

# XPages Extensibility API – going deep

René Winkelmeyer  
midpoints GmbH





**René Winkelmeyer**  
Senior Consultant

midpoints GmbH  
<http://www.midpoints.de>

IBM Advanced Business Partner  
IBM Design Partner for Domino Next  
IBM Mobile Design Partner  
Apple Enterprise Developer & MDM Program

#### Services

- Notes / Domino Consulting
- E-Mail Management
- App Development (IBM Connections, RCP, XPages, mobile)

#### We mobilize Notes

- Lotus Traveler planning & deployment
- mobile app development
- Apple iOS Device Management

OpenNTF Contributor und OpenNTF Director  
=> File Navigator: <http://filenavigator.openntf.org>  
=> XSnippets: <http://xsnippets.openntf.org>

## **Worüber wir heute sprechen werden**

- Unterschiede XPages, Extension Library und Extensibility API
- Vorgehensweise Plug-In-Development
- Codebeispiele
- Deployment

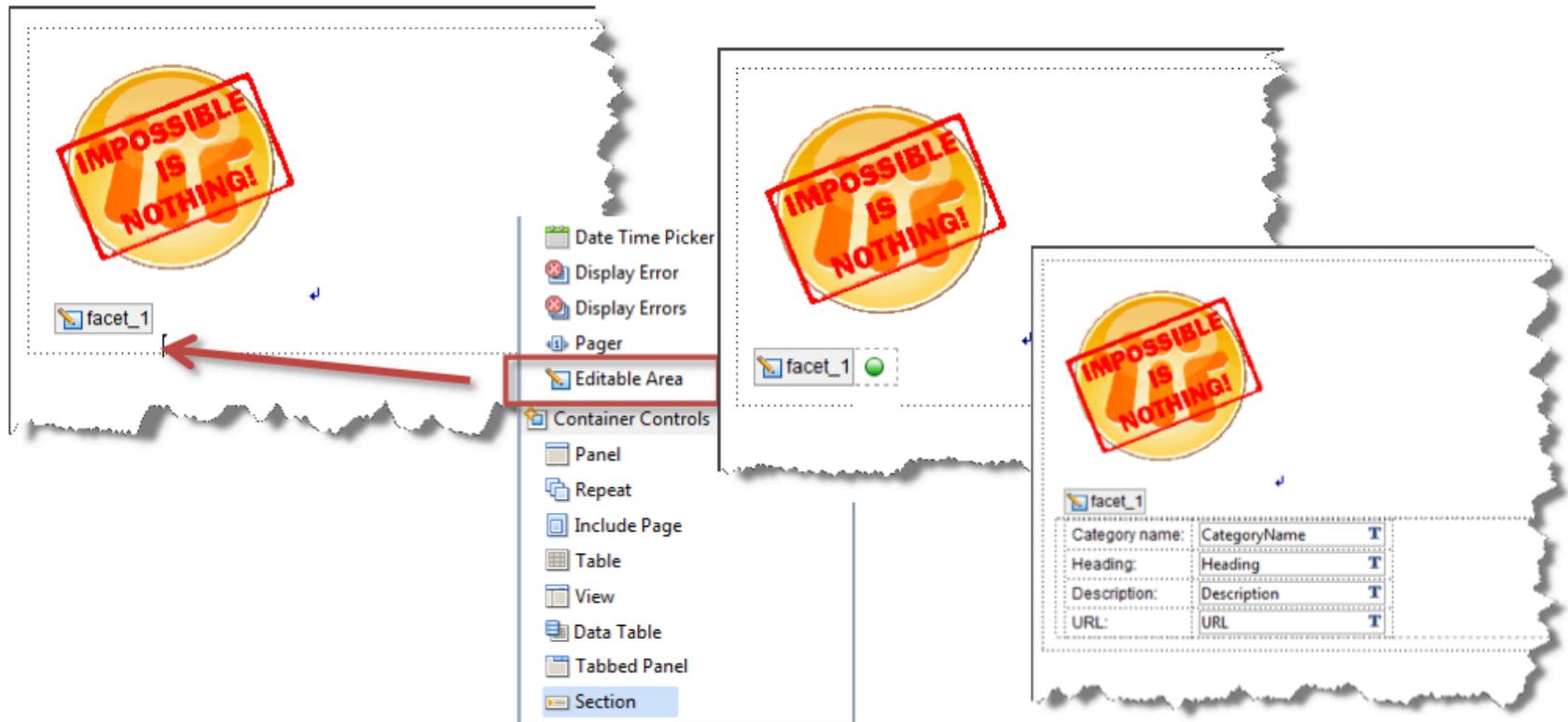
## **Worüber wir heute sprechen werden**

- Unterschiede XPages, Extension Library und Extensibility API
- Vorgehensweise Plug-In-Development
- Codebeispiele
- Deployment

- XPages ist eine großartige Technologie und erweitert – aus Entwicklungs-, aus Administrations- und vor allem aus Benutzersicht die Möglichkeiten erheblich.
- Mit XPages zu beginnen ist einfach. RAD in einer sehr guten Umsetzung. Einfaches Drag'n'Drop und (fast) alles kann berechnet werden.
- Um ehrlich zu sein: man konnte noch nie einfacher Web-Entwicklung mit Domino machen.



- Das mächtigste Werkzeug ist dabei die editable area.



The diagram illustrates the process of creating an editable area in XPages. It shows three stages of a page's development:

- A blank page with a label 'facet\_1' and a red stamp that reads "IMPOSSIBLE IS NOTHING!".
- A page with a label 'facet\_1' and a green dot, with a red arrow pointing to the "Editable Area" option in a central menu.
- A page with a label 'facet\_1' and a table containing fields for Category name, Heading, Description, and URL.

The central menu lists various controls and container controls:

- Date Time Picker
- Display Error
- Display Errors
- Pager
- Editable Area**
- Container Controls
  - Panel
  - Repeat
  - Include Page
  - Table
  - View
  - Data Table
  - Tabbed Panel
  - Section

