In dieser Starthilfe

Eclipse Starthilfe

1.1. Um was geht's?

Sie wollen sich mit **Eclipse**, einer gratis Java-IDE aus dem IBM *Eclipse Projekt* (Eclipse.org), vertraut machen. Ich wollte das auch und finde Eclipse sehr brauchbar. Einiges finde ich besser als im JBuilder (Refactoring, Plugins, Imports organisieren u.v.m.). JBuilder hat andere Vorteile (Help, UML, grosser Funktionsumfang [Enterprise Edition]). Aber es gibt noch viele andere IDE's, wie beispielsweise intelliJ (*www.intellij.com*).

1.2. Allgemeine Einführung in die Eclipse-Plattform

Wer noch gar nichts von Workbench, Plugin und Co gehört hat, sollte hier zuerst vorbeischauen. Sie können aber auch gleich starten und falls nötig diesen Abschnitt später durchlesen.

1.2.1. Wichtige Begriffe

1.2.1.1. Plugins

Eclipse ist ein Rahmenwerk zur Integration verschiedenster Anwendungen. Eine solche Anwendung ist z.B. die mitgelieferte Java Entwicklungsumgebung JDT (Java Development Tooling). Das neuartige an Eclipse ist, dass Erweiterungen in Form von *Plugins* zur Verfügung gestellt werden. Plugins werden in das /Plugin Verzeichnis entzippt und von Eclipse automatisch erkannt integriert.

1.2.1.2. Workspace

Eclipse bietet weiterhin Funktionalität zum Verwalten von *Ressourcen* (normalerweise Dateien) auf der Festplatte. Diese befinden sich im sogenannten *Workspace*, einem speziellen Verzeichnis im Dateisystem.

Ich lasse meine Projekte gleich im Workspace und exportiere die Dateien, wenn ich sie verteilen will. Im Workspace finden Sie aber auch Informationen über Classpath u.ä. Sie können aber problemlos für jedes Projekt irgendwo im Netzwerk angeben und dann alles projektbezogen irgendwo speichern. Wir kommen nach der Installation darauf zurück.

Verändert eine Anwendung eine Ressource im Workspace, können andere Programme mithilfe eines Benachrichtigungsmechanismus darüber informiert werden.

1.2.1.3. Perspektiven

Mehrere Fenster des Eclipse Workbench werden in sogenannten *Perspektiven* zusammengefasst. Eine Perspektive besteht aus *Views* und *Editoren*.

- Ein View bietet ist eine Sicht auf Ressourcen.
- Ein Editor dient zum Bearbeiten einer Ressource.
  Ressourcen haben einen fixen Lebenszyklus:
  Laden-Verändern-Speichern
  Der Benachrichtigungsmechanismus kann erst aktiv werden, wenn eine Ressource abgespeichert wird.

Sie können View und Editoren ziemlich frei kombinieren. Das testen wir gleich nach der Installation.

1.3. Die Installation von Eclipse

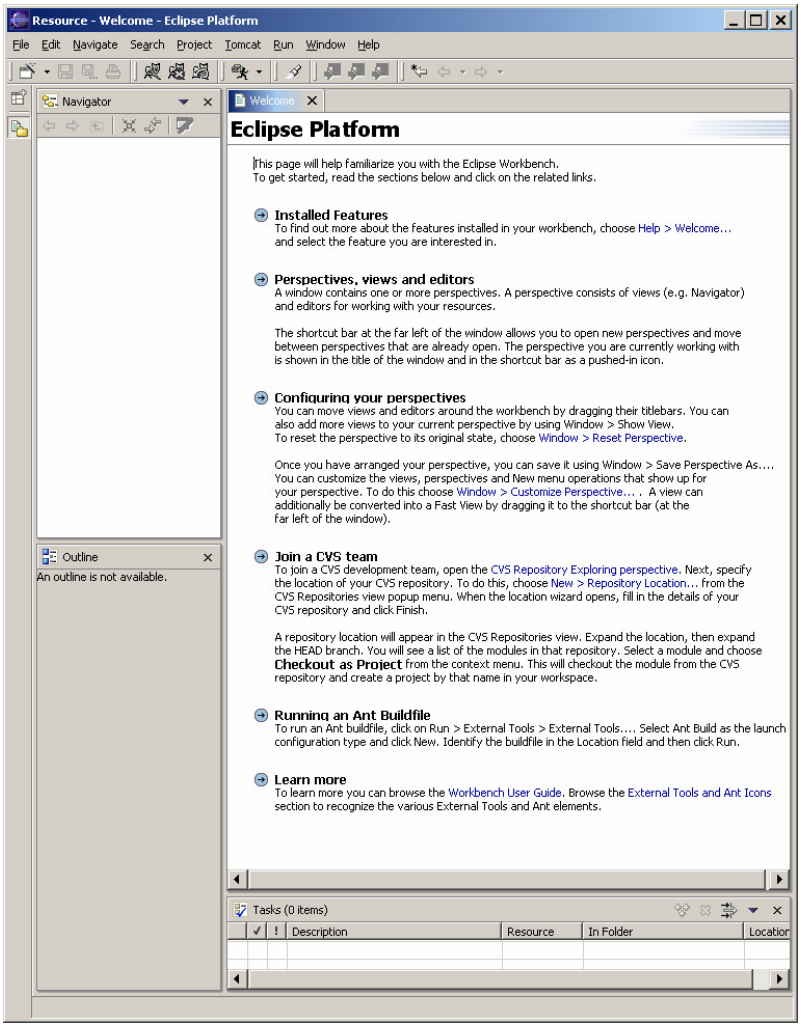
Benötigt werden:

- Ihr Rechner (Windows, Linux, Solaris, Mac OS/X-Betriebssystem)
  Wie immer frisst Java einigen Speicher: 256 MB oder mehr als Hauptspeicher sollte der PC schon haben.
- Eine Java 2 Laufzeitumgebung (JRE) oder ein Java 2 Software Development Kit (J2SDK) als Eclipse Laufzeitumgebung. Eclipse setzt Version 1.3 oder höher voraus. Ich verwende 1.4.2.
  Die Version für Ihre eigenen Java Projekte können Sie separat davon angeben!
- Eclipse SDK:
  - `eclipse-SDK-2.1.1-win32.zip`

- Installieren Sie das JRE bzw. J2SDK, falls noch nicht installiert.
  - Entpacken Sie das Eclipse-Archiv an eine beliebige Stelle auf der Festplatte.
    Ich verwende einfach `c:\` zum Entzippen und `C:\eclipse` als Eclipse Verzeichnis (im folgenden als *eclipse_home* bezeichnet).

    Eclipse findet unter Windows ein installiertes JRE/SDK automatisch über die Windows Registry. Sie können auch ein `jre`-Verzeichnis (mit allem, was dazugehört) in das `eclipse_home`-Verzeichnis kopieren (JBuilder macht das auch so. Sie haben damit alles ab einem Root Verzeichnis (`eclipse_home`) was zusammengehört.
    Zudem könnten Sie beim Starten von Eclipse den Pfad zum JRE/SDK angeben: Kommandozeilenoption `-vm`.

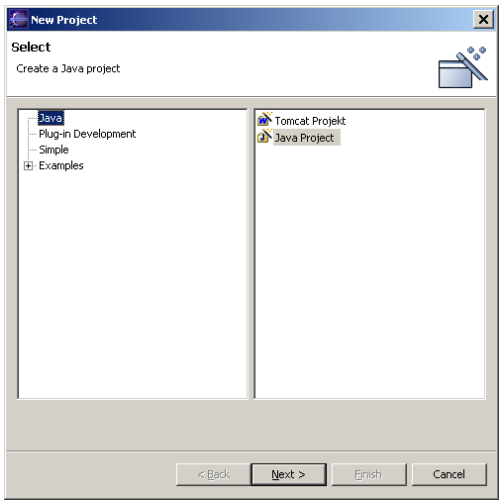
Eclipse starten Sie ganz einfach, indem Sie auf `eclipse.exe` in `eclipse_home` klicken. Wie bei Java so üblich bei Java dauert das Starten eine Weile. Nach einiger Zeit sehen Sie folgenden Bildschirm:



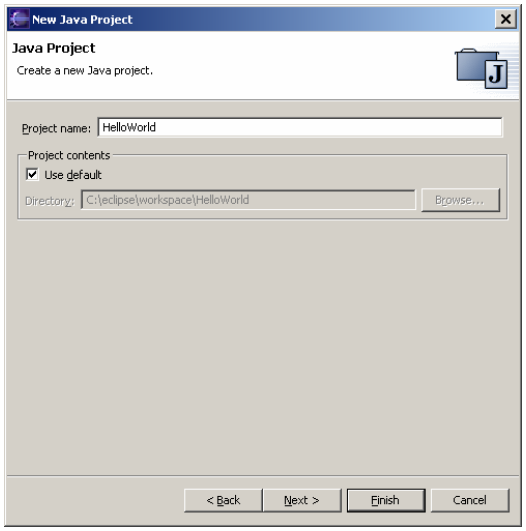
Wie Sie sehen, entspricht diese Einführung grob dem Raster dieses ersten Screens.

