

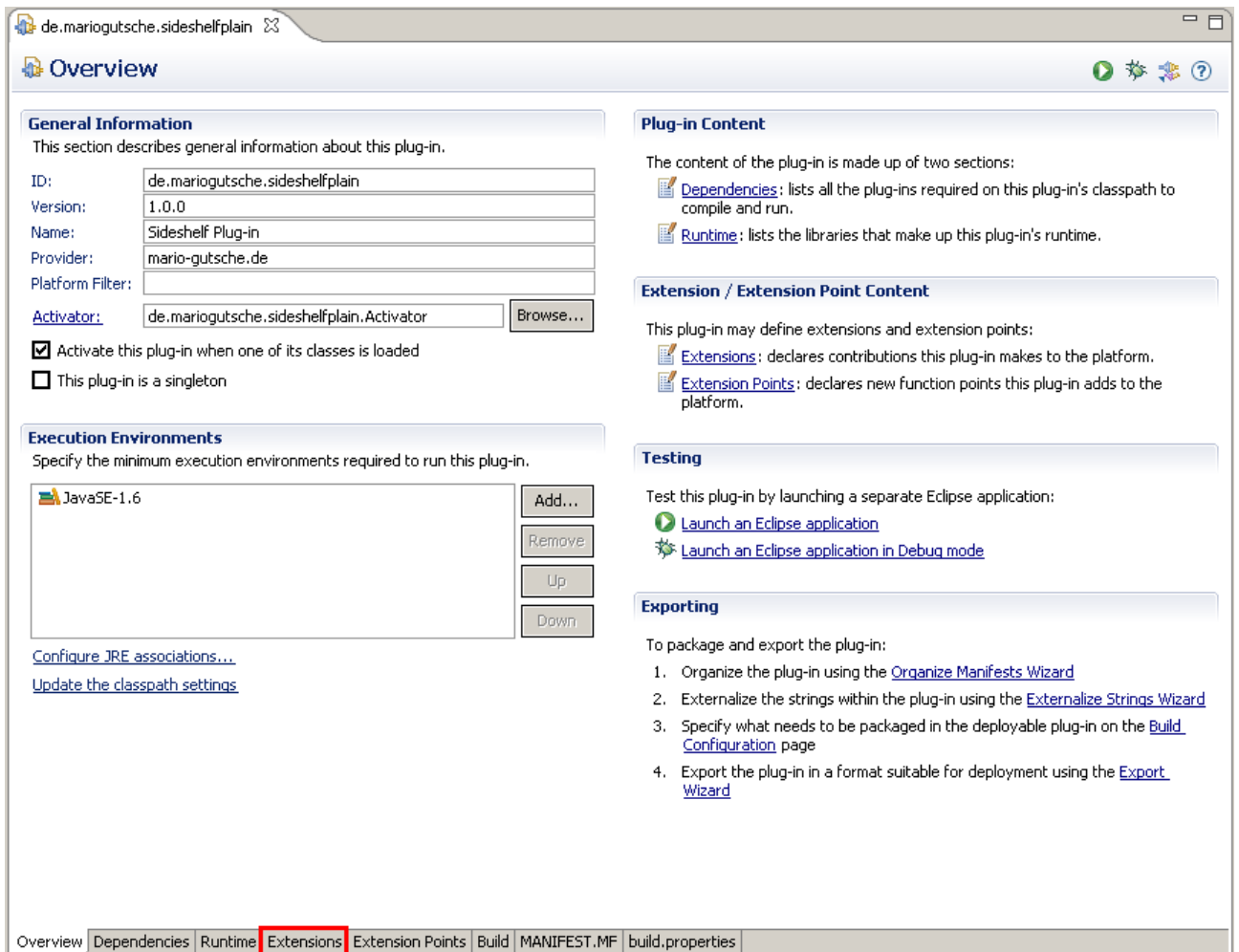


Abbildung 7.4: MANIFEST Editor

Optional kann noch eine Kategorie festgelegt werden, unter der alle zugehörigen Views gruppiert werden, wenn man *Window -> Show View* ausführt.