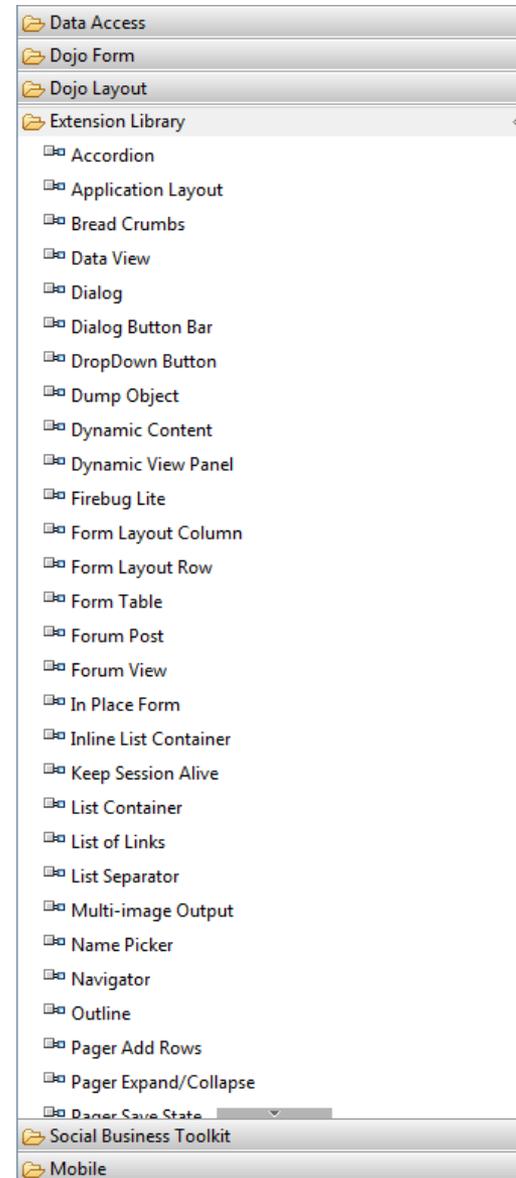
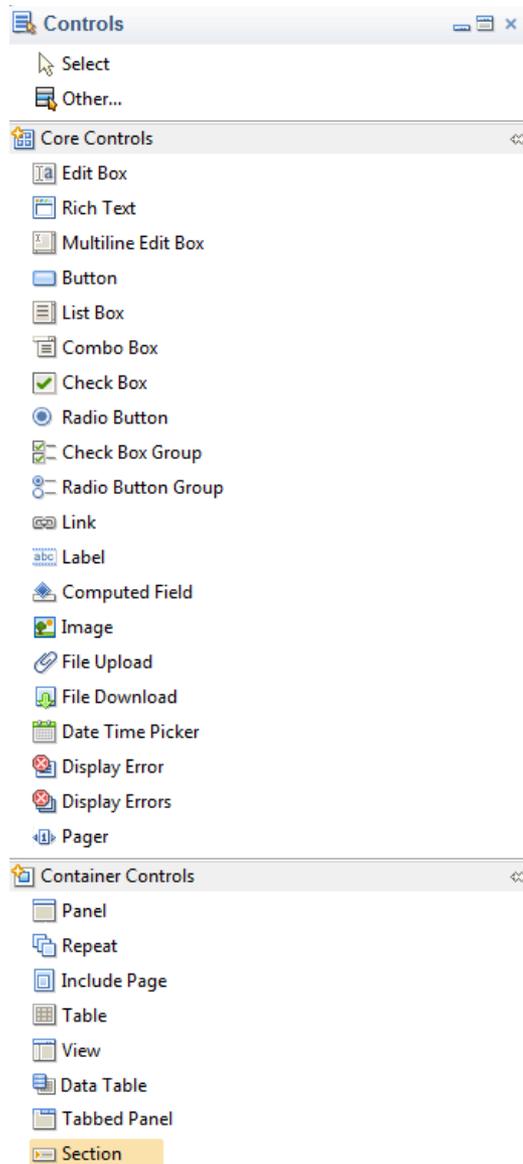


## XPages / XPages Extension Library

- Es gibt jedoch gewisse Dinge, die entweder schwierig umzusetzen sind – oder einfach nur fehlen...
  - In der Palette sind nicht die erforderlichen Controls vorhanden
  - OneUI ist zu komplex
  - Custom rendering wird benötigt
  - ...
- Viele Dinge können hier schon mit der XPages Extension Library gelöst werden.



# XPages Extension Library



## XPages Extension Library

- IBM hat mit Notes/Domino 8.5.2 die XPages Extensibility API eingeführt.
- Diese API ist ein Set von Java-Klassen und –Methoden mit denen man spezialisierte Funktionen direkt in XPages integrieren kann.
  - Dies gilt sowohl für XPages im Browser als auch im Notes-Client (XPinC).
- Mit dieser API hat IBM die viel bekannte XPages Extension Library bereitgestellt – (zuerst) auf OpenNTF und dann als Bestandteil des core product!



# XPages Extension Library

<ul style="list-style-type: none"> <li>Data Access           <ul style="list-style-type: none"> <li>JDBC Connection Manager</li> <li>Remote Service</li> <li>REST Service</li> </ul> </li> <li>Dojo Form           <ul style="list-style-type: none"> <li>Dojo Button</li> <li>Dojo Check Box</li> <li>Dojo Combo Box</li> <li>Dojo Currency Text Box</li> <li>Dojo Date Text Box</li> <li>Dojo Filtering Select</li> <li>Dojo Horizontal Slider</li> <li>Dojo Number Spinner</li> <li>Dojo Number Text Box</li> <li>Dojo Radio Button</li> <li>Dojo Simple Text Area</li> <li>Dojo Slider Rule</li> <li>Dojo Slider Rule Labels</li> <li>Dojo Text Area</li> <li>Dojo Text Box</li> <li>Dojo Time Text Box</li> <li>Dojo Toggle Button</li> <li>Dojo Validation Text Box</li> <li>Dojo Vertical Slider</li> <li>Dojo Image Select</li> <li>Dojo Link Select</li> <li>Dojo List Text Box</li> <li>Dojo Name Text Box</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Dojo Layout           <ul style="list-style-type: none"> <li>Dojo Accordion Container</li> <li>Dojo Accordion Pane</li> <li>Dojo Border Container</li> <li>Dojo Border Pane</li> <li>Dojo Content Pane</li> <li>Dojo Data Grid</li> <li>Dojo Data Grid Column</li> <li>Dojo Data Grid Row</li> <li>Dojo Stack Container</li> <li>Dojo Stack Pane</li> <li>Dojo Tab Container</li> <li>Dojo Tab Pane</li> </ul> </li> <li>Extension Library           <ul style="list-style-type: none"> <li>Accordion</li> <li>Application Layout</li> <li>Bread Crumbs</li> <li>Data View</li> <li>Dialog</li> <li>Dialog Button Bar</li> <li>DropDown Button</li> <li>Dump Object</li> <li>Dynamic Content</li> <li>Dynamic View Panel</li> <li>Firebug Lite</li> <li>Form Layout Column</li> <li>Form Layout Row</li> <li>Form Table</li> <li>Forum Post</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Forum View</li> <li>In Place Form</li> <li>Inline List Container</li> <li>Keep Session Alive</li> <li>List Container</li> <li>List of Links</li> <li>List Separator</li> <li>Multi-image Output</li> <li>Name Picker</li> <li>Navigator</li> <li>Outline</li> <li>Pager Add Rows</li> <li>Pager Expand/Collapse</li> <li>Pager Save State</li> <li>Pager Show/Hide Details</li> <li>Pager Sizes</li> <li>Pop-up Menu</li> <li>Sort Links</li> <li>Switch</li> <li>Tag Cloud</li> <li>Toolbar</li> <li>Tooltip</li> <li>Tooltip Dialog</li> <li>Value Picker</li> <li>Widget Container</li> <li>Dialog Content</li> </ul>	<ul style="list-style-type: none"> <li>Social Business Toolkit           <ul style="list-style-type: none"> <li>Connections Client</li> <li>Connections Widget</li> <li>Sametime Client</li> <li>Sametime Widget</li> </ul> </li> <li>Mobile           <ul style="list-style-type: none"> <li>Mobile Page</li> <li>Mobile Switch</li> <li>Page Heading</li> <li>Rounded List</li> <li>Single Page Application</li> <li>Static Line Item</li> <li>Tab Bar</li> <li>Tab Bar Button</li> </ul> </li> </ul>
---	---	---	---