1.4. Ihr erstes Projekt

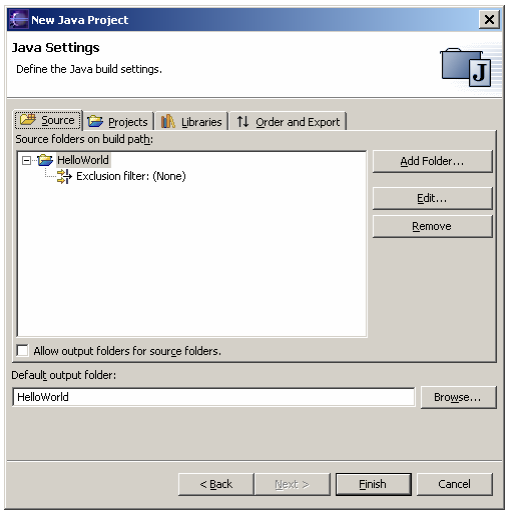
Wählen Sie "`File->New->Project`". Da ich das Plugin für Tomcat bereits installiert habe kann ich aus zwei Projekttypen auswählen.



Wählen Sie das Java Projekt (rechts) aus.



Geben Sie dem Projekt einen Namen: "`HelloWorld`".

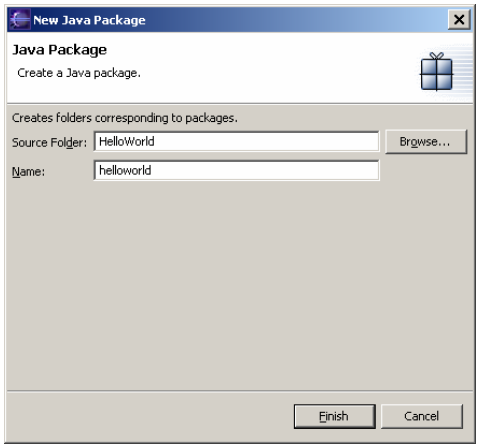


Den Rest lassen wir einfach stehen.

Sie könnten angeben, welche Klassen sie benötigen. Im ersten Beispiel kommen wir mit den Java Standard-Klassen aus.

Später können wir beliebig viele Java Archive einbinden (SOAP, XERCES, ...).

Jetzt legen wir im Projekt ein Package an.

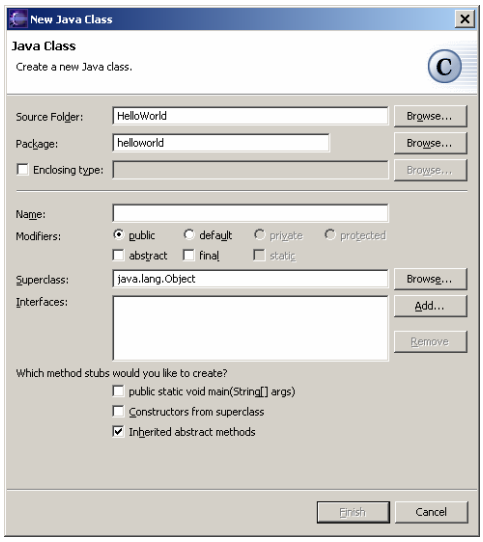


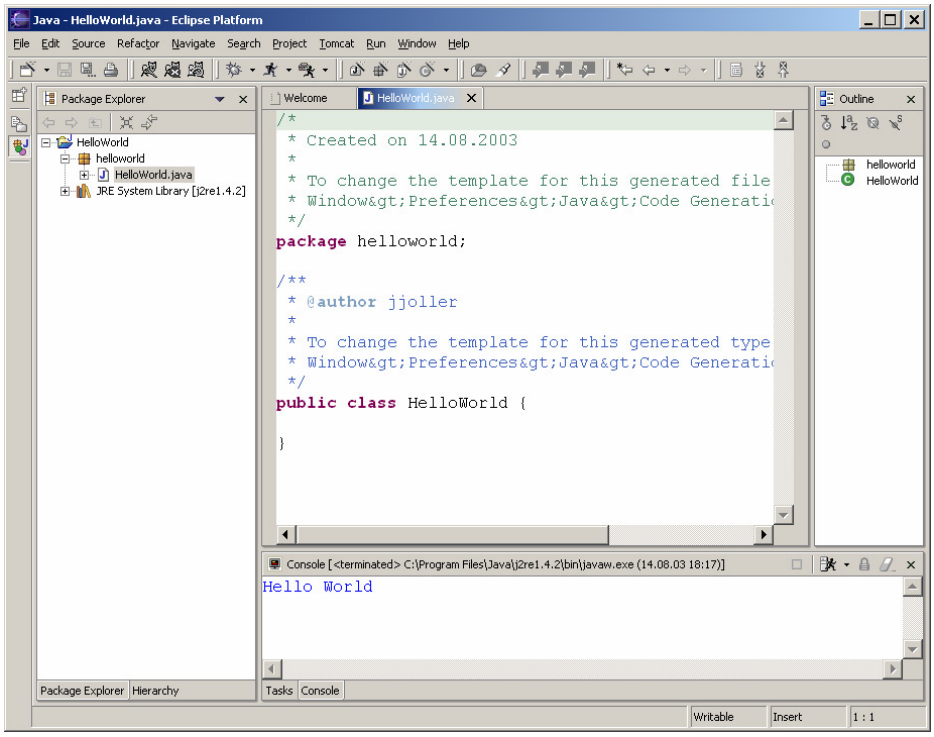
Falls Sie diesen Schritt weglassen, wird ein Default Package angelegt.

Jetzt sind wir bereit, unsere erste Java Klasse anzulegen.

Eclipse überprüft die Gross/Kleinschreibung.
Falls Sie einen Package Namen mit Grossbuchstaben beginnen, erscheint ein entsprechender Hinweis.

Wir legen die Java Klasse im oben angelegten Package an und taufen die Klasse "`HelloWorld`"





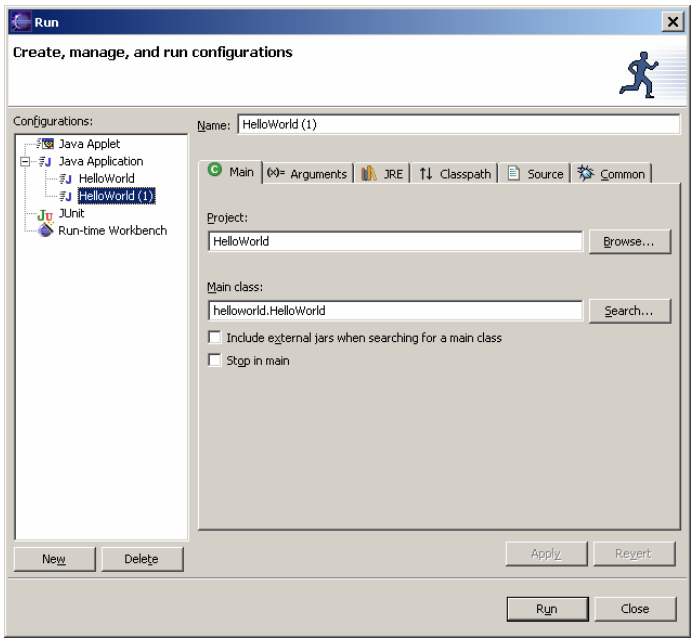
Eclipse generiert für Sie einen Rahmen, mit einem minimalen JavaDoc.

Tippen Sie die fehlenden Zeilen ein.
Speichern Sie den Sourcecode (File).

Unter "Project-Rebuild" können Sie den Java Code übersetzen.

