Einführung Eclipse RCP

Vortrag für die berlin.jar...
der Java-Konferenz 2008 in Berlin

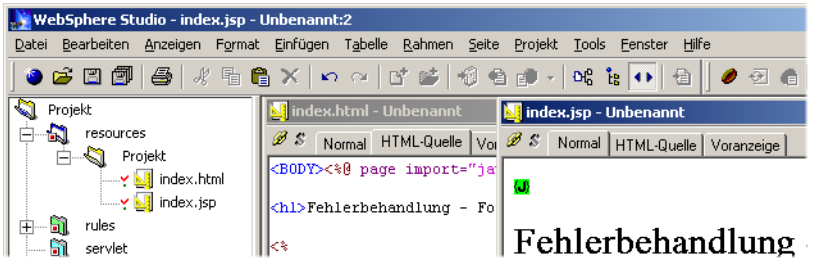
Wer spricht?

Manfred Borzechowski
Freier Java-Entwickler, Trainer und Berater

...und worüber?

- *Nicht* über Entwicklung *mit* Eclipse
- *Keine* step-by-step RCP instructions
- *Kaum* Code-Beispiele
- Sondern
  - Was Eclipse RCP *ist*, und
  - Was angehende Eclipse RCP-Entwickler *erwartet*:
    - „Extensions“
    - Neue GUI-Bibliotheken
    - Entscheidungen
    - Bauchschmerzen

Eine Geschichte voller „Spannungen“

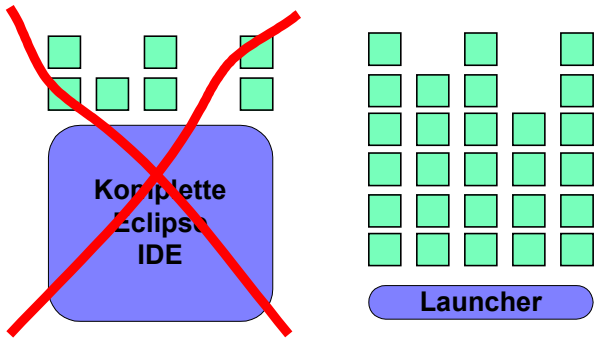


Es standen sich gegenüber:

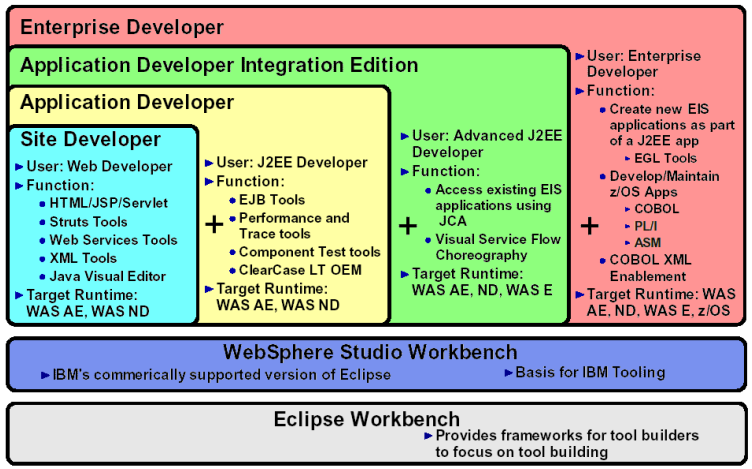
- VisualAge for Java
- Smalltalk & Repository
- IBM & WAS & J2EE
- Frei & Populär
- AWT & „Features“
- Fat

- WebSphere Studio
- ? & Files
- MS & .NET
- Proprietär & Lukrativ
- Swing & „Speed“
- Light

Die Lösung



- IDEs für Java *und* J2EE
  - Gemeinsame Basis durch geniale Modularisierung
- Frei *und* Lukrativ
  - Java-IDE frei: Eclipse
  - J2EE-IDEs für WAS kommerziell
- Speed *und* Features:
  - SWT – eine native UI-Bibliothek
- Fat *und* schnell *und* Light
  - Lazy loading, Update-Mechanismen
- Aber wohl kein J2EE *und* .NET



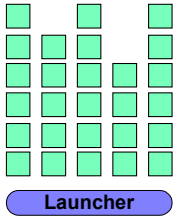
Plug-ins und Features

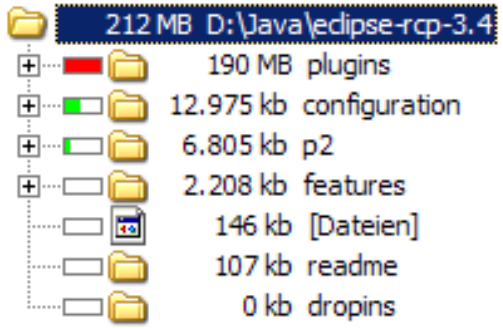
Die Module von Eclipse

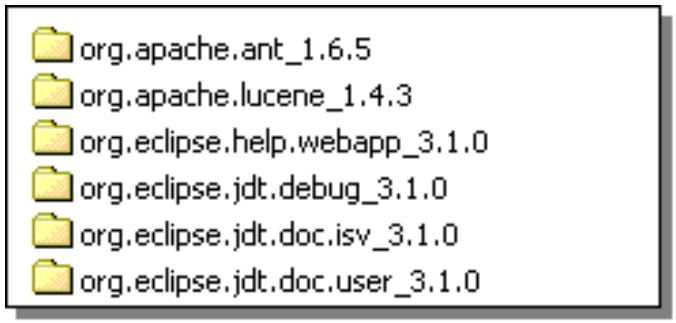
- „*Everything* is a Contribution“

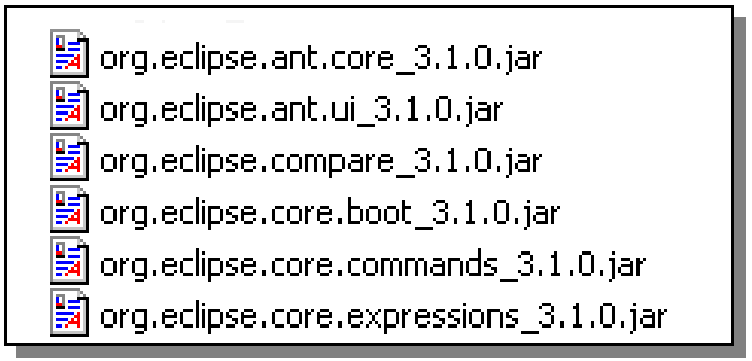
  Erich Gamma
  Pattern Papst

  - Weit mehr als Firefox Add-ons
  - Debugger, Compiler, Projektverwaltung, CVS,...
- Bis auf:
  - Launcher, Konfigurationsdateien
- Verzeichnisse und Jars im Verzeichnis *plugins*





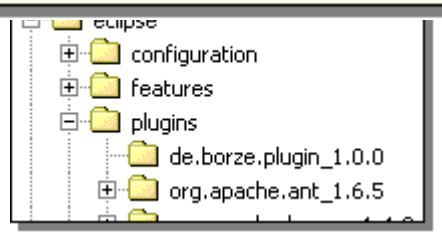


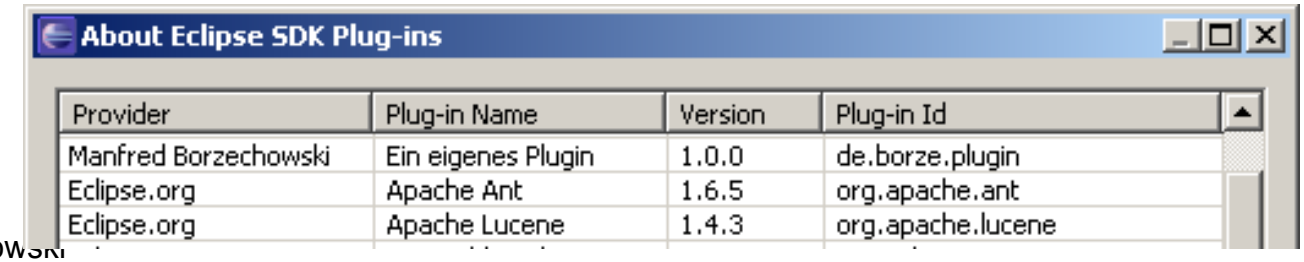


Was braucht man für ein minimales Plug-in?