# XPages Extension Library

- Auf OpenNTF zu finden unter <http://extlib.openntf.org>

## XPages Extension Library - Releases

Release Date	Author	Name	Downloads	Direct download
Jan 26, 2012	Niklas Heidloff	853.20120124-1243a	1,159	<a href="#">ExtensionLibraryOpenNTF-853.20120126-0415.zip</a>
Dec 21, 2011	Philippe Riand	853.20111221-0529b	1,318	<a href="#">ExtensionLibraryOpenNTF-853.20111221-0529b.zip</a>
Dec 16, 2011	Philippe Riand	853-20111215-0914c	331	<a href="#">ExtensionLibraryOpenNTF-853.20111215-0914c.zip</a>
Nov 19, 2011	Philippe Riand	853-20111117-1058	1,298	<a href="#">ExtensionLibraryOpenNTF-853-20111117-1058.zip</a>
Oct 28, 2011	Philippe Riand	853.20111027-1245a	1,028	<a href="#">ExtensionLibraryOpenNTF-853.20111027-1245a.zip</a>
Oct 28, 2011	Philippe Riand	8.5.2.201110260152NTF	683	<a href="#">XPagesExtensionLibrary-852.zip</a>
Oct 28, 2011	Philippe Riand	8.5.2.201110260152NTF	683	<a href="#">XPagesExtensionLibrary-852.zip</a>



- Verfügbar als Notes/Domino 8.5.3 Upgrade Pack 1 – voller Produktsupport durch IBM!
  - Part number: CI5GIEN
- Achtung: Bei Einsatz des Upgrade Pack 1 sind vorherige Installation der OpenNTF Extension Library vollständig zu entfernen.
  - IBM installiert die Inhalte des UP1 in das Programmverzeichnis, während die OpenNTF Extension Library in das Data-Verzeichnis eingeführt wird.
  - Aktuell nur verfügbar als Installer – nicht per Updatesite o. ä.



# XPages Extensibility API

[All Classes](#)

Packages

- [com.ibm.commons](#)
- [com.ibm.commons.extension](#)
- [com.ibm.commons.log](#)
- [com.ibm.commons.util](#)
- [com.ibm.commons.util.io](#)
- [com.ibm.commons.util.io.base64](#)
- [com.ibm.commons.util.io.json](#)
- [com.ibm.commons.util.profiler](#)
- [com.ibm.xsp](#)
- [com.ibm.xsp.acl](#)
- [com.ibm.xsp.actions](#)
- [com.ibm.xsp.actions.client](#)
- [com.ibm.xsp.actions.document](#)
- [com.ibm.xsp.ajax](#)

**All Classes**

- [AbstractClientSimpleAction](#)
- [AbstractCompiledPage](#)
- [AbstractCompiledPage.ComponentInfo](#)
- [AbstractCompiledPageDispatcher](#)
- [AbstractConfirmAction](#)
- [AbstractConverter](#)
- [AbstractDataContainer](#)
- [AbstractDataModel](#)
- [AbstractDataSource](#)
- [AbstractDataSource.RuntimeProperties](#)
- [AbstractDocumentAction](#)
- [AbstractDocumentConfirmAction](#)
- [AbstractDocumentDataSource](#)
- [AbstractDominoViewEntry](#)
- [AbstractException](#)
- [AbstractIndexAction](#)

**Overview** Package Class **Tree** [Deprecated](#) [Index](#) [Help](#)

[PREV](#) [NEXT](#)

[FRAMES](#) [NO FRAMES](#)

## Packages

<a href="#">com.ibm.commons</a>	Common Platform class.
<a href="#">com.ibm.commons.extension</a>	Common Extension classes
<a href="#">com.ibm.commons.log</a>	Common Log classes
<a href="#">com.ibm.commons.util</a>	Common utility classes.
<a href="#">com.ibm.commons.util.io</a>	Common Input,Output and Reader classes
<a href="#">com.ibm.commons.util.io.base64</a>	Common Base64 classes
<a href="#">com.ibm.commons.util.io.json</a>	Common JSON classes
<a href="#">com.ibm.commons.util.profiler</a>	Common Profiler classes
<a href="#">com.ibm.xsp</a>	XSP Core classes.
<a href="#">com.ibm.xsp.acl</a>	ACL classes.
<a href="#">com.ibm.xsp.actions</a>	Action classes.
<a href="#">com.ibm.xsp.actions.client</a>	Client Simple Action classes
<a href="#">com.ibm.xsp.actions.document</a>	Document classes.
<a href="#">com.ibm.xsp.ajax</a>	AJAX classes
<a href="#">com.ibm.xsp.application</a>	Application classes.
<a href="#">com.ibm.xsp.application.events</a>	Application Events classes.
<a href="#">com.ibm.xsp.binding</a>	Binding classes.
<a href="#">com.ibm.xsp.complex</a>	Complex value binding classes.
<a href="#">com.ibm.xsp.component</a>	Component classes.
<a href="#">com.ibm.xsp.component.internal</a>	Component View Column Text class



## XPages Extensibility API

- Es macht nur dann Sinn eigene XPages Extensions zu erstellen, wenn man in mehrere Applikationen häufig die selbe Funktionalität benötigt.
- Eine Extension (OSGi Plug-In) muss dabei nur einmal auf den Domino-Server/den Notes-Client deployed werden – im Gegensatz zu custom controls.

<http://www.osgi.org> <= unbedingt anschauen



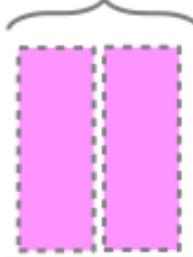
## XPages Extensibility API

- So genannte „Artefakte“ könne über die vorhandene API zur Runtime hinzugefügt werden.
- JSF ist ein „offener Standard“ dessen APIs öffentlich verfügbar sind.
- Seit Domino 8.5.2 sind die XPages APIs veröffentlicht worden.
- Artefakte können sein:
  - UI Controls
  - Converters
  - Validators
  - Data Sources
  - Simple Actions
  - Language Bindings
  - ...