Zum Starten haben Sie mehrere Optionen:

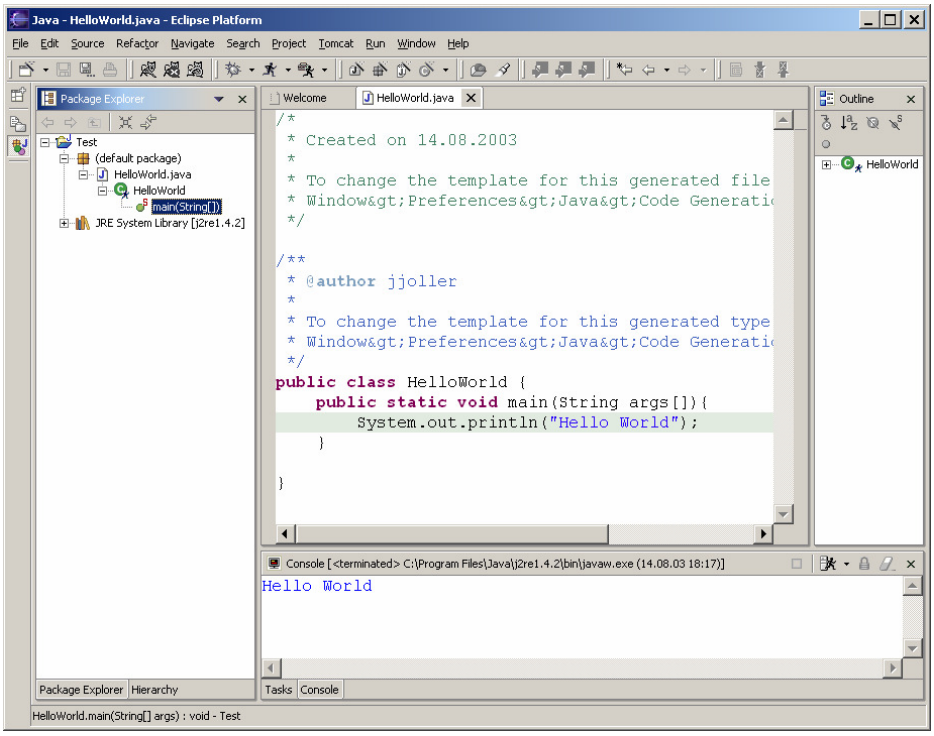
1) Unter "Run..." können Sie ein Profil für Ihre Anwendung anlegen:



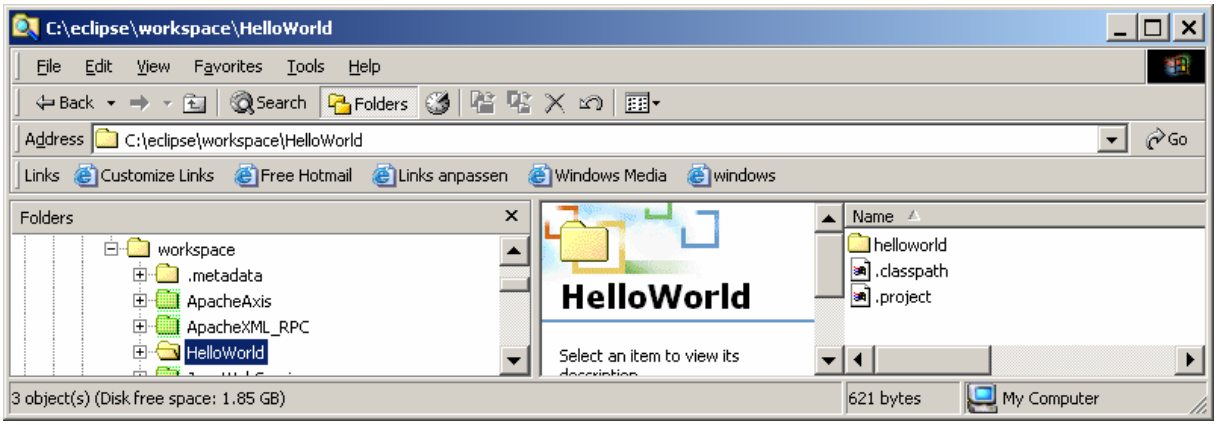
2) Sie können auch "`Run As..`" anklicken und Ihr Java HelloWorld "`Run As Application`" starten.

1.4.1. Ordnerstruktur bei neuen Java-Projekten.

Wenn Sie mit den Standardeinstellungen arbeiten, legt Eclipse die Java und die Class Dateien ins selbe Verzeichnis:



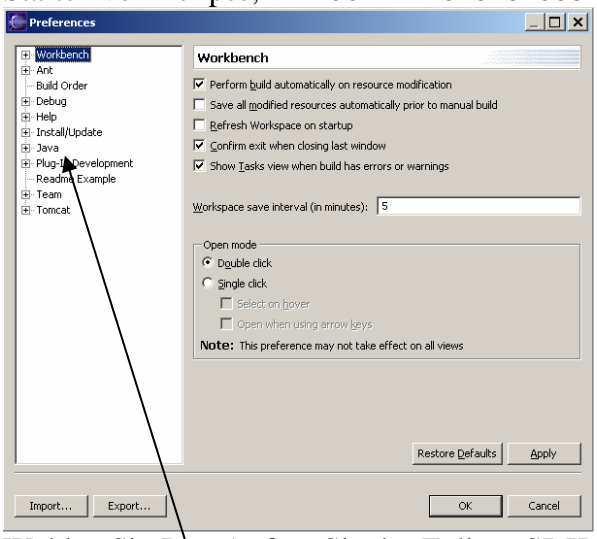
Falls Sie damit leben können, brauchen Sie nichts zu ändern:



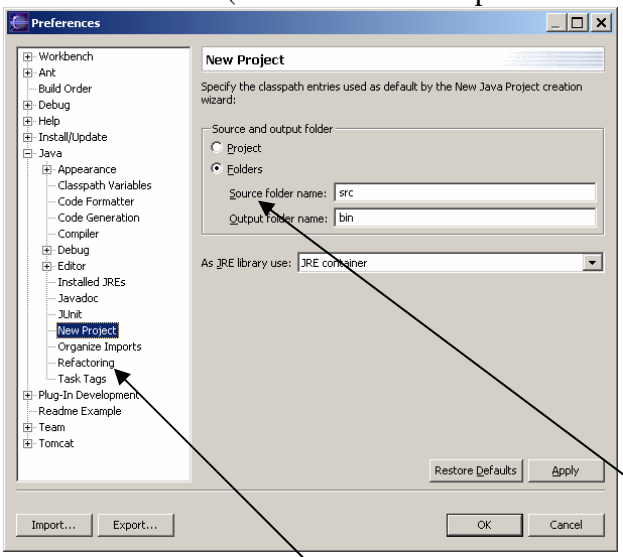
1.5. Grundeinstellungen anpassen

In der Regel ist es aber angenehmer, Class und Java Dateien in getrennten Verzeichnissen zu speichern. Dazu müssen wir müssen wir Ordner anlegen und Eclipse passen umkonfigurieren:

1. Starten von Eclipse, "`Window->Preferences`"



2. Wählen Sie Java (sofern Sie das Eclipse SDK installiert haben).

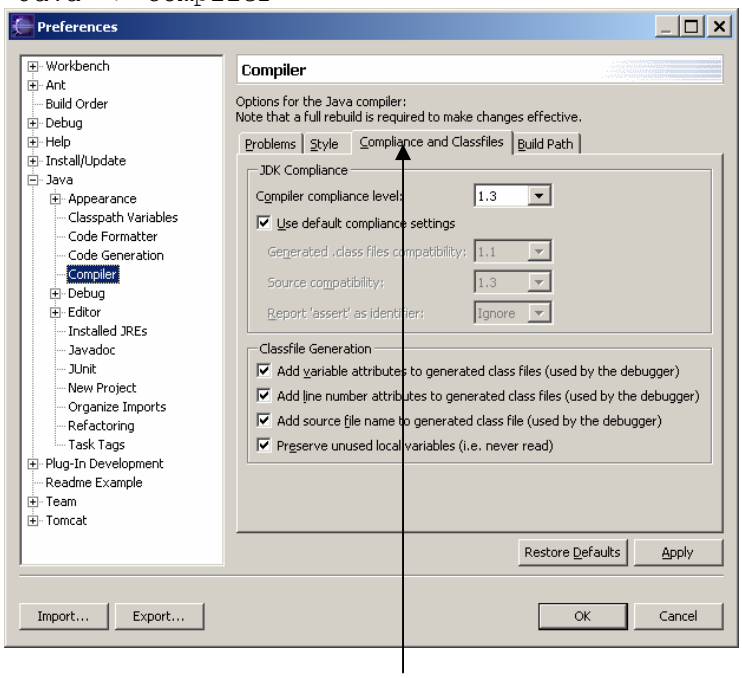


3. Wählen Sie "`New Project`" und aktivieren Sie dort die "`Folders`" Option. Mit "`Apply`" speichern Sie diese.

1.5.1. J2SDK festlegen

Die J2SDK Version können Sie analog zu den Verzeichnissen unter "Preferences" festlegen.