- Datei plugin.xml
  - *Kein* Code
- In Unterverzeichnis plugins
  - Oder neuerdings auch *dropins*
- Start von Eclipse liest alle *plugin.xml*
  - Liste mit Menü **Help > About Eclipse Platform**, Knopf **Plug-in Details**

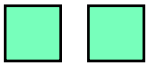
```
<?xml version="1.0" encoding="UTF-8"?>
<plugin
   name="Ein eigenes Plugin"
   id="de.borze.plugin1"
   version="1.0.0"
   provider-name="Manfred Borzechowski">
</plugin>
```





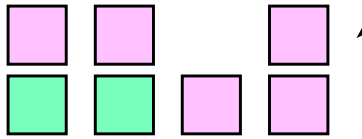
Zwei grundlegende Fakten!

- Was ist die RCP (Rich Client Platform)?
  - RCP ist ein rudimentäres Eclipse!
    - Nur die allernotwendigsten Infrastruktur-Plug-ins
      - Runtime, Workbench, SWT und JFace



Launcher

- Was bedeutet Entwicklung für RCP?
  - RCP-Entwicklung ist Entwicklung einer Anwendung durch Plug-ins für dieses rudimentäre Eclipse!
    - völlig unabhängig von der Java-IDE



Launcher

Warum sollte ich RCP-Projekte machen?

- Kunde braucht *Modularität*
  - Ausbaustufen und Erweiterbarkeit
- UI ist *reichhaltig*, *ansprechend* und *flott*
  - Perspektiven, MPEs, Views, Coolbar, Preferences, Update-Mechanismen
- Neue Kompetenz im Profil
  - Angebote im Gulp-Roboter: Swing 12, RCP 6
    http://www.gulp.de/robot/seek.html, Stand 2.9.2008
- Es gibt auch Gründe *dagegen*

OSGi – Bundles

- *Abhängigkeiten*: MANIFEST.MF definiert
  - Welche Versionen anderer Bundles verlangt sind
  - Welche eigenen Packages in anderen Bundles sichtbar sind

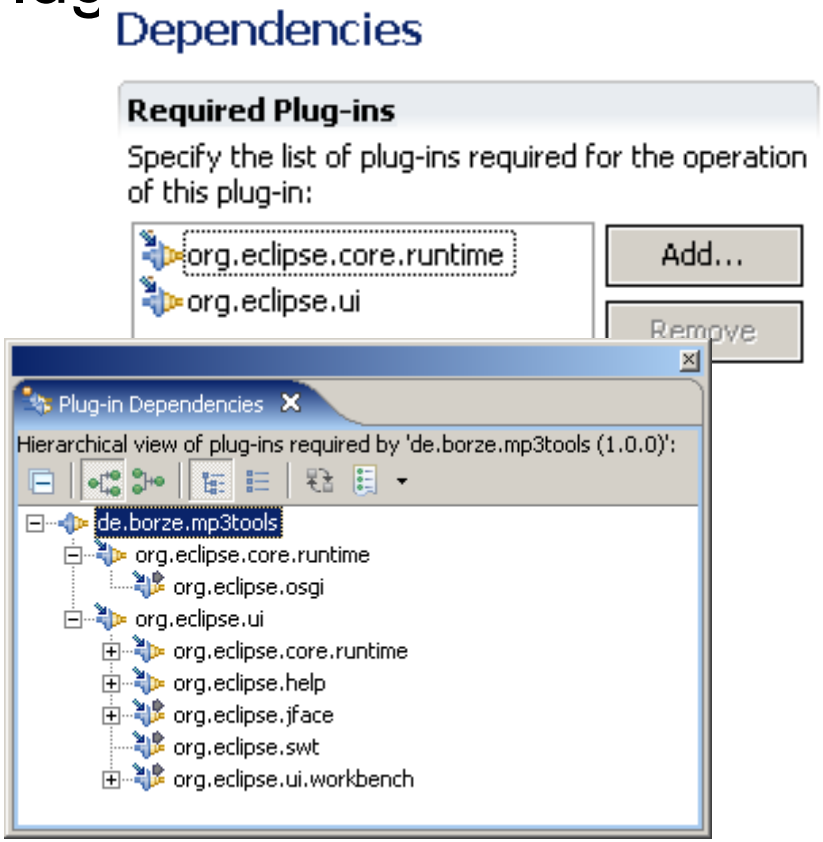
```
$ java -jar org.eclipse.osgi_3.3.2.R33x_v20080105.jar -console
osgi> ss

Framework is launched.

id      State       Bundle
0       ACTIVE      org.eclipse.osgi_3.3.2.R33x_v20080105

osgi> install file:///workspaces/osgi/tutorial/tutorialbundle.jar
Bundle id is 4

osgi> start 4
Apr 17, 2008 11:57:29 AM tutorial.TutorialActivator start
INFO: started
Apr 17, 2008 11:57:29 AM baselib.BaseService sayHello
INFO: Hello, world!
```

- *Lifecycle*:
  - Bundles können geladen, gestartet gestoppt und ausgetauscht
    - „Bei Registrierung erhalten Sie LifecycleEvents“
- *Nicht* durch JVM, sondern Container
  - Equinox, Felix, Knopflerfish

Plug-ins *sind* OSGi Bundles

- Haben also eine MANIFEST.MF
  - Nennt die zwei benötigten Plug-ins der RCP:

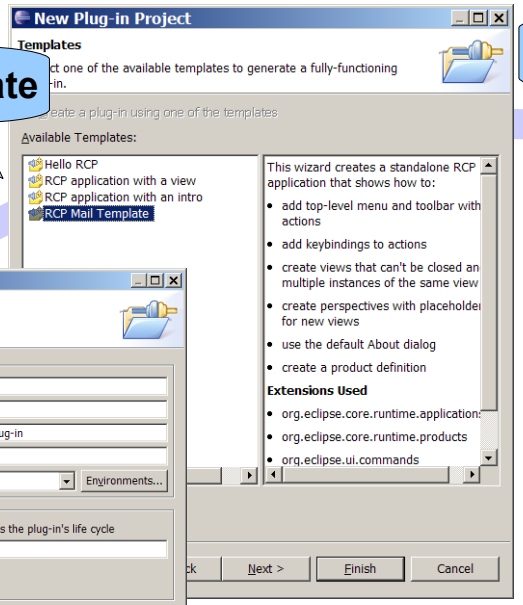
```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: MP3Tools Plug-in
Bundle-SymbolicName: de.borze.mp3tools; singleton:=true
Bundle-Version: 1.0.0
Bundle-Localization: plugin
Require-Bundle: org.eclipse.core.runtime,
 org.eclipse.ui
Eclipse-AutoStart: true
```