# XPages Architektur

NSF Applications

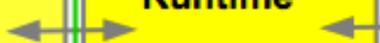


XPages  
Extensions  
OSGi bundles



XPD Profile

XPages  
Runtime



OSGi Runtime

Domino HTTP Task

## Domino Server

NSF Applications

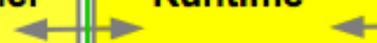


XPages  
Extensions  
OSGi bundles



XPD Profile +  
Web Container

XPages  
Runtime



OSGi Runtime

Notes Client Process

## Notes Client



## XPages API Konzepte

- XPages Library
  - OSGi bundle, enthält Java Code, Konfigurationsdateien
- Controls
  - Komponenten die in der Designer-Palette erscheinen
- Renderer
  - Renderer geben HTML oder CSJS aus
- Complex Types
  - Hiermit können z. B. Parameter gespeichert werden. Sie werden auch verwendet um gekapselte Funktionen zu realisieren.



## **Worüber wir heute sprechen werden**

- Unterschiede XPages, Extension Library und Extensibility API
- Vorgehensweise Plug-In-Development
- Codebeispiele
- Deployment



**IBM.**

**Lotus. Notes. 8**

Oh my!

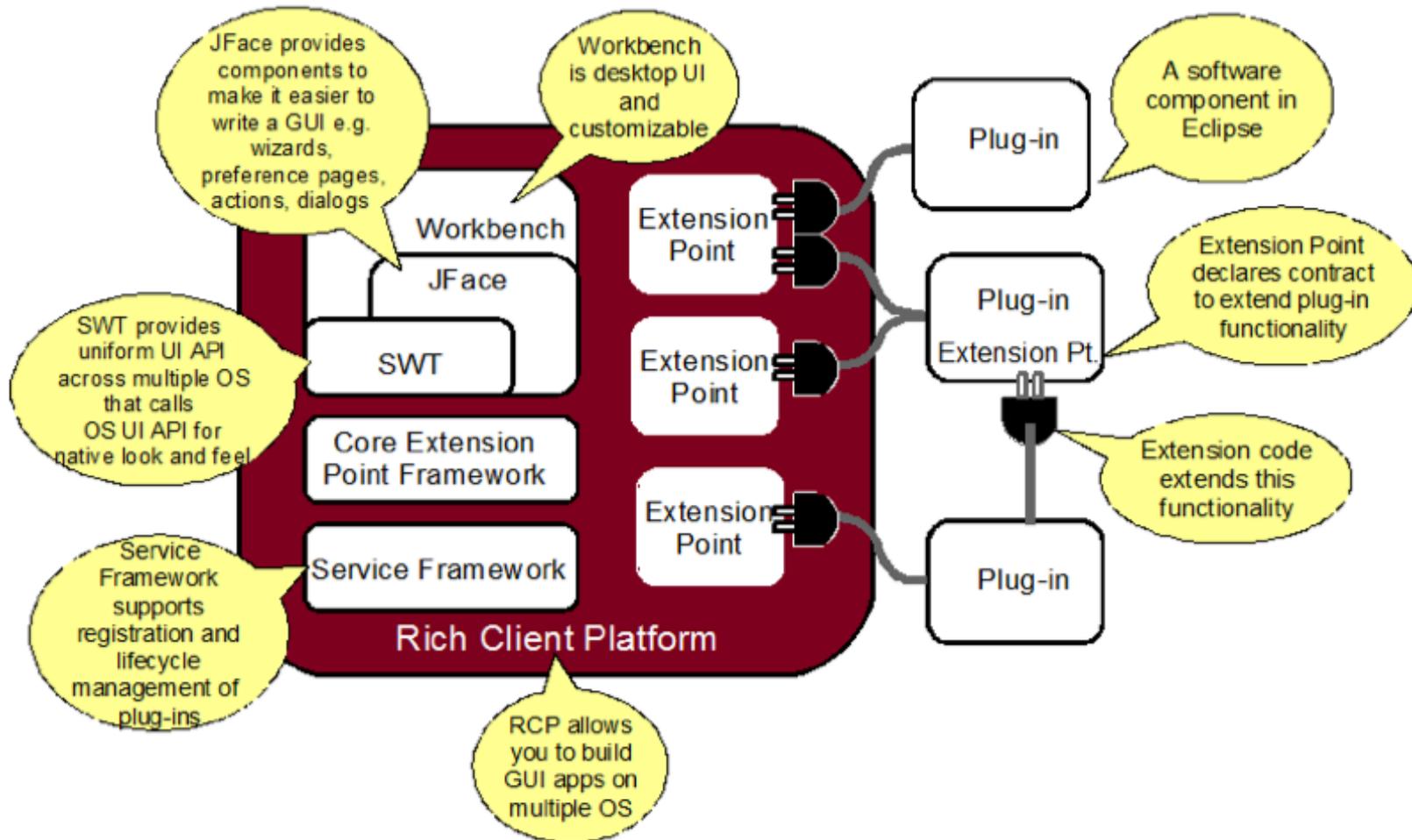
Licensed Materials - Property of IBM L-GHUS-8KVQCV © Copyright IBM Corporation and its licensors 1985, 2007. All Rights Reserved. IBM, the IBM logo, Lotus and Notes are trademarks of IBM Corporation in the United States, other countries, or both. Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. Built on Eclipse is a trademark of Eclipse Foundation, Inc. Other company, product or service names may be trademarks or service marks of others. This Program is licensed under the terms of the license agreement accompanying the Program. This license agreement may be either located in a Program directory folder or library identified as "License" or "Non\_IBM\_License", if applicable, or provided as a printed license agreement. Please read the agreement carefully before using the Program. By using the Program you agree to these terms.

**BUILT ON eclipse**

**Java**  
CONTRIBUTE



# Eclipse Plug-In-Architektur



# IBM Clients/Server auf Basis von Eclipse



Portal



Browser



Desktop



Mobile

ecosystem of partners

Apps and Plug-ins (ISV + Customer)