1. "`Window -> Preferences`"
2. "`Preferences -> Java`"
3. "`Java -> Compiler`"

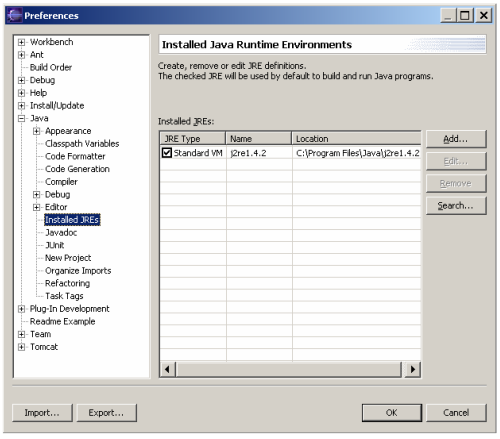


4. "`Compiler -> Compliance and Classifiers`"
   Umschalten von 1.3 auf 1.4
5. "`Apply, OK`"

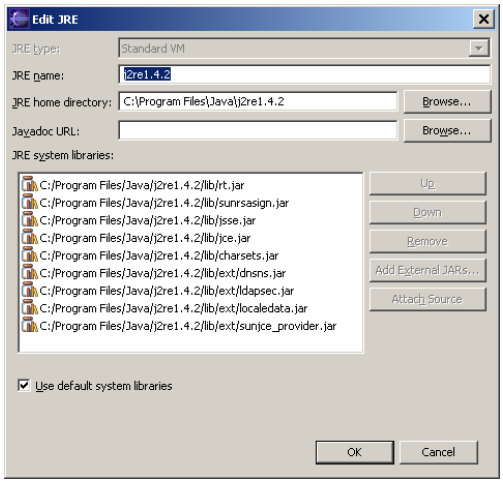
1.5.2. Ein neues JRE/JDK installieren

Falls Sie kein J2SDK installiert haben, kann Ihnen Eclipse weniger Kontext spezifische Hilfen anbieten. Eclipse benötigt dazu `src.zip`. Das Installieren eines neuen J2SDK's ist denkbar einfach:

1. installieren Sie das gewünschte J2SDK
2. starten Sie Eclipse. Dann "`Window -> Preferences`"
3. "`Preferences -> Installed JRE`"

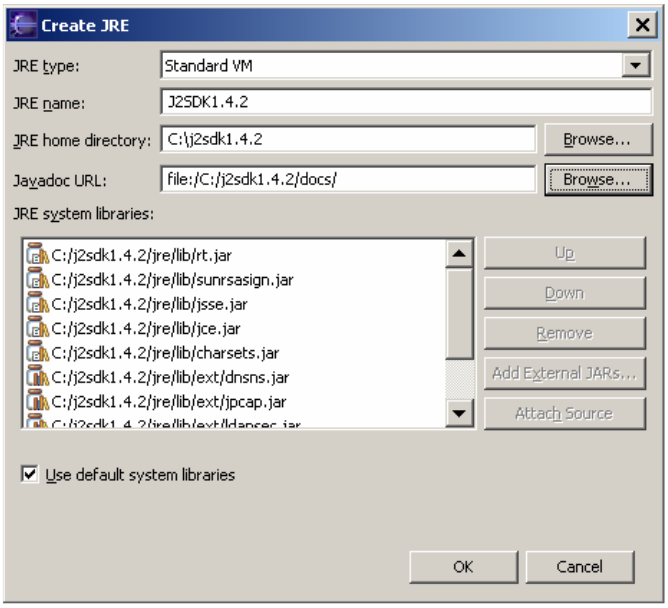


4. "`Installed JRE -> Edit`"



wir wollen aber J2SDK, nicht J2JRE.
Entweder tauschen wir JRE durch J2SDK aus,
oder wir fügen einen weiteren Eintrag ein.

5. "`Installed JRE -> Add`"



6. "`Add -> OK`"
7. "`Installed JREs -> OK`"

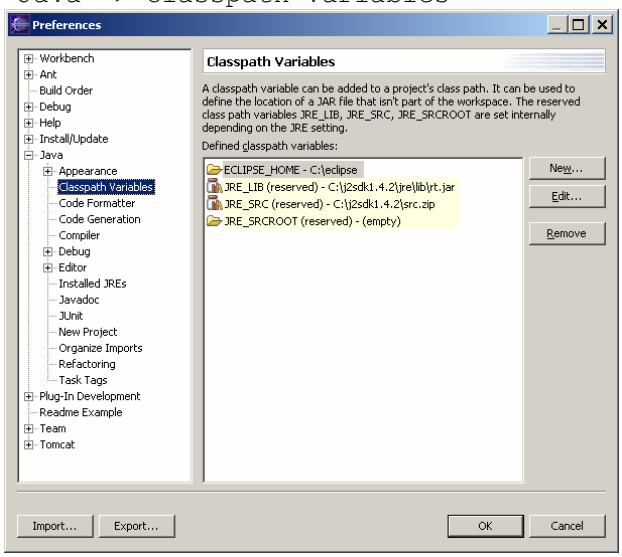
1.5.3. Klassenpfadvariablen / Bibliotheken definieren

Falls wir mehrere bestimmte Java Archive (Jar's) in mehreren Projekten verwenden, lohnt es sich in der Regel, diese unter einem Namen zusammenzufassen. Beispielsweise werden Sie beim Einsatz von JavaMail dauernd die 3-5 selben Jar's verwenden müssen.

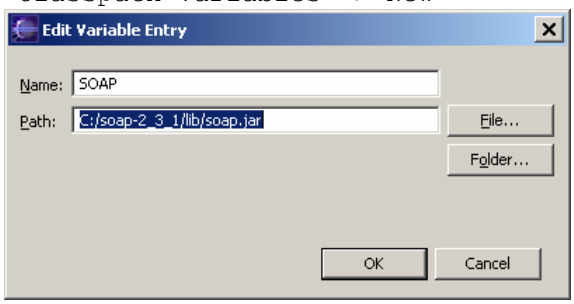
Im Sinne einer besseren Konfiguration von Eclipse können wir solche "Bibliotheken" im voraus definieren und immer wieder verwenden.

Vorgehen:

1. "`Window -> Preferences`"
2. "`Preferences -> Java`"
3. "`Java -> Classpath Variables`"



4. "`Classpath Variables -> New`"



Die Wahl "`Folder`" oder "`File`" ist sehr praktisch, speziell falls sich mehrere Jar's und Class Dateien in *einem* Verzeichnis befinden.

5. "`New -> File / Folder`"
6. "`File/Folder -> OK`"

Sie können Libraries oder besser Klassenpfadvariablen in Ihr Projekt einbinden

1) indem Sie diese Variable beim Einrichten des (neuen) Projekts eingeben.
2) Indem Sie im linken Fenster auf das Projekt klicken, rechte Maustaste -> "`Properties`" und dann den *Java Buildpath* anpassen.

1.5.4. Workspace

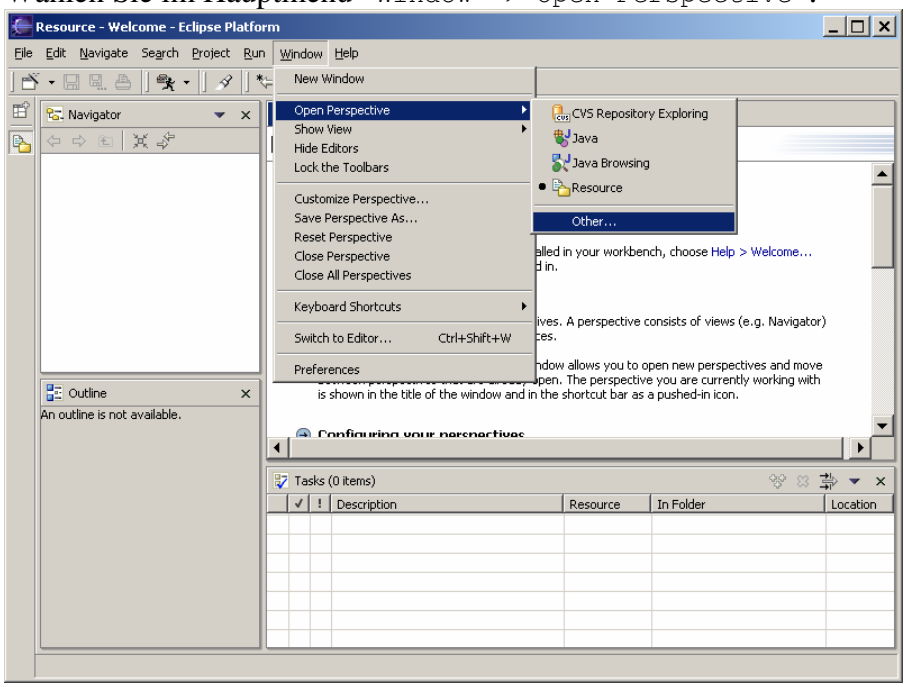
Die Grundeinstellung sieht als Wurzelverzeichnis für den Workspace das Verzeichnis `%ECLIPSE_HOME%/workspace` vor. Gerade wenn mehrere Benutzer auf die Installation zugreifen wollen oder sich einen Rechner teilen, ist eine Trennung der Arbeitsbereiche sinnvoll. Man kann beim Starten von Eclipse mit der Option `-data` ein beliebiges Verzeichnis zum Workspace-Wurzelverzeichnis machen, zum Beispiel `eclipse -data c:\MyWorkspace`.