Nun muss die referenzierte Klasse

de.mariogutsche.sideshelfplain.views.MainViewPart, die für die Darstellung des Inhaltes des Sidebar-Elements zuständig ist, erstellt werden. Hierzu kann einfach auf die Bezeichnung **class***: vor dem Feld, in das wir die Klasse eingetragen haben, geklickt werden.

Der nun erscheinende Dialog ist bereits richtig ausgefüllt und muss nur noch bestätigt werden. In der Datei **MainViewPart.java** kann der Ansicht nun Inhalt über die Funktion **createPartControl()** hinzugefügt werden, für unsere Testzwecke erstellen wir die obligatorische „Hallo Welt!“ Nachricht. Der Quellcode ist in Anlage 1.A: dargestellt.

Damit die soeben erstellte Ansicht in der Sidebar von Lotus Notes erscheint, muss nun noch eine Extension zum Extension Point **com.ibm.rcp.ui.shelfViews** hinzugefügt werden. Dazu gehen wir wieder in den Extension-Reiter, klicken auf „Add...“ und deaktivieren diesmal die Funktion „**Show only extension points from the required plug-ins**“ da das Paket **com.ibm.rcp.ui** noch nicht im „Dependencies“-Reiter unter **Required Plug-ins** eingetragen ist, und wir den benötigten Extension-Point **com.ibm.rcp.ui.shelfViews** somit nicht sehen würden.

Auf diese Weise kann die „Dependency“ automatisch gesetzt werden, was uns Arbeit erspart. *(Die Eintragung von benötigten Bibliotheken als „Dependency“ sorgt dafür, dass diese automatisch dem Klassenpfad hinzugefügt werden, die Reihenfolge in der Dependency-Liste entspricht dabei der Reihenfolge, in der die Elemente zur Laufzeit geladen werden.)*

Wir wählen diesen Extension-Point nun in der Liste aus und schließen den Dialog, darauf wird automatisch die Liste der Abhängigkeiten um den Eintrag **com.ibm.rcp.ui** ergänzt.

Um in größeren Projekten unbenötigte Dependencies zu finden, die während der Entwicklungsphase testweise hinzugefügt wurden, bietet Eclipse die Funktion „Find Unused Dependencies“, die per Rechts-klick in das Dependency-Fenster gestartet werden kann.

Um die in der Sidebar Shelf-View nun die MainView anzuzeigen werden fol-

gende Einstellungen in der Shelf-View vorgenommen:

id de.mario_gutsche.sideshelfplain.views.shelfView (Die eigene Id der shelfView und somit frei wählbar)

view de.mario_gutsche.sideshelfplain.views.MainViewPart
(Die Id der MainView)

Nun kann das Plug-in über **Run -> Run Configurations -> Notes 8.5.1** getestet werden, vorher müssen eventuell offene Notes-Fenster geschlossen werden, da ansonsten eine Fehlermeldung erscheint.

Sollten auf der Console von Eclipse zahlreiche Fehler auftauchen, dass Pakete nicht gefunden wurden, prüfen Sie, ob Sie in der Run-Configuration wie in Kapitel 6 beschrieben, die Workspace Location angepasst haben. Sollten dennoch viele Fehler erscheinen, aktivieren Sie in der Run-Configuration **Clear: Workspace**.

7.3 API-Tools

Falls beim Anlegen des Projektes die Api-Tools aktiviert wurden (*Enable Api Analysis*) so wird in der **Problems-View** höchstwahrscheinlich die Fehlermeldung „An API baseline has not been set for the current workspace“ angezeigt werden.

Die API-Baseline ist quasi ein Grundzustand, mit dem spätere Entwicklungen abgeglichen werden, um festzustellen ob es Änderungen in der API und somit potentielle Probleme gibt.