Industry knowledge and experience

collaborative services

Symphony

Sametime

Notes

Integrated applications

open and extensible

Expeditor



Security

Multiplatform support

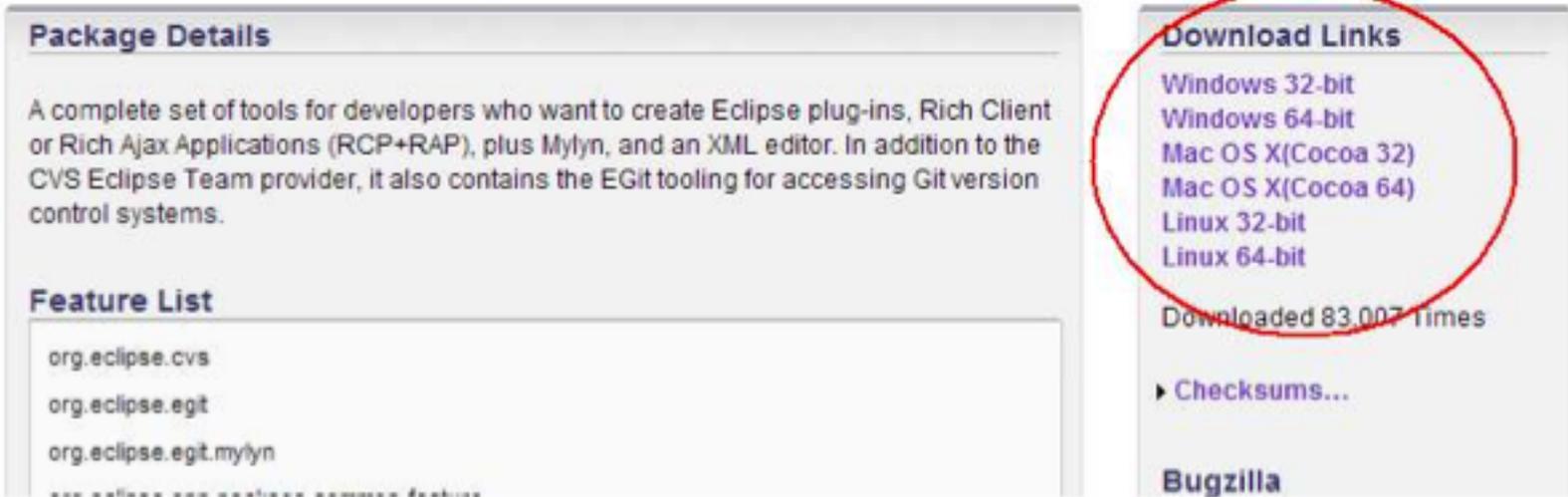
Scalability



## Setup der Entwicklungsumgebung

- Die Entwicklung erfolgt mittels Eclipse. Für diese Präsentation wird Eclipse 3.7 (32bit) eingesetzt.

### Eclipse for RCP and RAP Developers



The screenshot shows the Eclipse package details page. The 'Package Details' section describes the tools for creating Eclipse plug-ins, RCP+RAP, Mylyn, and XML editor. The 'Feature List' includes org.eclipse.cvs, org.eclipse.egit, and org.eclipse.egit.myllyn. The 'Download Links' section, circled in red, lists Windows 32-bit, Windows 64-bit, Mac OS X(Cocoa 32), Mac OS X(Cocoa 64), Linux 32-bit, and Linux 64-bit. Below the links, it states 'Downloaded 83.007 Times' and has a 'Checksums...' link. The 'Bugzilla' link is also visible.

**Package Details**

A complete set of tools for developers who want to create Eclipse plug-ins, Rich Client or Rich Ajax Applications (RCP+RAP), plus Mylyn, and an XML editor. In addition to the CVS Eclipse Team provider, it also contains the EGit tooling for accessing Git version control systems.

**Feature List**

- org.eclipse.cvs
- org.eclipse.egit
- org.eclipse.egit.myllyn

**Download Links**

- Windows 32-bit
- Windows 64-bit
- Mac OS X(Cocoa 32)
- Mac OS X(Cocoa 64)
- Linux 32-bit
- Linux 64-bit

Downloaded 83.007 Times

► Checksums...

Bugzilla



# Setup der Entwicklungsumgebung

- Zur Vereinfachung setzen wir das „XPages SDK for Eclipse RCP“ ein.
  - Bereitgestellt auf OpenNTF durch Nathan T. Freeman:  
<http://www.openntf.org/internal/home.nsf/project.xsp?action=openDocument&name=XPages%20SDK%20for%20Eclipse%20RCP>

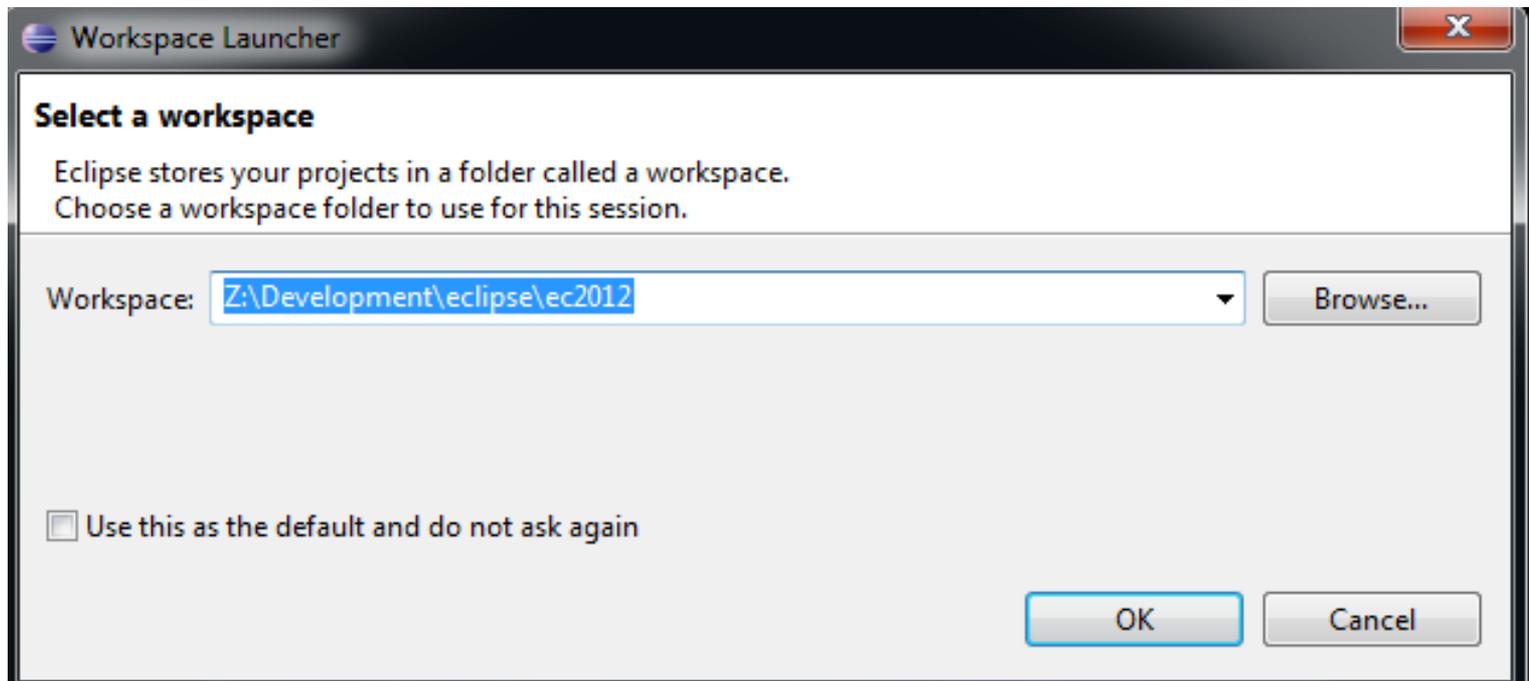
## XPages SDK for Eclipse RCP

Owners	<a href="#">Nathan T Freeman</a> , <a href="#">Niklas Heidloff</a>	Category	Admin Utilities
Contributors	<a href="#">Niklas Heidloff</a>	Platform	R 8.5.2
Downloads	256 <a href="#">Download latest release</a>	Last Release	Feb 7, 2012
Rating	☆☆☆☆☆ (0 ratings)	Project Creation	Jan 10, 2012
Status	Active	Short URL	Not defined
Description	XPages SDK for Eclipse RCP	In Catalog	No