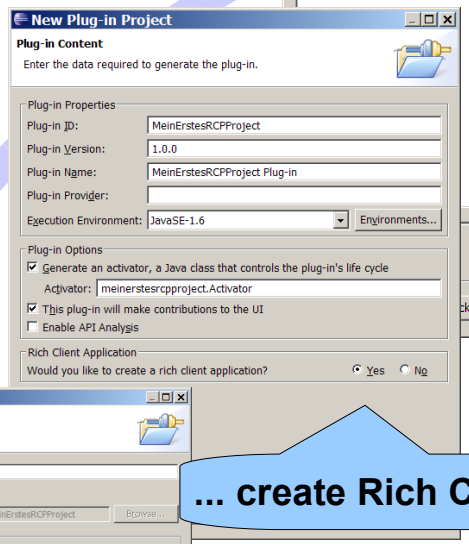


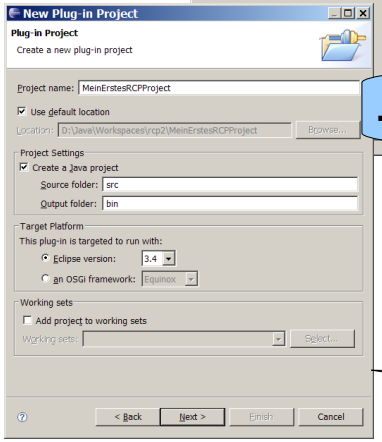
  - Die ihrerseits weitere Plug-ins benötigen
  - Warum nur verlangt das Crystal Report Plug-in alte Eclipse-Bundle-Versionen?

Ein RCP Template Projekt

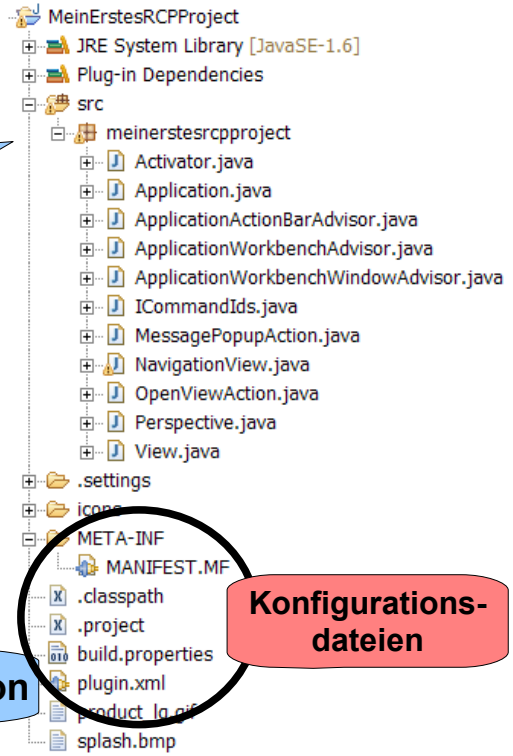
z.B. RCP Mail Template





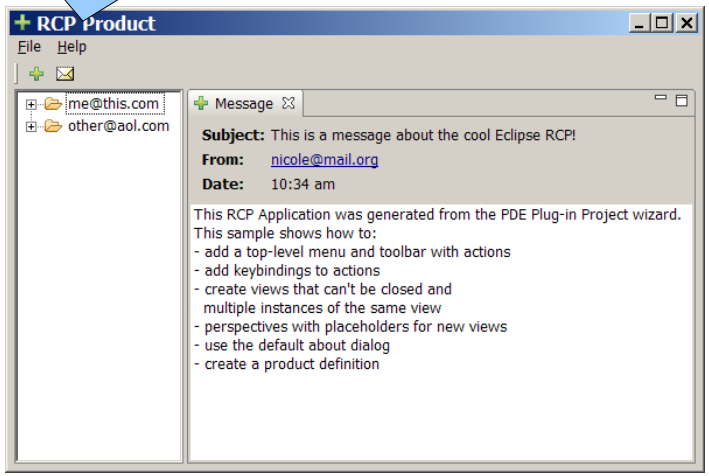


Generierte Projektinhalte



Konfigurations-dateien

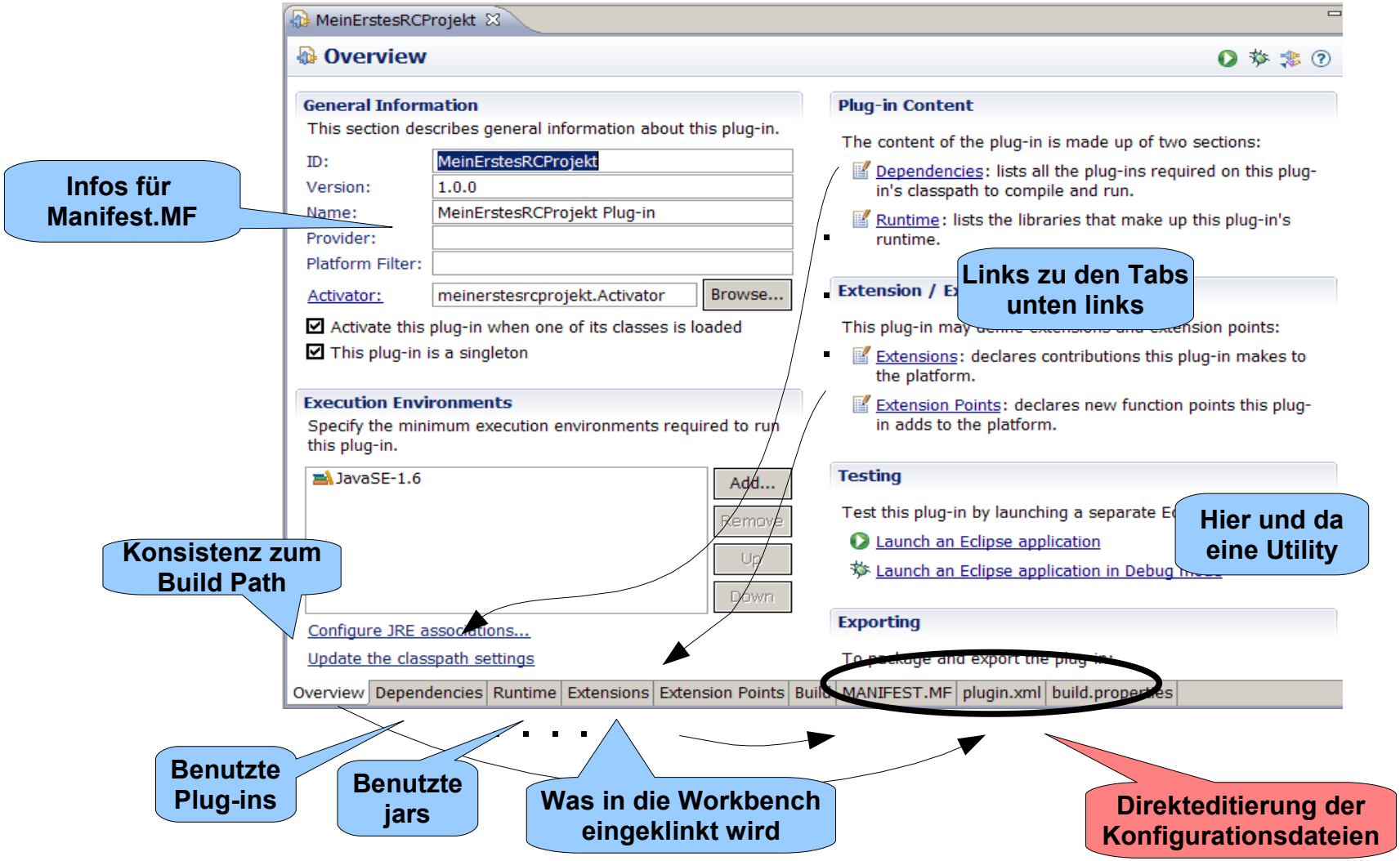
Run As > Eclipse Application



... create Rich Client Application

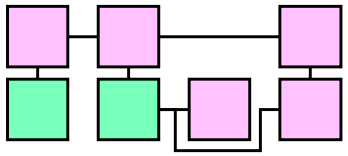
New > Project... > Plug-in Project

Konfiguration mit dem Plug-in Manifest Editor



I'm so lazy - but for a reason

- plugin.xml enthält Beschreibung aller durch die Workbench verwalteten GUI-Elemente
  - Menüs, Actions, Wizards, Preferences, Views, Editoren, Key Bindings, ...
  - Sogenannte Extensions