1.5.5. Perspektive

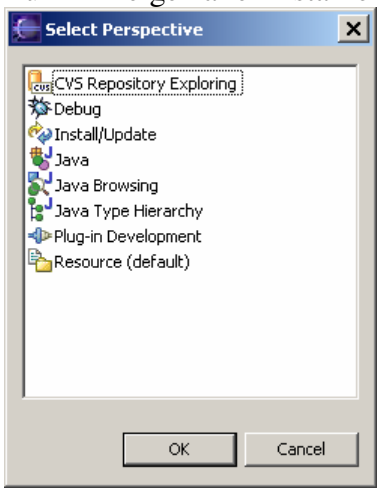
Bisher haben wir unser Workbench einfach eingesetzt, ohne irgend welche Fenster zu verschieben oder neue Fenster einzublenden – wir haben mit einer Standard Perspektive gearbeitet.

Jetzt wollen wir Perspektiven anpassen.
Um eine neue Perspektive in die Workbench zu laden, geht man folgendermassen vor:

1. Wählen Sie im Hauptmenü `'Window -> Open Perspective'`:



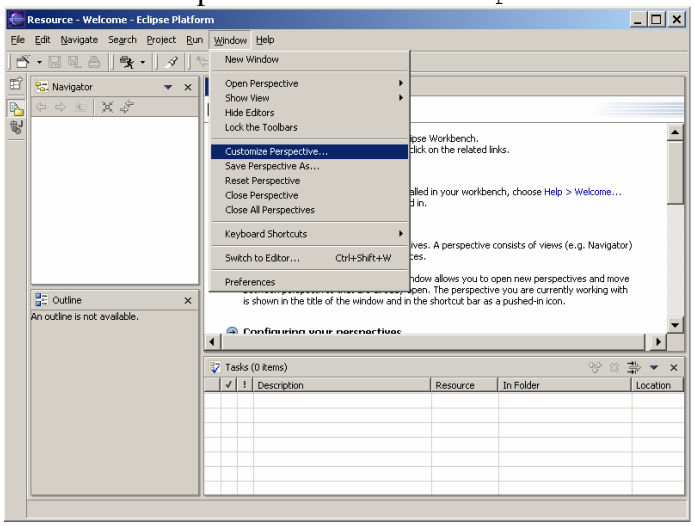
2. Zum Anzeigen aller installierter Perspektiven wählt man '`Other`':



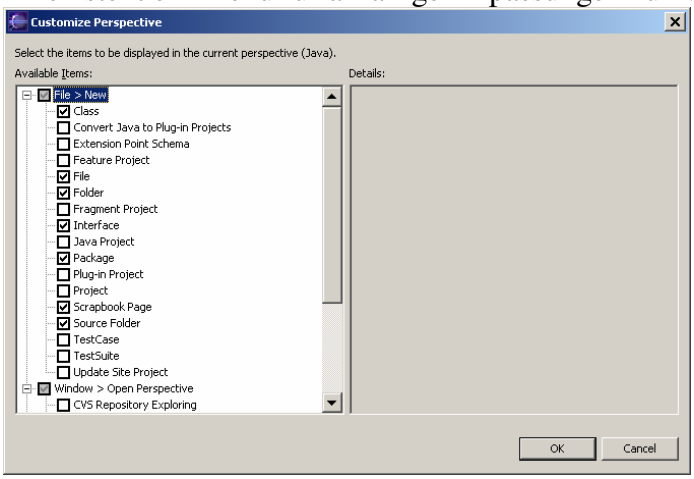
Falls Sie eine eigene Perspektive erstellen oder eine bestehende Perspektive anpassen wollen, selektieren Sie am besten zuerst jene Perspektive, welche in etwa Ihren Wünschen entspricht. Diese Perspektive können Sie dann anpassen und unter einem neuen Namen abspeichern.

Praktisches Vorgehen:

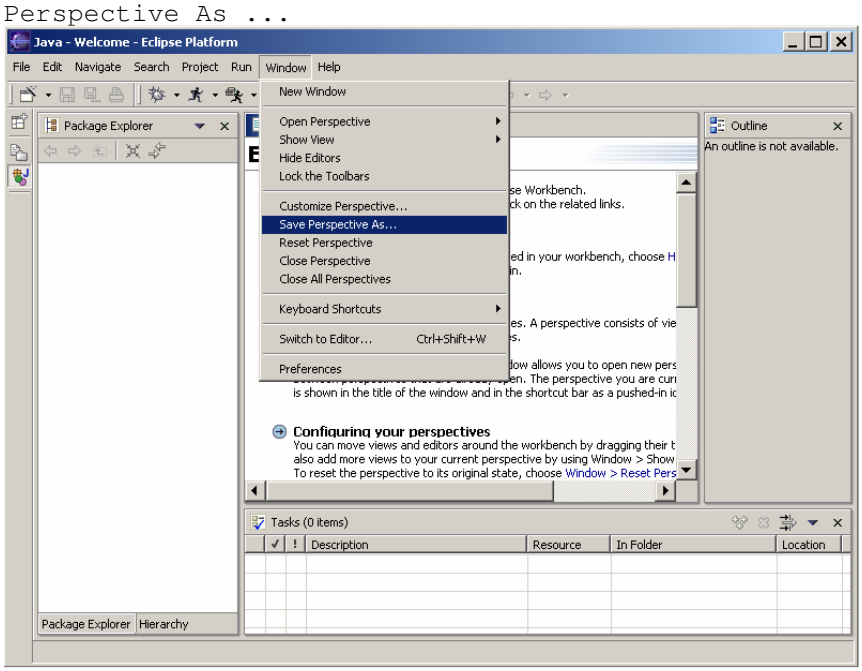
1) Selektieren einer Perspektive, wie oben
2) Auswahl der Option `Customize Perspective`:



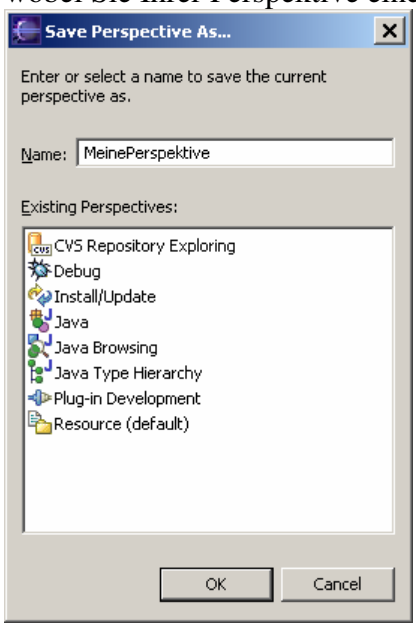
Ihnen steht ein Menü für allfällige Anpassungen zur Verfügung:



3) Nach Ihren Änderungen speichern Sie diese mit der Option "`Save Perspective As ...`



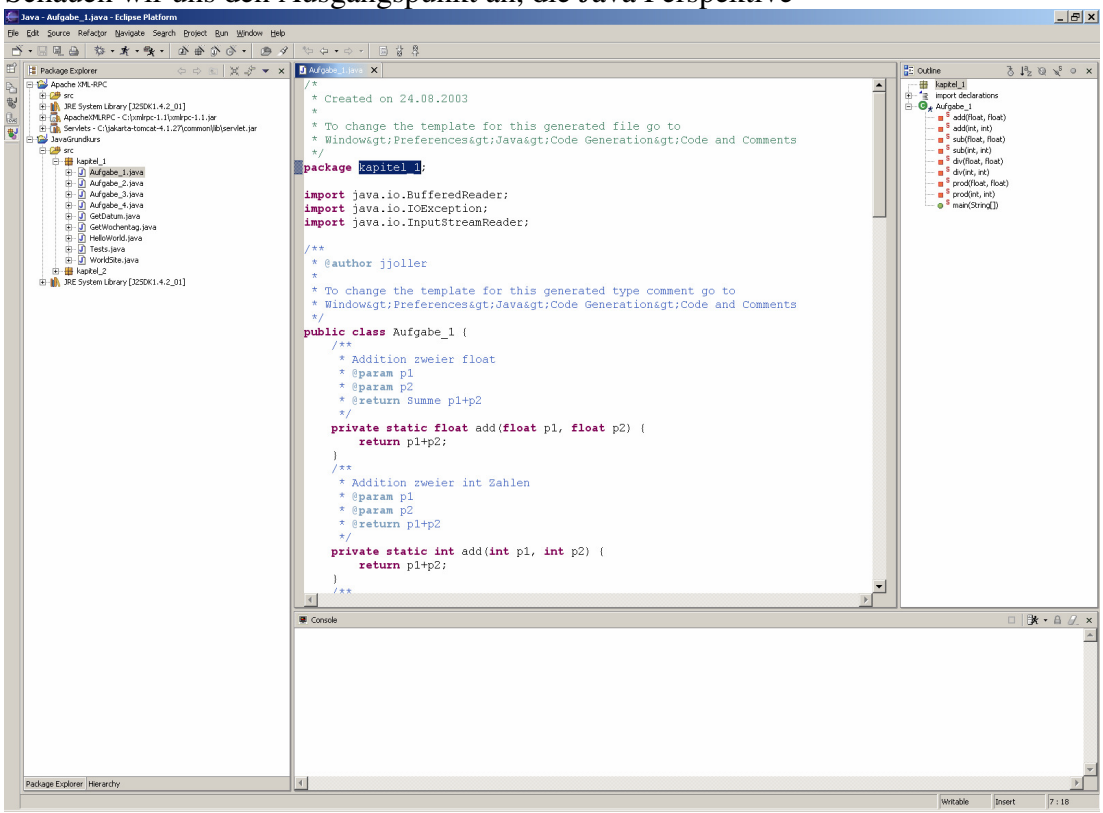
wobei Sie Ihrer Perspektive einen passenden Namen geben können:



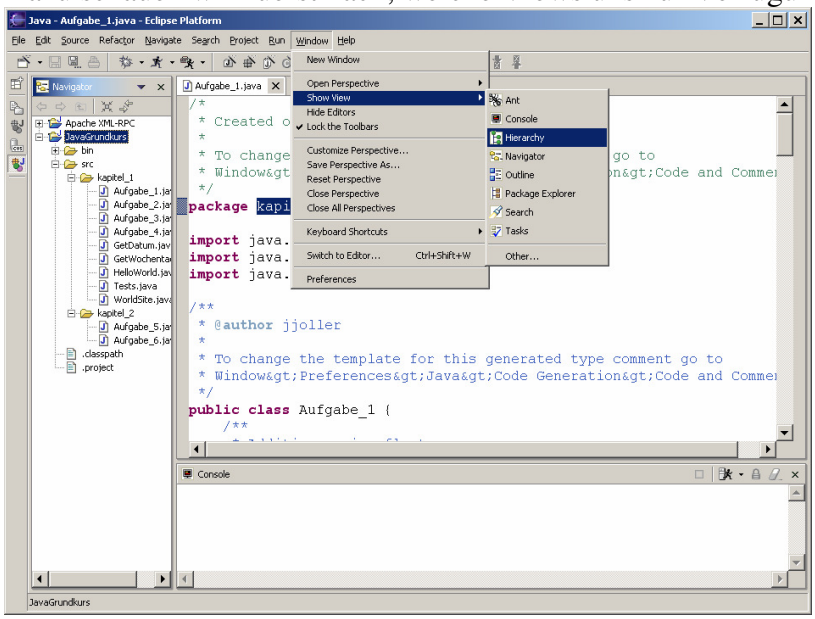
1.5.6. Views und Perspektiven

Eine Perspektive besteht aus mehreren Views. Sie können Views einer bestehenden Perspektive (die Sie evtl. selbst definiert haben) zuordnen.

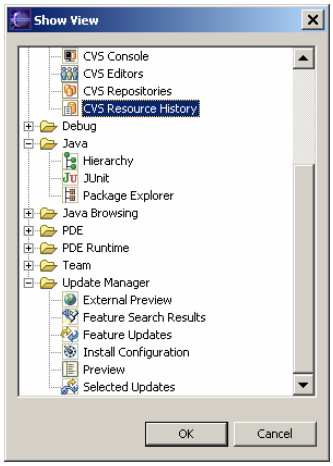
1. Schauen wir uns den Ausgangspunkt an, die Java Perspektive

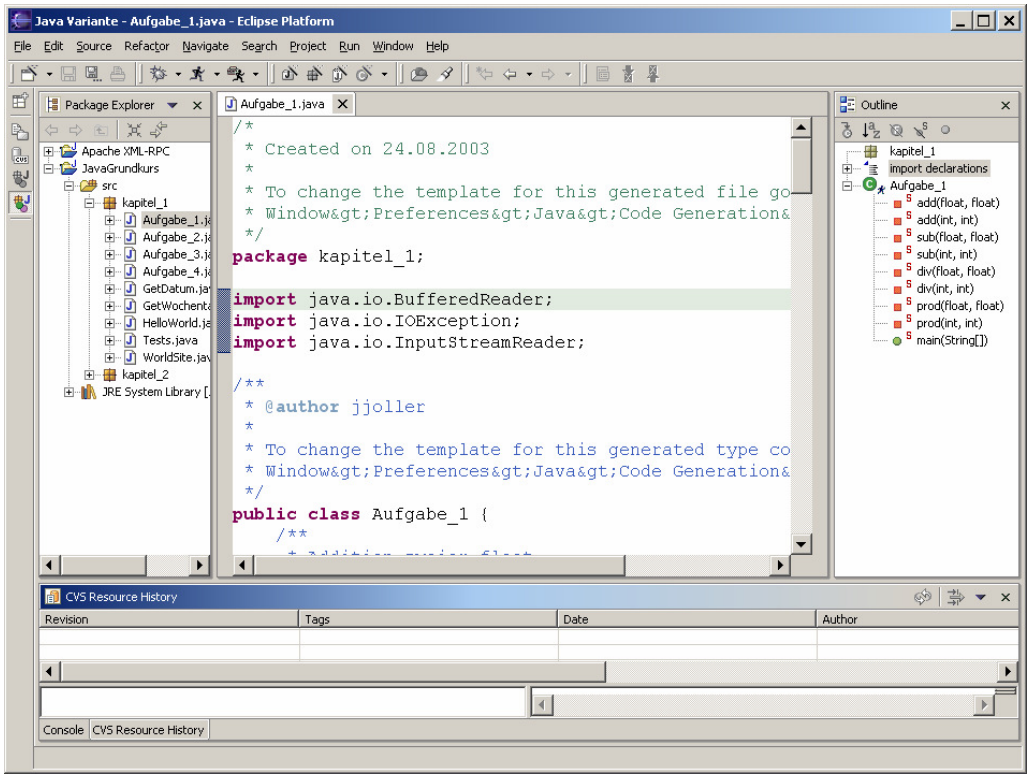


2. Nun schauen wir uns an, welche Views in unserer Perspektive noch sinnvoll wären. Dazu schauen wir zuerst nach, welche Views uns zur Verfügung stehen:



3. Wähle im Menü den Eintrag '`Show View -> CVS Resource History`':

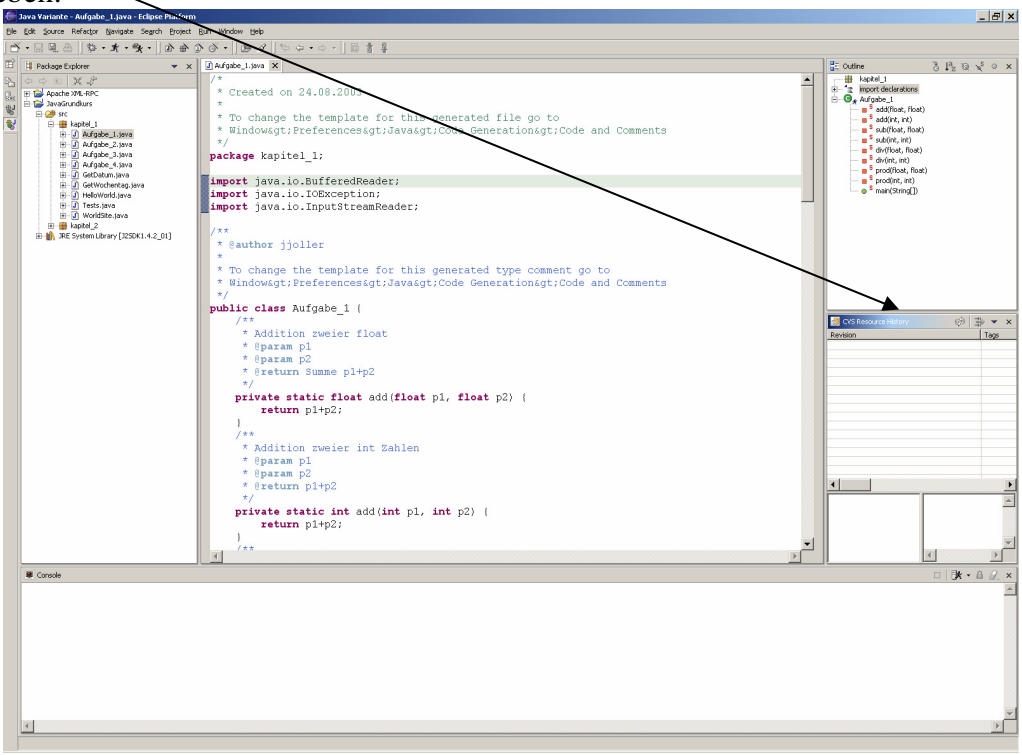




4. Nun möchten Sie sicher das Layout anpassen.
   Falls Sie lediglich die Grösse der einzelnen Fenster ändern möchten, können Sie dies, indem Sie das Fenster (oben links) anwählen und im Popup "`Size`" anwählen. Sie können dann je nachdem wo sich das Fenster befindet, links, oben, unten oder rechts die Seite neu setzen (die entsprechende Kante erscheint fett).