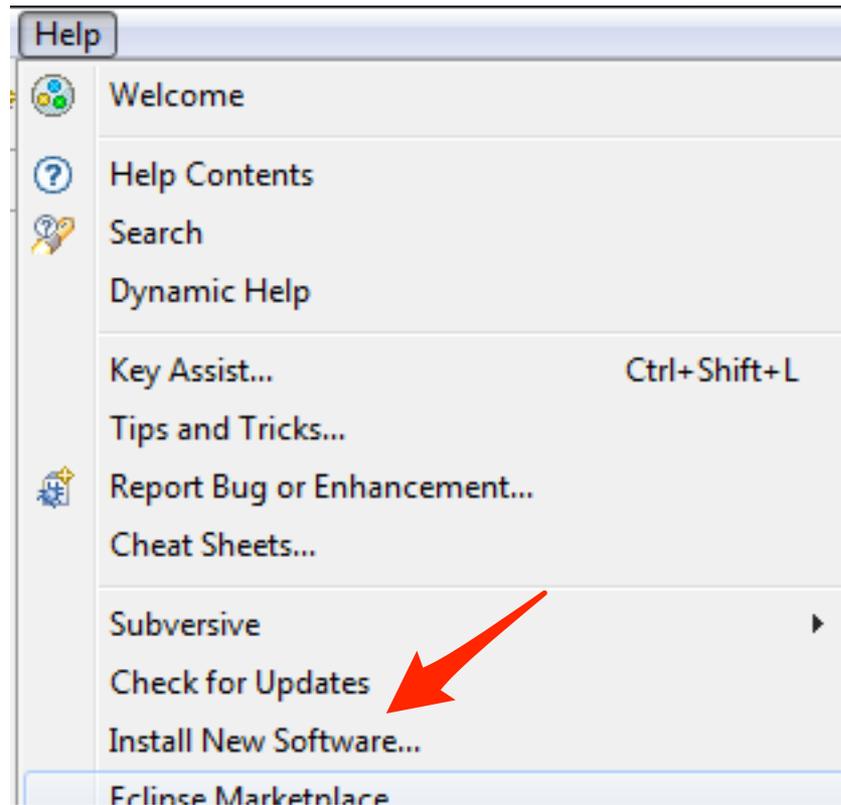


## Setup der Entwicklungsumgebung

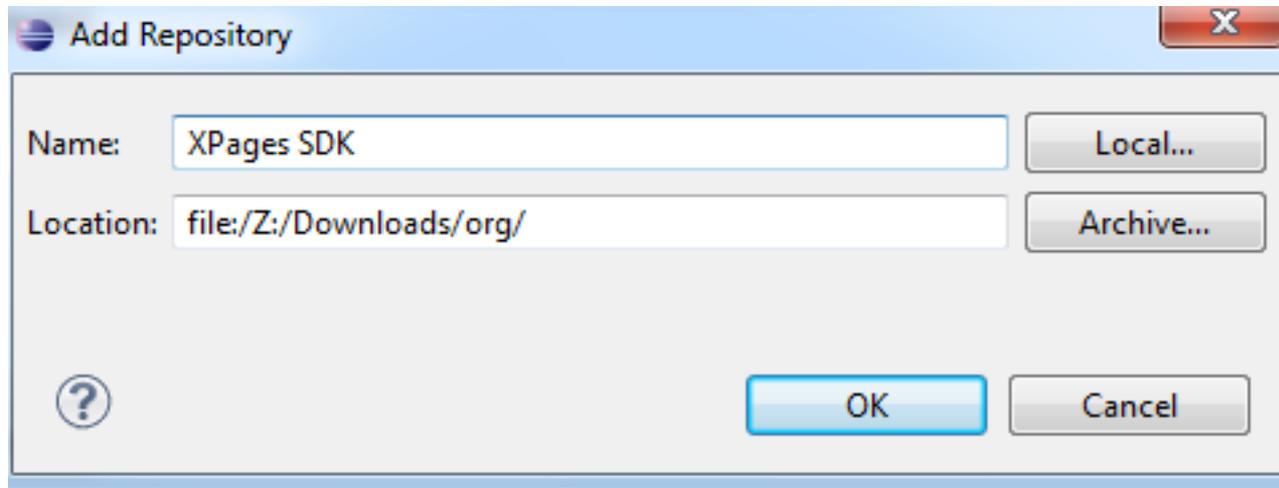
- Zuerst erstellen wir nach dem Start von Eclipse einen neuen Workspace.