Launcher

- "Contributions are only loaded when they are needed"
  - Minimierung der Startup-Zeit

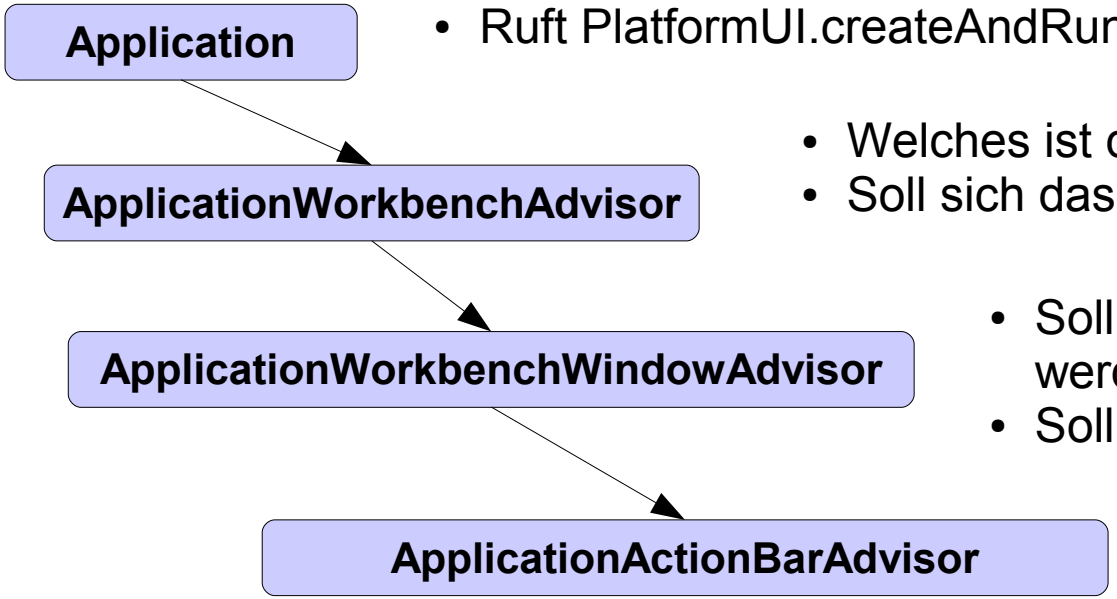
Erich Gamma
Pattern Papst

- Demo: Schmeiß den Kater 'raus

Extension #1:

Die Anwendung selbst

- Template erstellt beispielhafte Implementierung
  - Mit mehreren Konfigurationsobjekten



- Ruft PlatformUI.createAndRunWorkbench

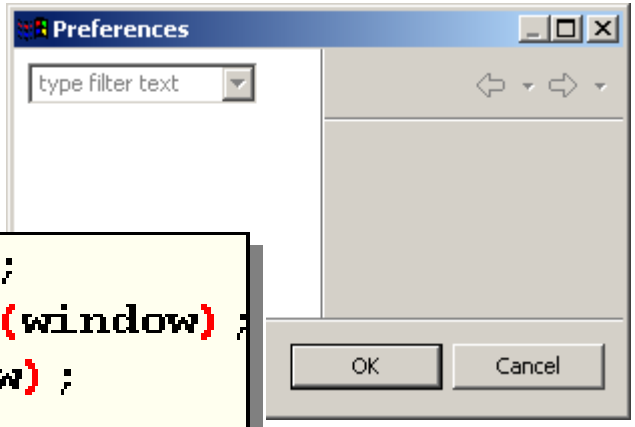
- Welches ist die ID der ersten Perspektive
- Soll sich das Fenster seine Position merken?

- Soll die Perspective Bar gezeigt werden?
- Soll es Statusbar und Coolbar geben?

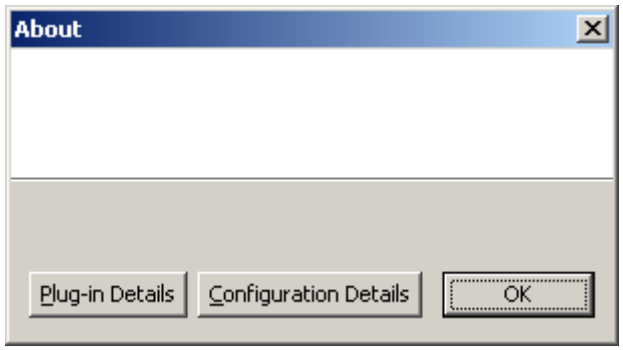
- Welche Actions sollen im Coolbar in jedem Falle vorhanden sein

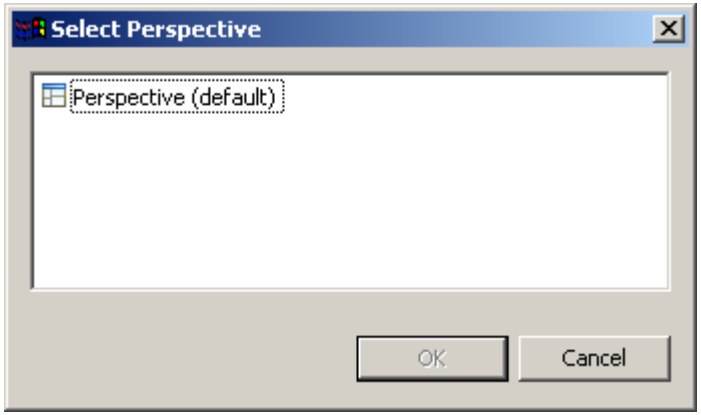
Geschenkt: Standard Actions

- Im ApplicationActionBarAdvisor
  - Erzeugen und registrieren von Instanzen von Standard-Actions




```
ActionFactory.OPEN_NEW_WINDOW.create(window);
ActionFactory.OPEN_PERSPECTIVE_DIALOG.create(window);
ActionFactory.RESET_PERSPECTIVE.create(window);
ActionFactory.HELP_CONTENTS.create(window);
ActionFactory.ABOUT.create(window);
ActionFactory.PREFERENCES.create(window);
```