1.5.7. Ändern der Anordnung in der Workbench

Sie können aber auch das ganze Fenster verschieben: in diesem Fall erscheinen je nachdem wo Sie die View hin verschieben, ein schwarzer Pfeil oder ein Tab Symbol. Damit können Sie beispielsweise die CVS Sicht vom unteren Ende an die rechte Seite verschieben:

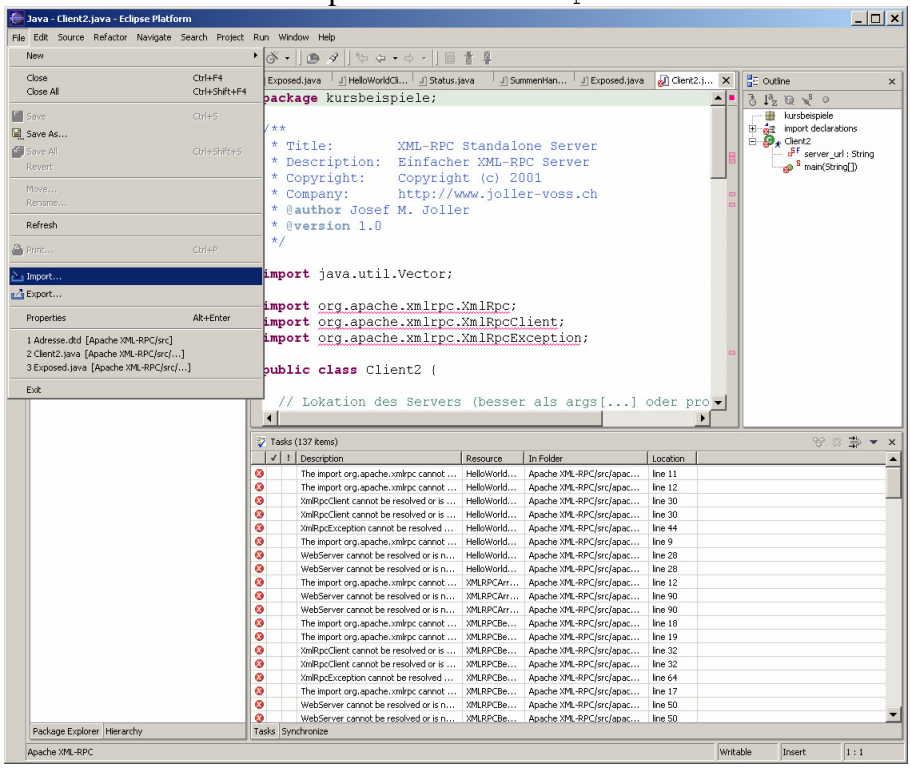


Sie müssen etwas üben, bis Sie mit der Technik voll zurecht kommen:
Sie selektieren die View (anklicken) und verschieben diese dann (Maus-Taste gedrückt halten).

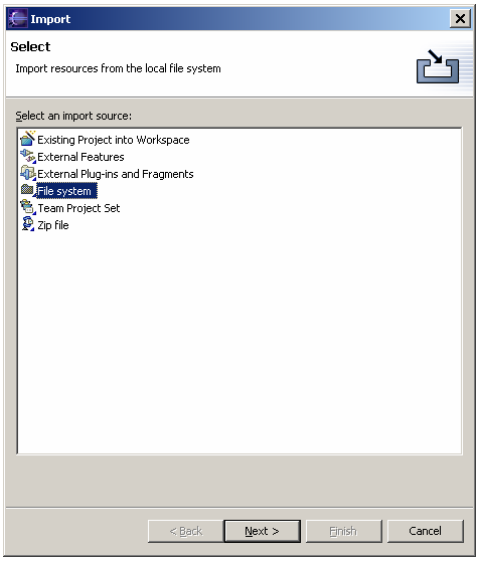
1.6. Import eines bestehenden Projekts

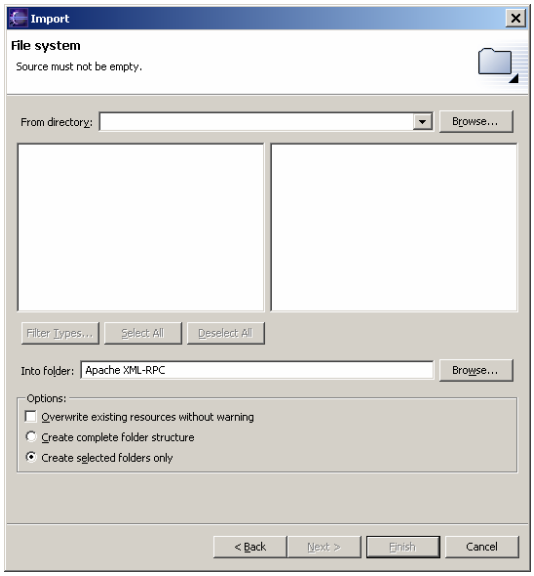
Da ich bisher fast alles mit dem JBuilder entwickelt habe, benötige ich irgend eine Möglichkeit diese Projekte in Eclipse zu importieren. Das funktioniert recht einfach:

1. Zuerst erstellen wir ein neues Java Projekt mit dem gewünschten Namen.
2. Dann wählen wir im Hauptmenü "`File->Import`"



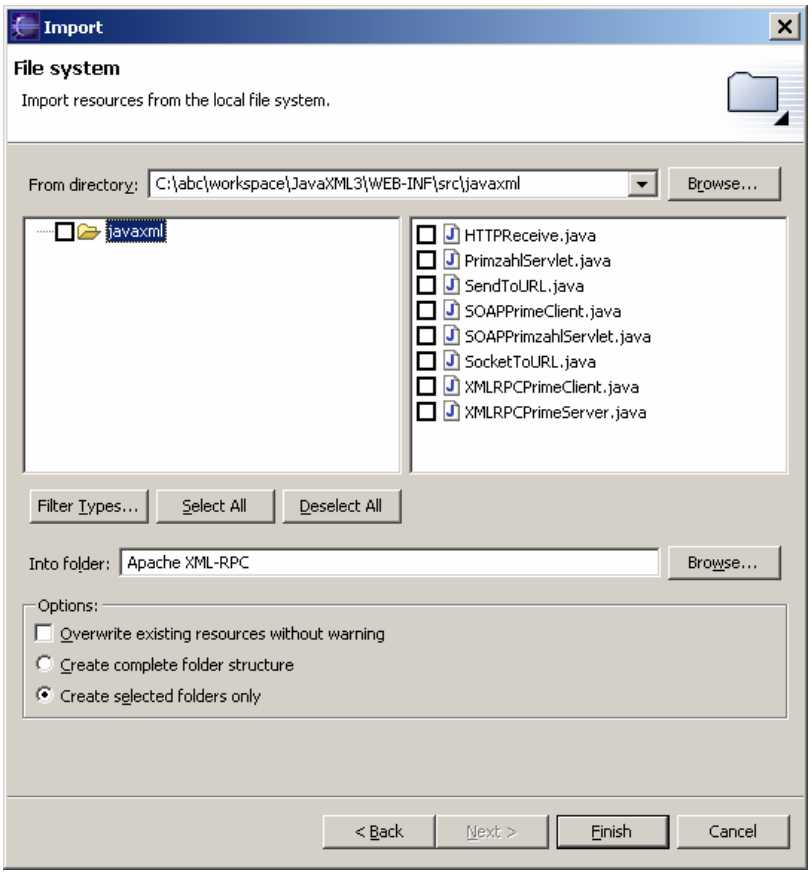
3. Es erscheint das Auswahlmenü. Wir wählen "`File->Import`"





Oben geben wir das Verzeichnis an und unten sehen Sie das Package (als Verzeichnis im Workspace).

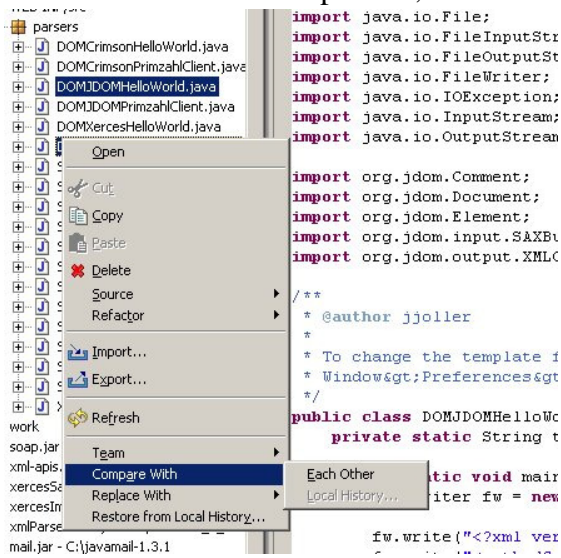
Dann erscheint ein Auswahlmenü



Hier können wir die zu importierenden Dateien (auch nicht Java Dateien) auswählen.
Falls die Dateien im falschen Package sind, können Sie diese einfach mit Drag and Drop ins korrekte Verzeichnis verschieben. Der Package Namen wird dabei angepasst.