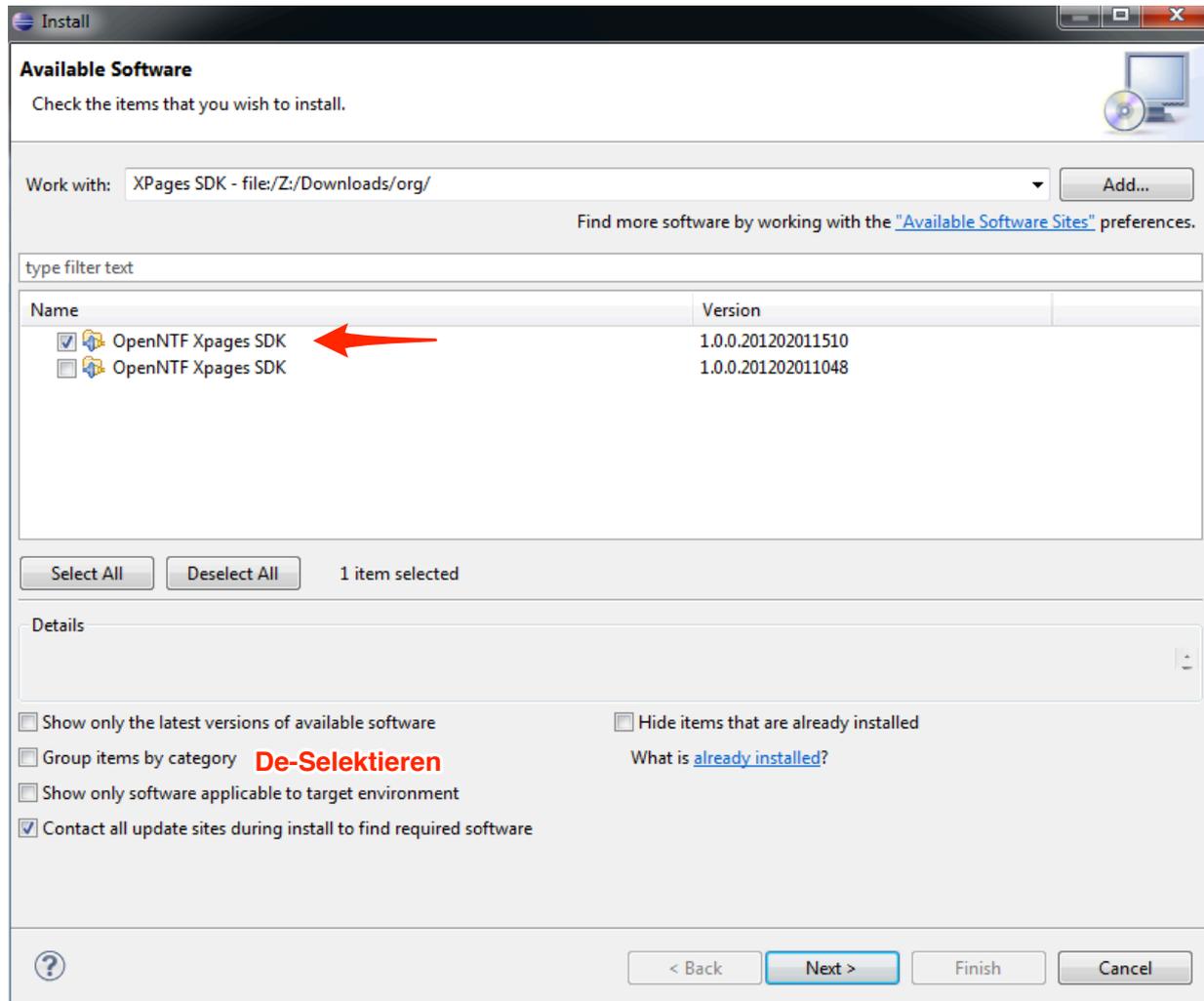
- Installation neuer Software innerhalb von Eclipse