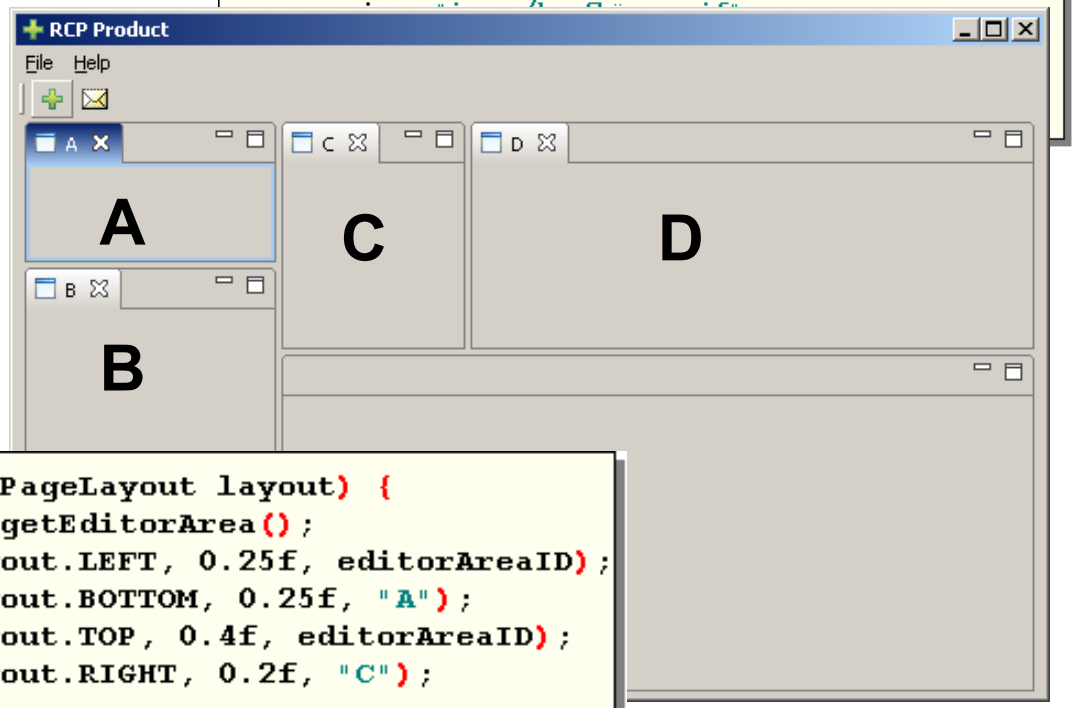


Extension #2:

Die erste Perspektive

- Layout der initial enthaltenen Views
  - Relativ zur *EditorArea*
  - Views *nur* über deren IDs
  - u.U. *nur* für ersten Aufruf

```
<extension point="org.eclipse.ui.perspectives">
   <perspective
         class="de.borze.mb3tools.BasisPerspective"
         icon="icons/note.gif"
         id="de.borze.mb3tools.BasisPerspective"
         name="Basis">
   </perspective>
   <perspective
         class="de.borze.mb3tools.VerwaltungPerspective"
```




```
public void createInitialLayout(IPageLayout layout) {
    String editorAreaID = layout.getEditorArea();
    layout.addView("A",  IPageLayout.LEFT, 0.25f, editorAreaID);
    layout.addView("B",  IPageLayout.BOTTOM, 0.25f, "A");
    layout.addView("C",  IPageLayout.TOP, 0.4f, editorAreaID);
    layout.addView("D",  IPageLayout.RIGHT, 0.2f, "C");
}
```

Extension #3

Views

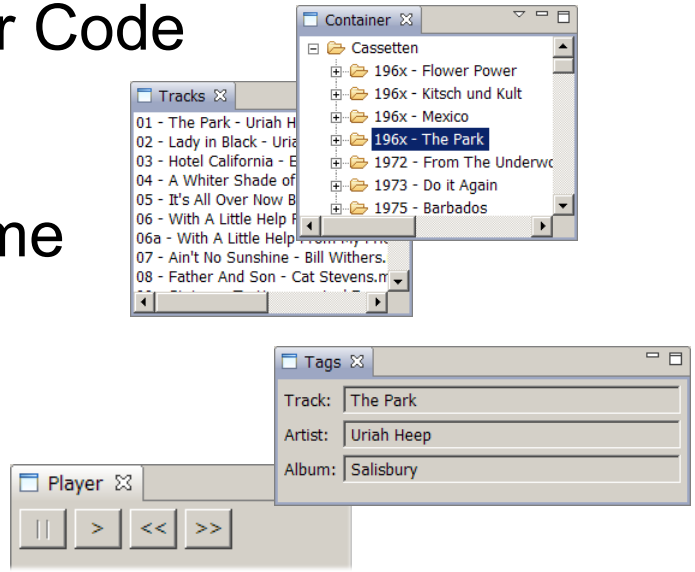
```
public class TracksView extends ViewPart

    @Override
    public void createPartControl(Composite parent) {
```

Behälter für SWT-GUI

- Für *Inhalt* sind sie *selbst verantwortlich*

Es *gibt* ein Common Navigator Framework, aber...

  - *Keine* Extension
    - sondern echter, handgearbeiteter Code
  - Einsatz von *JFace*
    - *Viewer* für Listen, Tabellen, Bäume
      Editoren *nicht* in Views
  - Einsatz von *SWT*
    - Masken, Controls, Graphiken



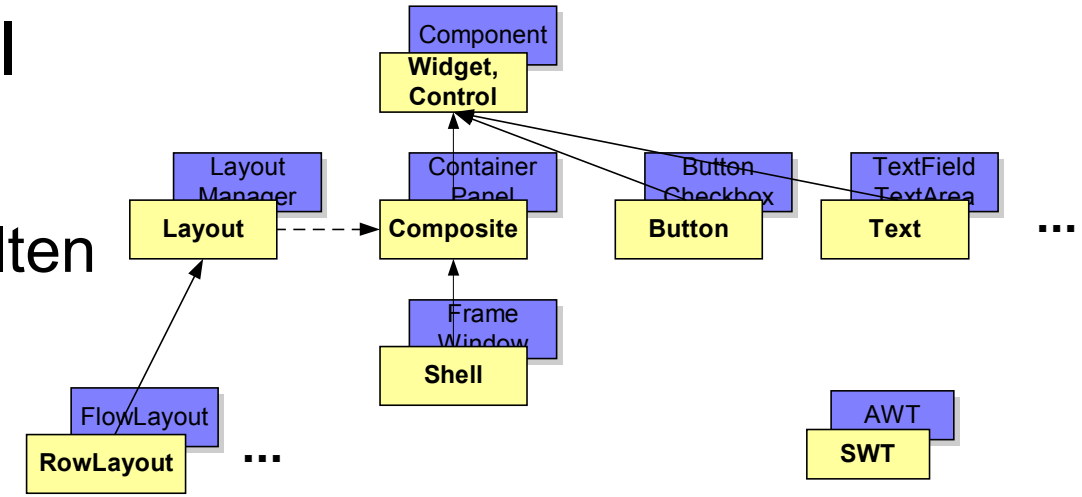
- *Aber:*
  - Menüs und Kommandos sind *separate* Extensions

SWT – Simple/Standard Widget Toolkit

- „AWT done right“

Dr. André Weinand
IBM, Eclipse Comitter

  - Dünner Layer über native Oberflächenkomponenten
  - Windows-Oriented API
    - Native Libraries
    - Natives I&F und Verhalten
    - Unflexibel



James Gosling
Sun, Schöpfer von Java

- „it pissed me off beyond words“

http://www.builderau.com.au/news/soa/James-Gosling-Q-A/0,339028227,339176462,00.htm

Was lernt man schon am ersten SWT-Beispiel?

```
import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.*;

public class ButtonDemo1 {

    public static void main(String[] args) {
        Display display = new Display();
        Shell shell = new Shell(display);
        shell.setText("ButtonDemo1");

        Button button = new Button(shell, SWT.PUSH);
        button.setText("OK");
        button.setSize(100, 25);
        button.setLocation(10, 10);

        shell.setSize(130, 100);
        shell.open();

        while (!shell.isDisposed()) {
            if (!display.readAndDispatch())
                display.sleep();
        }
        display.dispose();
    }
}
```