1.7. Vergleich zweier Dateien

Wenn Sie eine Datei importiert, aber bereits eine eigene Version in Arbeit haben, können Sie die beiden Dateien miteinander vergleichen:



In der Package Exporer View selektieren Sie die zu vergleichenden Dateien.

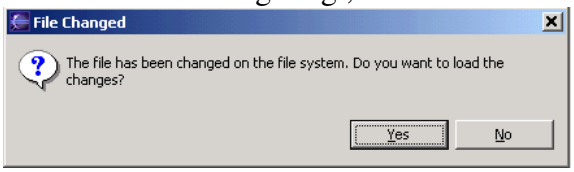
Klicken Sie auf die rechte Maustaste und wählen Sie „`Compare With`“ -> „`Each Other`“.

Damit können Sie die zwei Dateien vergleichen.

1.8. Externes Manipulieren von Ressourcen und Projekten

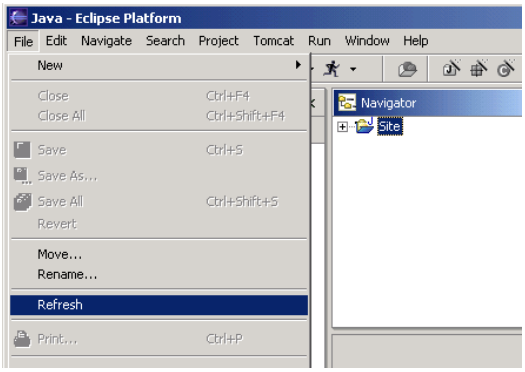
Verändert man Ressourcen im Projektverzeichnis innerhalb des Eclipse IDEs, werden die einzelnen beteiligten Komponenten automatisch über Callbacks informiert. Wenn Sie Ressourcen oder Projekte mit externen Werkzeugen verändern oder dem Projektverzeichnis neue Dateien hinzu fügt, gibt es natürlich keinen Benachrichtigungsmechanismus.

Sobald Sie extern veränderte Ressourcen in einem Editor öffnen, wird man beim nächsten Editieren der Datei gefragt, ob man die Veränderung von der Festplatte laden möchte:



Erstellt man neue Dateien oder Verzeichnisse in einem Projektverzeichnis (ausserhalb von Eclipse), muss Eclipse mitgeteilt werden, dass dies geschehen ist:

- Selektieren Sie die View, in der sich die Veränderungen ausgewirkt haben sollten und selektieren Sie "`File->Refersh`".



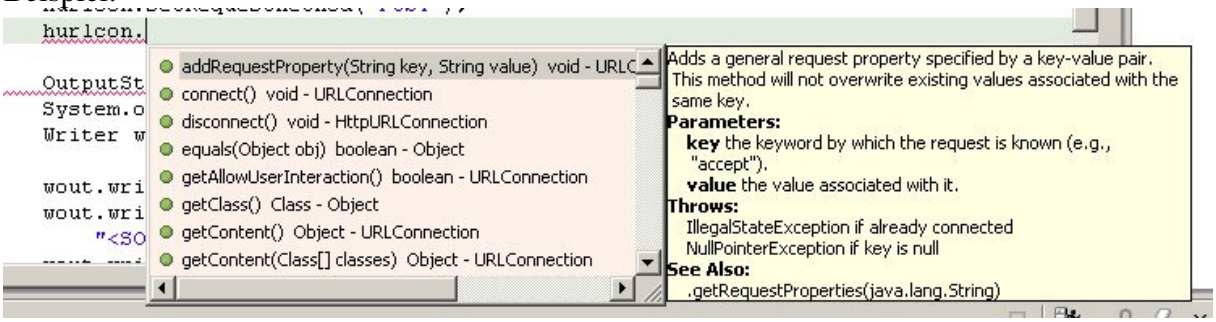
Jetzt sind die Ressourcen wieder synchronisiert.

1.9. Shortkeys

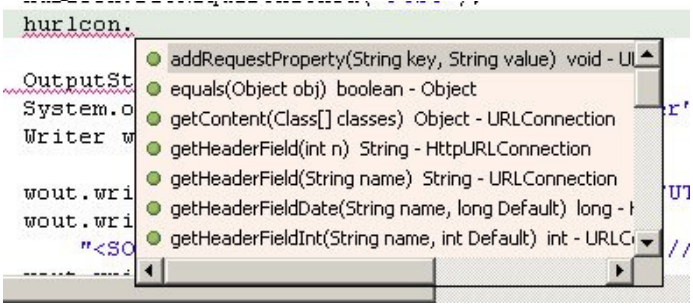
Eclipse besitzt einige sehr nützliche vordefinierte Key-Kombinationen:

1.9.1. Hilfe und Tipps

CTRL-Space Content-Assist
Beispiel:

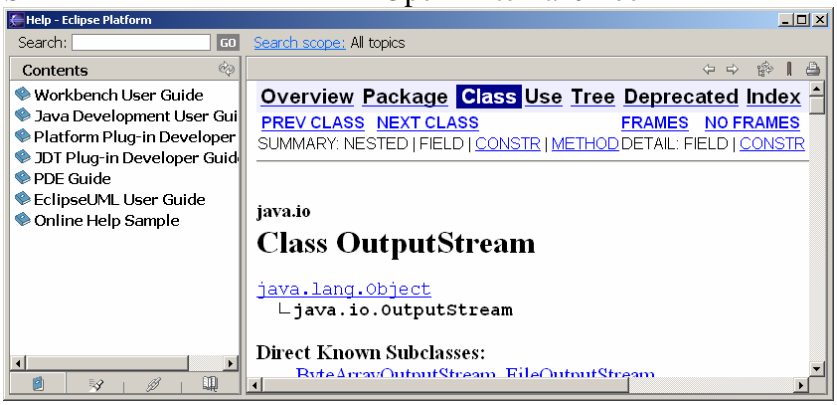


CTRL-SHIFT-Space Parameter-Hinweise
Beispiel:



CTRL-1 Quick-Fix

F2 Tooltip-Description
SHIFT-F2 Open External JDoc

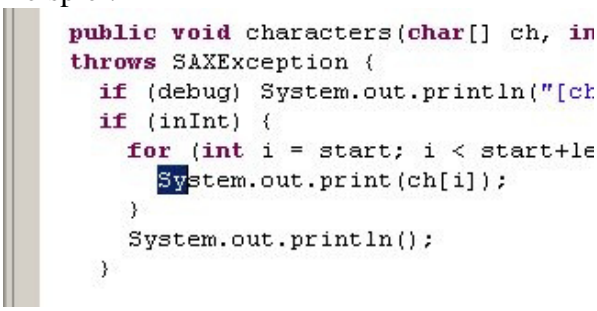


Falls dies nicht gelingt, könnte es sein, dass der Pfad auf die Java API Dokumentation falsch ist. Sie können diese unter „Window->Preferences->InstalledJRE-><auswählen des passenden JDK's->Edit->JavaDoc URL: file:/C:/j2sdk1.4.2_03/docs/api/“

Falls Sie die JavaDoc des Projekts einbinden wollen und per Shift-F2 verfügbar machen wollen, dann müssen Sie die JavaDoc Location im Projekt entsprechend angeben.

F3 Open-Declaration

Beispiel:



```
public void characters(char[] ch, in
throws SAXException {
  if (debug) System.out.println("[ch
  if (inInt) {
    for (int i = start; i < start+le
      System.out.print(ch[i]);
    }
    System.out.println();
  }
```

F3 auf System liefert den Source Code der betreffenden Klasse:

```
 * @author  Arthur van Hoff
 * @version 1.131, 01/29/03
 * @since   JDK1.0
 */
public final class System {

    /* First thing---register the natives */
    private static native void registerNatives();
    static {
        registerNatives();
    }

    /** Don't let anyone instantiate this class */
    private System() {
    }
```

F4 Type Hierarchy

1.9.2. Ausführung und Verwaltung

CTRL-F11	Run last launched
F11	Debug last launched
CTRL-F	Find & Replace
CTRL-S	Save
CTRL-F4	Close Window

1.9.3. Debugging

F5	Step Into
F6	Step Over
F8	Resume
CTRL-SHIFT-B	Toggle-Breakpoint

1.9.4. Formatierung, Code, Makros

CTRL-/ (ALT-S Return)	Einkommentieren
CTRL-\ (ALT-S Pfeil down)	Kommentare entfernen
CTRL-SHIFT-**F**	Formatiere Code
CTRL-SHIFT-**O**	Repariere Imports
sys[o,e] (CTRL-SPACE)	System.[out,err].println();