Um eine API-Baseline zu setzen, öffnen Sie „Window -> Preferences“ und dort „Plug-in Development -> API Baselines“. Über **Add Baseline** fügen Sie ihr aktuelles Eclipse-Verzeichnis hinzu.

Für weitere Informationen verweise ich auf die Artikel *API Tools in Eclipse: An introduction*², *PDE/API Tools/User Guide*³ sowie *Comparing API Baselines*⁴.

²<http://www.ibm.com/developerworks/opensource/library/os-eclipse-api-tools/index.html>

³http://wiki.eclipse.org/PDE/API_Tools/User_Guide

⁴<http://eclipsesource.com/blogs/2009/04/28/comparing-api-baselines/>

Literaturverzeichnis

[Aniszczyk 2008] ANISZCZYK, Chris: *API Tools in Eclipse: An introduction*. Version: 09 2008. <https://www.ibm.com/developerworks/opensource/library/os-eclipse-api-tools/>, Abruf: 02.03.2010

[Cuber 2005] CUBER, Ulrich: *Einstieg in Eclipse 3.0. Einführung, Programmierung, Plugin-Nutzung*. 1. Auflage. Galileo Press, 2005 <http://amazon.de/o/ASIN/3898425525/>. – ISBN 9783898425520

[Heisterberg 2009] HEISTERBERG, M.: *Notes / Eclipse / Lotus Expeditor mapping*. Version: 11 2009. http://www.lekkimworld.com/pages/notes_eclipse_xpd.html, Abruf: 03.11.2009

[Wütherich u. a. 2008] WÜTHERICH, G. ; HARTMANN, N. ; KOLB, B. ; LÜBKEN, M.: *Die OSGI Service Platform-Eine Einführung mit Eclipse Equinox*. 1. Auflage. dpunkt Verlag, 2008 <http://amazon.de/o/ASIN/389864457X/>. – ISBN 9783898644570

Anlagenverzeichnis

Anlage 1: Quellcode	ii
Anlage 1.A: MainViewPart.java	ii
Anlage 1.B: MANIFEST.MF	iii
Anlage 1.C: plugin.xml	iv

Anlage 1: Quellcode

Anlage 1.A: MainViewPart.java

```
package de.mariogutsche.sideshelfplain.views;

import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Label;
import org.eclipse.ui.part.ViewPart;

public class MainViewPart extends ViewPart {
    private Label label;

    public MainViewPart() {
        super();
    }

    @Override
    public void createPartControl(Composite arg0) {
        this.label = new Label(arg0, SWT.NONE);
        label.setText("Hallo Welt!");
    }

    @Override
    public void setFocus() {
        // give focus to first created control
        label.setFocus();
    }
}
```

Anlage 1.B: MANIFEST.MF

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Sideshelf Plug-in
Bundle-SymbolicName: de.mariogutsche.sideshelfplain;singleton:=true
Bundle-Version: 1.0.0
Bundle-Activator: de.mariogutsche.sideshelfplain.Activator
Require-Bundle: org.eclipse.ui,
    org.eclipse.core.runtime,
    com.ibm.rcp.ui
Bundle-ActivationPolicy: lazy
Bundle-RequiredExecutionEnvironment: JavaSE-1.6
```

Anlage 1.C: plugin.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
<plugin>
  <extension
    point="org.eclipse.ui.views">
    <view
      class="de.mariogutsche.sideshelfplain.views.MainViewPart"
      id="de.mariogutsche.sideshelfplain.views.MainViewPart"
      name="Main View"
      restorable="true">
    </view>
  </extension>
  <extension
    point="com.ibm.rcp.ui.shelfViews">
    <shelfView
      id="Shelf View"
      region="TOP"
      showTitle="true"
      view="de.mariogutsche.sideshelfplain.views.MainViewPart">
    </shelfView>
  </extension>
</plugin>
```