Keine Beans

Style Bits :-(

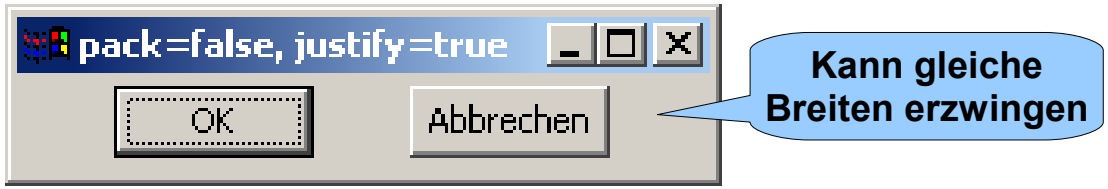
Kein separater Thread

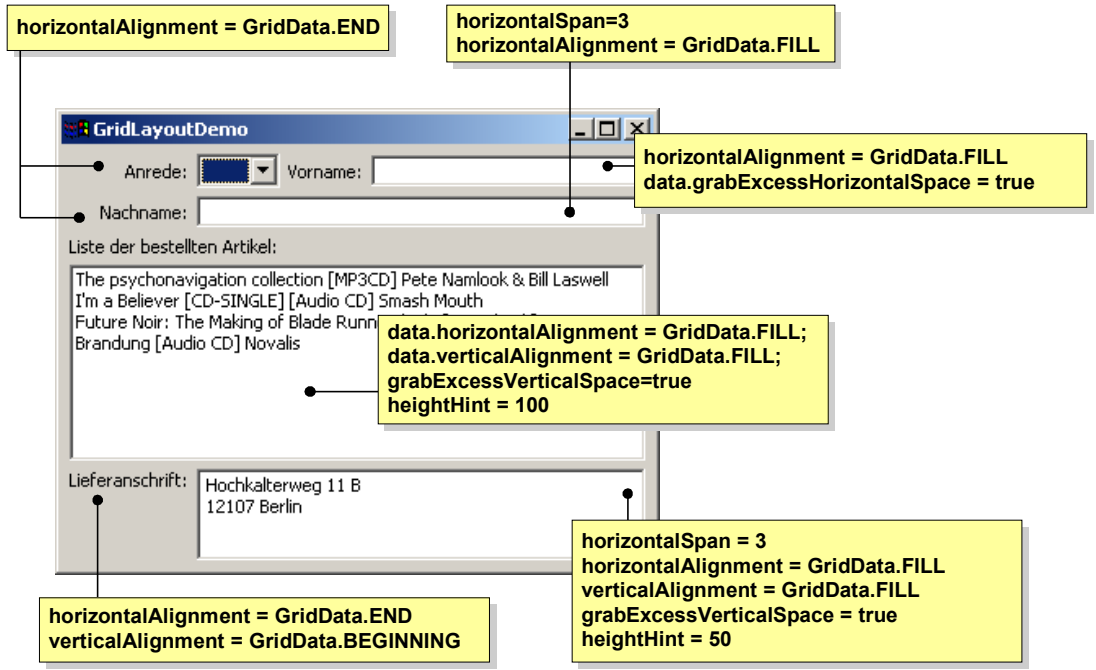
Ressourcen müssen disposed werden

Andere Styles

SWT Layouts

- *RowLayout*
  - ähnlich FlowLayout
- *FillLayout*
  - ähnlich AWT GridLayout
- *GridLayout*
  - ähnlich AWT GridBagLayout
- *FormLayout*
  - ähnlich Swing SpringLayout





SWT Listener – Auswahl

- Untypisiert
  - Auswahl des interessierenden Events durch Konstante
    - Von SWT :-(
- Typisiert
  - VerifyListener
    - Texteingaben widersprechen
  - HelpListener
    - Bei F1
  - TraverListener,...

```
button.addListener(SWT.Selection, new Listener() {
    public void handleEvent(Event event) {
        System.out.println("handleEvent(" + event + ")");
    }
});
```

```
SWT.S
    SAVE : int - SWT
    SCROLL_LINE : int - SWT
    SCROLL_LOCK : int - SWT
    SCROLL_PAGE : int - SWT
    SEARCH : int - SWT
    SELECTED : int - SWT
    Selection : int - SWT
    SEPARATOR : int - SWT
    SetData : int - SWT
    Settings : int - SWT
    SHADOW_ETCHED_IN : int - SWT
```

Suchbild: Finde die Event-Konstante

```
Text input = new Text(shell, SWT.BORDER);
input.addVerifyListener(new VerifyListener() {
    public void verifyText(VerifyEvent vEvent) {
        String text = vEvent.text;
        vEvent.doit = text.matches("[0-9]*");
    }
});
```

Tu es – oder lass es sein!

SWT Ressourcen

Arial Black
Bitstream Vera Sans
Bitstream Vera Sans
Bitstream Vera Serif
Comic Sans MS

- *If you created it, you dispose it (if you didn't create it, don't dispose it).*
- *Disposing the parent disposes the children.*

- Umsetzungs-ratschläge:

```
Font font  = new Font (display, "Courier", 10, SWT.NORMAL);
GC gc = new GC(canvas);
gc.setFont(font);
gc.drawText...     // do some text drawing on the canvas using the GC
gc.dispose();
font.dispose();
```

  - http://www.eclipse.org/articles/swt-design-2/swt-design-2.html
- Handling *mehrfach* verwendeter Ressourcen:
  - ResourceManager
  - Registries

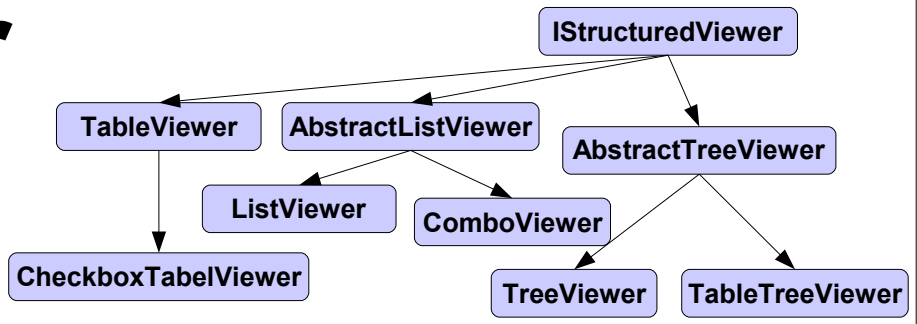
ResourceManager

Color	createColor(RGB descriptor) Allocates a color, given its RGB value.
Font	createFont(FontDescriptor descriptor) Returns the Font described by the given FontDescriptor.
Image	createImage(ImageDescriptor descriptor) Creates an image, given an image descriptor.

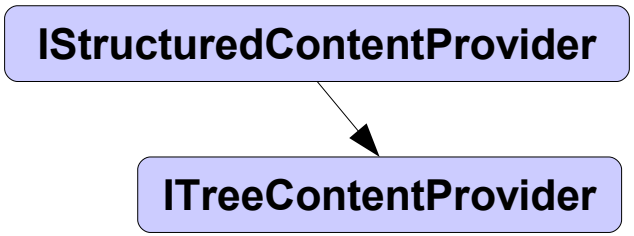
JFaceResources

static ColorRegistry	getColorRegistry() Returns the color registry for JFace itself.
static FontRegistry	getFontRegistry() Returns the font registry for JFace itself.
static ImageRegistry	getImageRegistry() Returns the image registry for JFace itself.

JFace Viewer und ContentProvider



- Grob vergleichbar den Models von Swing
  - IStructuredContentProvider
    - Gibt ein Array von anzuzeigenden Objekten
    - Verwendbar für Listen und Tabellen
      - Column-spezifisch durch LabelProvider



Method Summary

Object[]	**getElements**(Object inpu Returns the elements t

Method Summary

Object[]	**getChildren**(Object parentElement) Returns the child elements of the given parent ele
Object	**getParent**(Object element) Returns the parent for the given element, or null
boolean	**hasChildren**(Object element) Returns whether the given element has children.

  - ITreeContentProvider
    - Gibt zusätzlich Kinder und Eltern

JFace LabelProvider

- Grob vergleichbar mit den CellRenderern von Swing:
  - ILabelProvider
    - Gibt Text und Image für ein Objekt
    - Image nur in Tabelle sichtbar, nicht in Liste
    - Einfache Implementierung: LabelProvider
  - ITableLabelProvider
    - Gibt spaltenspezifischen Text und Image für ein Objekt

Image	**getImage**(Object element) Returns the image for the label of the given element.
String	**getText**(Object element) Returns the text for the label of the given element.

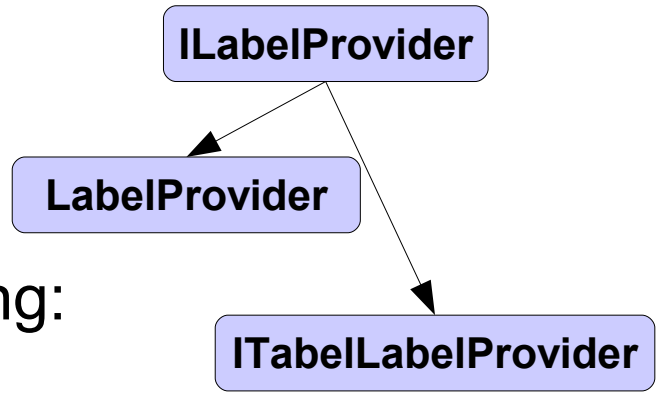
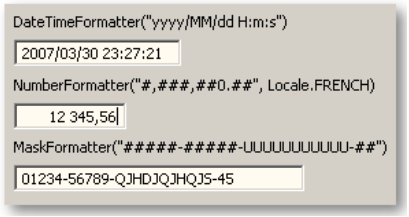


Image	**getColumnImage**(Object element, int Returns the label image for the given col
String	**getColumnText**(Object element, int c Returns the label text for the given colum

Neue Widgets im Nebel

- *Brutstätte* für neue SWT-Widgets
  - http://www.eclipse.org/nebula/



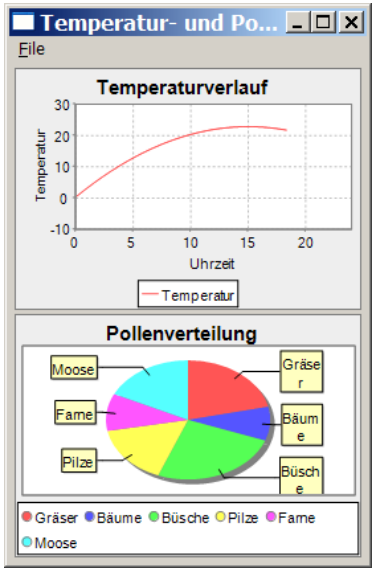
- Grid hat mich *nicht* begeistert
- *Alternativ*: http://sourceforge.net/projects/ktable
  - originäre SWT-Lösung
  - richtig brauchbar

...und ich etwas aus der anderen Welt brauche?

- Wenn Bibliotheken Swing-GUIs bieten
  - Einbinden mit *Embedded Frames*

```
frame = SWT_AWT.new_Frame(this);
```

    - *OpenOffice.org*
      - OOo kombinieren mit NOA – Nice Office Access
    - *JFreeChart*
      - Bietet selber (experimentellen) SWT-Support
    - *Crystal Reports*
  - Aber *nicht* unter OS X
  - OLE geht wohl auch, aber keine Erfahrung

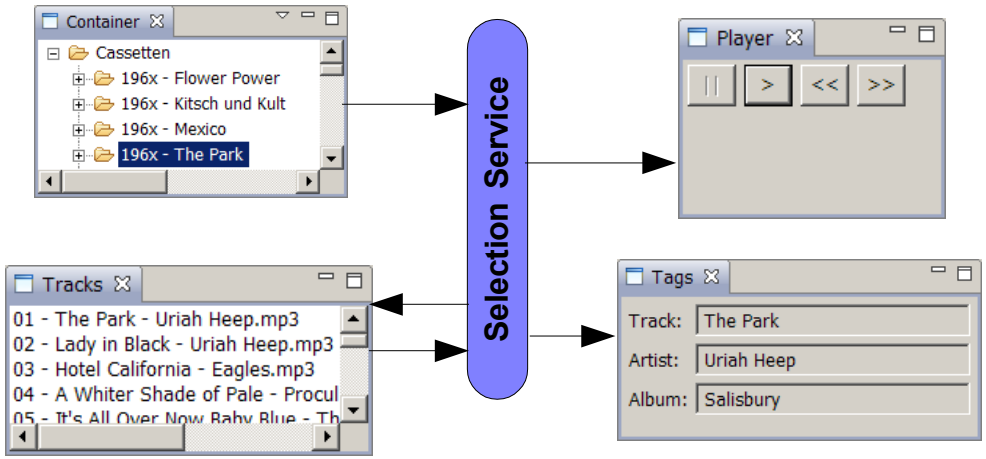


Views verdrahten

- *Lose* Kopplung bevorzugt
- Über SelectionService

```
getSite().setSelectionProvider(listViewer);
```

  - Viewer ist SelectionProvider
  - View ist Listener
    - IStructuredSelection
    - ITextSelection
    - ITreeSelection
    - IMarkSelection




```
// Einklinken als Listener in den SelectionService
ISelectionService service = (ISelectionService) getSite().getService(ISelectionService.class);
service.addSelectionListener("rcpTunes.views.tracks", this);
```

- http://www.eclipse.org/articles/Article-WorkbenchSelections/article.html

Extensions #4 bis #6: Command, Handler, Binding

- *Command*
  - Möglicherweise *mehrere* Handler
    - Aber nur ein aktiver wird angesprochen
  - *Kein* Code
- *Handler*
  - *Konkrete* Klasse für Command mit
    - execute-Methode
    - enabled-Property
    - Zustandslistenern
  - Zugriff auf *HandlerUtil*
- *Binding*
  - Optionale Tastenkombination für Command

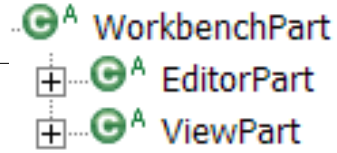
Extension #7: Menu

- Sichtbare Einbettung eines Commands
  - In Views oder Workbench Window
    - Dropdown Menü,
    - Toolbar,
    - Popup Menü
  - Konfiguriert durch *Location URI*
  - Eventuelle Bedingung durch `<visibleWhen`
- View kann durch *andere* Plug-ins Einträge erhalten
  - Ort definiert durch „spezielle“ Einträge

Ersatz für:
actionSets
editorActions
popupMenus
viewActions

```
<menuContribution
    locationURI="popup:rcpTunes.views.container">

<menuContribution
    locationURI="menu:org.eclipse.ui.main.menu?after=additions">
```

```
menuManager.add(new GroupMarker(IWorkbenchActionConstants.MB_ADDITIONS));
```



Extension #8: Editor

- Angezeigter Inhalt wird wie View erstellt

```
public void createPartControl(Composite parentComposite) {
```

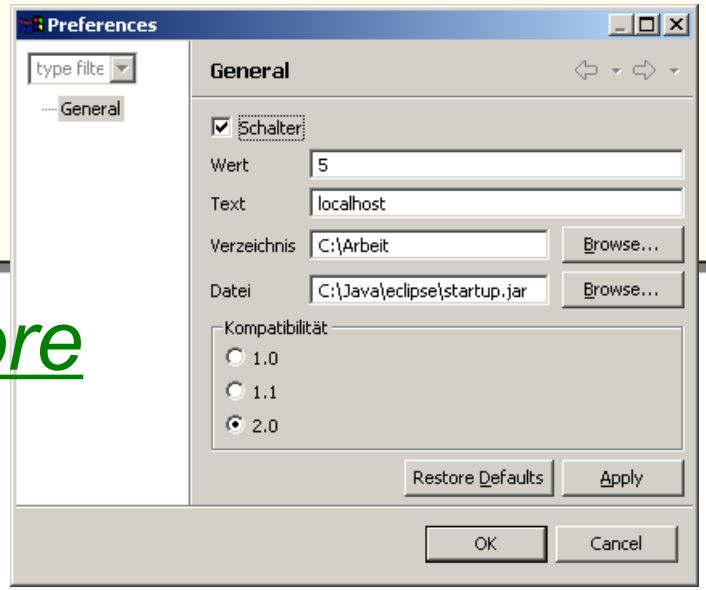
- *Dirty*-Zustand und *Save*-Metaphor
- Kann in mehreren Instanzen geöffnet werden
  - Jeweils für *IEditorInput*
- *Kein* Notwendigkeit daß
  - es sich um *Datei* handelt
  - irgendein *textueller* Inhalt dargestellt und bearbeitet wird

`exists()` Returns whether the editor input exists.
`getImageDescriptor()` Returns the image descriptor for this input.
`getName()` Returns the name of this editor input for display purposes.
`getPersistable()` Returns an object that can be used to save the state of this editor input.
`getToolTipText()` Returns the tool tip text for this editor input.

Extension #9: Preference Page

- Von FieldEditorPreferencePage ableiten
- Editoren mit Label hinzufügen

```
addField(new BooleanFieldEditor(SCHALTER, "Schalter", getFieldEditorParent()));
addField(new IntegerFieldEditor(WERT, "Wert", getFieldEditorParent()));
addField(new StringFieldEditor(TEXT, "Text", getFieldEditorParent()));
addField(new DirectoryFieldEditor(VERZEICHNIS, "Verzeichnis", getFieldEditorParent()));
addField(new FileFieldEditor(DATEI, "Datei", getFieldEditorParent()));
addField(new RadioGroupFieldEditor(KOMPATIBILITÄT, "Kompatibilität", 1,
        new String[][] {
            {"1.0", "1.0"},
            {"1.1", "1.1"},
            {"2.0", "2.0"},
        },
        getFieldEditorParent(),
        true));
```



- Außerdem: *PreferenceStore*
  - Vorinitialisieren
  - Auslesen

Extension #10: Product

- Sammelsurium von Properties
  - Zum Gebrauch innerhalb der Anwendung

```
windowImages
aboutImage
aboutText
appName
welcomePage
```

- In *separater* .product-Konfigurationsdatei
  - Splash-Screen
  - Optionen für Linux, Solaris, MacOS X, Windows
  - Launcher-Name
  - Icons
- Nötig für den Produkt*export*

Extension #11: Help

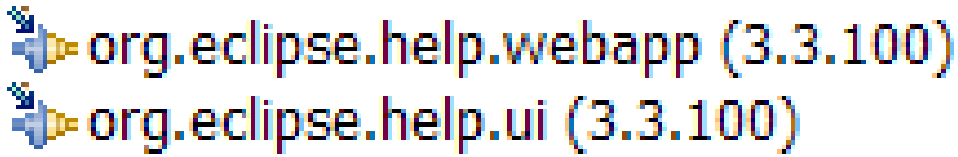
- http://www.eclipse.org/articles/article.php?file=Article-AddingHelpToRCP/index.html
- plugin.xml referenziert toc.xml-Dateien
  - Nennen Themen und Unterthemen
  - Referenzieren Hilfe-texte im HTML-Format

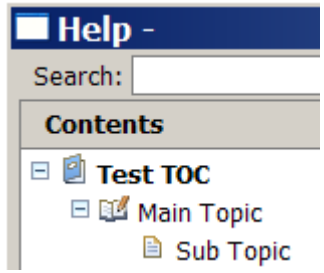
```
<toc label="Sample Table of Contents">
<topic label="Main Topic"  href="html/maintopic.html">
<topic label="Sub Topic" href="html/subtopic.html"/>
</topic>
<topic label="Main Topic 2"/>
</toc>
```

- ActionBarAdvisor registriert *HelpActions*

```
showHelpAction = ActionFactory.HELP_CONTENTS.create(window);
searchHelpAction = ActionFactory.HELP_SEARCH.create(window);
dynamicHelpAction = ActionFactory.DYNAMIC_HELP.create(window);
```

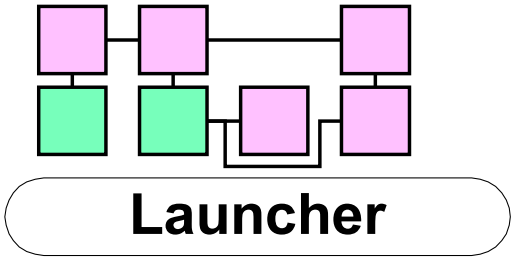
Benötigte weitere Plug-ins





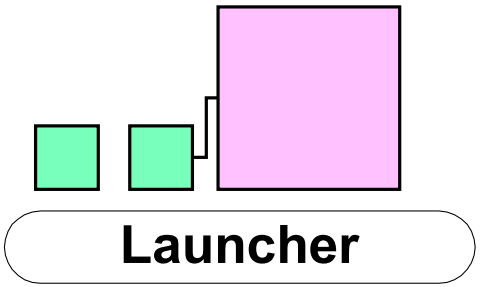
Follow me ... which way?

Eclipse Avenue



- *Viele* Plug-ins
  - *Modularität* und *Separation*
  - Verzahnung durch eigene Extension Points

Classic Road



- *Ein* Plugin,
  - sonst herkömmliches, programmatisches Vorgehen
  - Listener, Models, Events, Observer,

Bauchschmerzen #1

- Deklaratives Programmieren in XML
  - Schema, aber keine API-DOC
  - Muss ich gar das Framework *debuggen*?
  - Keine Code Completion von *CDATA*
    - kleiner Ersatz: Wizard im Plug-in Editor
- 4 Konfigurationsdateien pro Plug-in müssen zusammenspielen
  - .classpath, plugin.xml, build.properties, manifest.mf
    - *Achtung:* mit *übergreifenden* Abhängigkeiten
  - Zusätzlich: Launch-Konfiguration

Bauchschmerzen #2

- Hoher *Lernaufwand*
  - auch für sehr erfahrene Java-Entwickler
- Absolutes *Overengineering* für kleine Anwendungen
  - Interface > IAdaptable > AdaptorFactories
- *Einschränkungen* der verfügbaren Widgets
  - Button mit Icons? Tab disablen?
- Bugfixes je nach Wichtigkeit *für Eclipse*
- Feature-Stabilität
  - 3.3 Commands statt Actions
  - 3.4 Neuer Update-Mechanismus