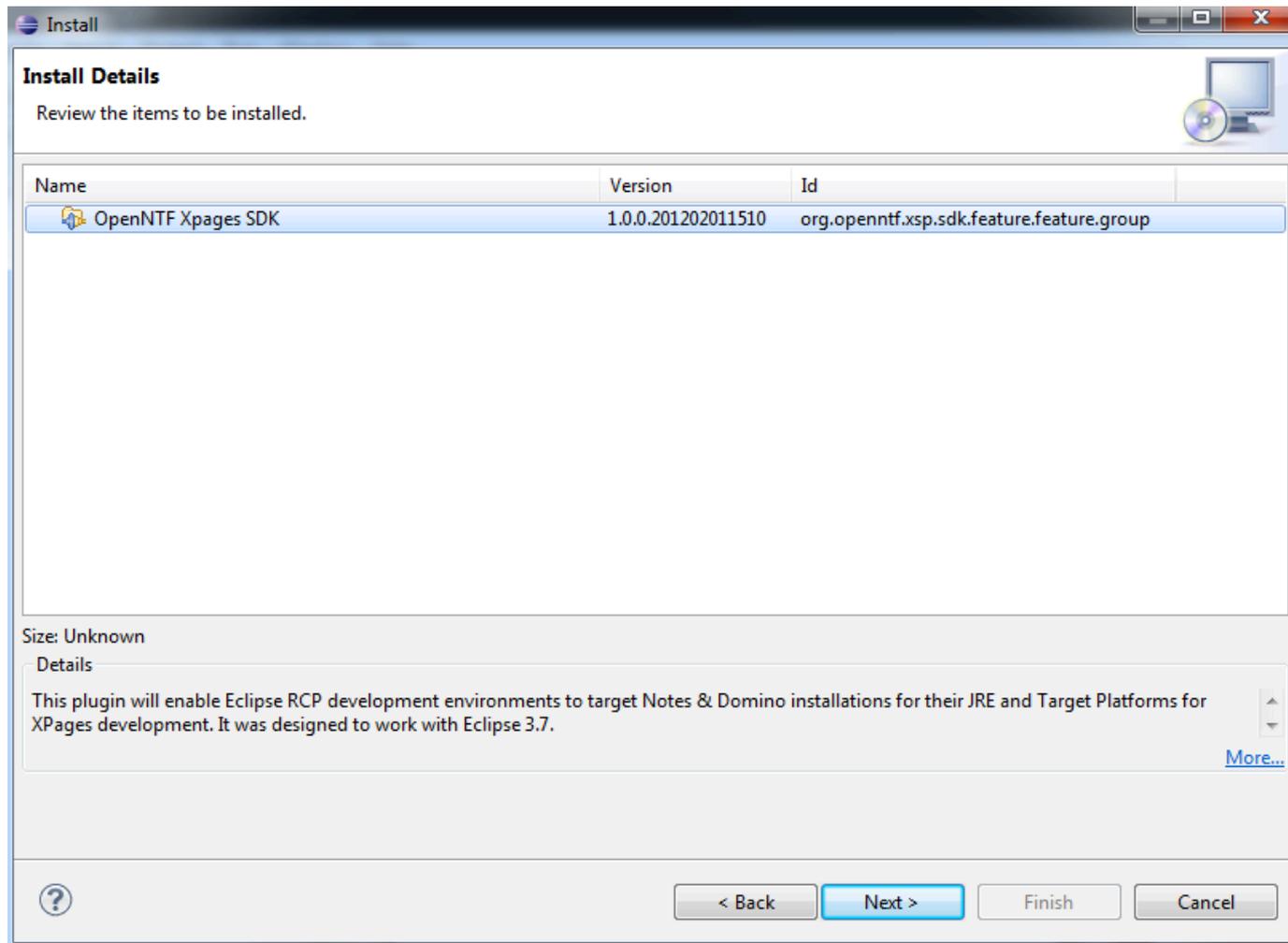
- Auswahl des XPages SDK



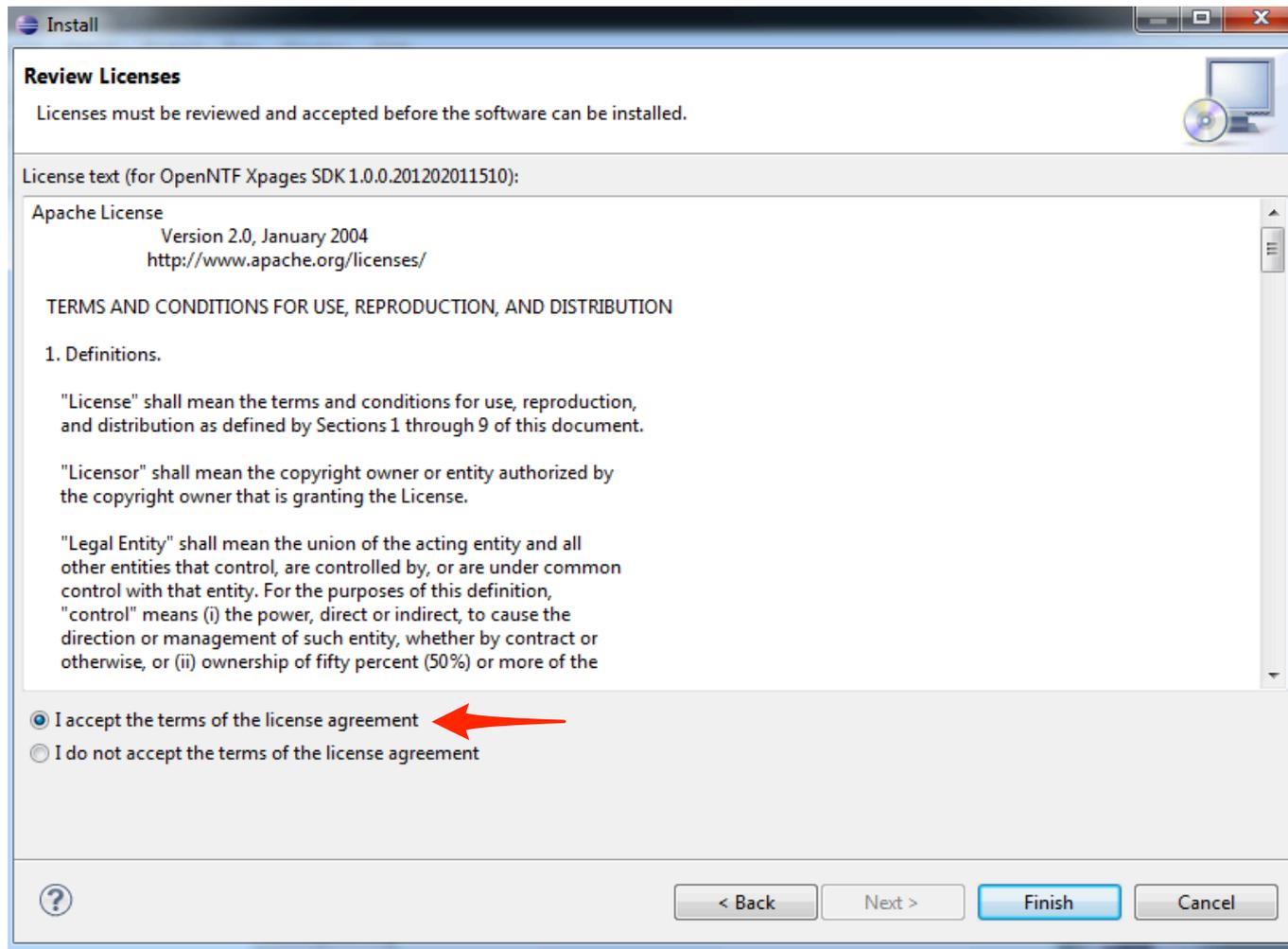
## ■ Installation des XPages SDK (I)



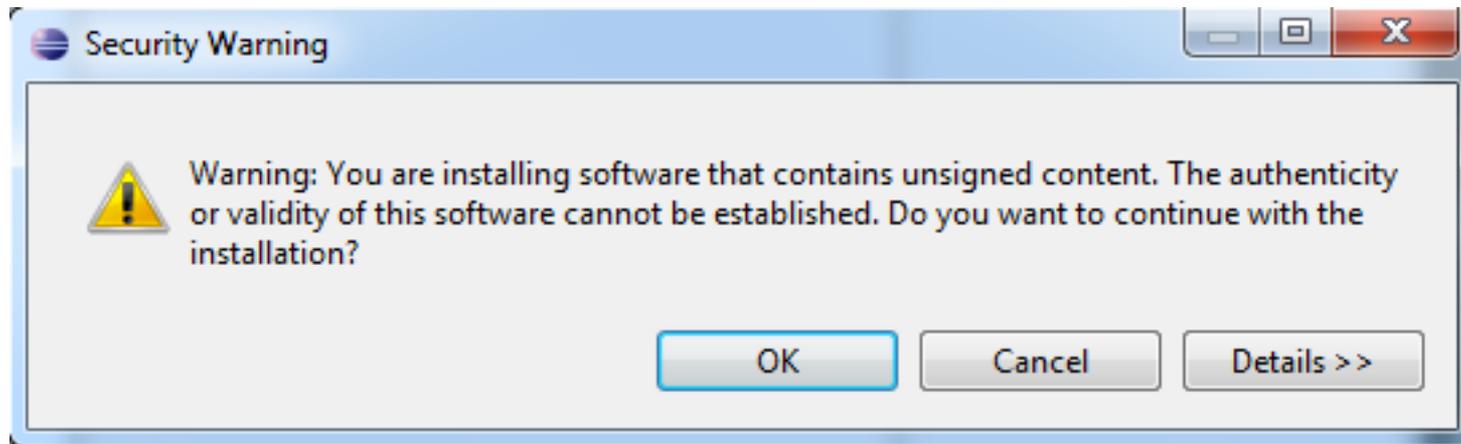
- Installation des XPages SDK (II)



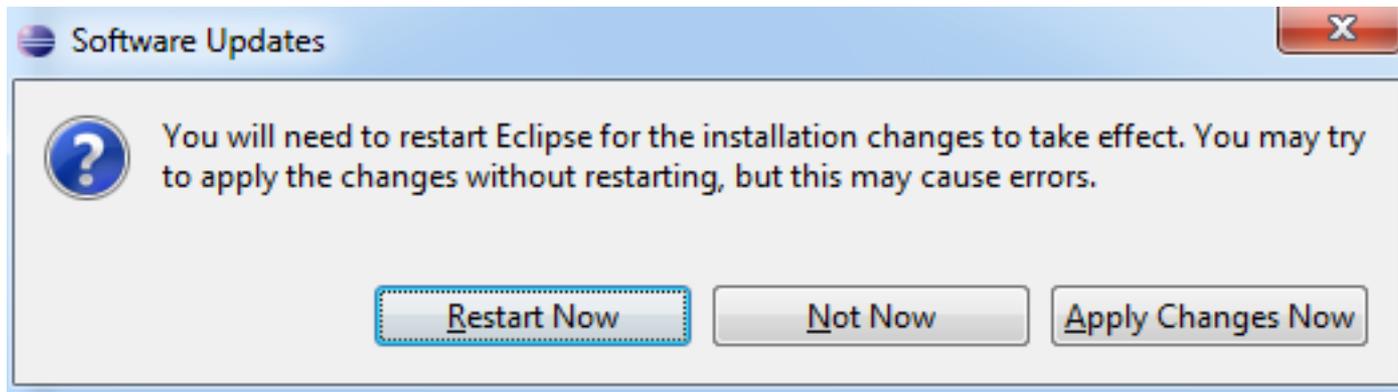
## ■ Installation des XPages SDK (III)



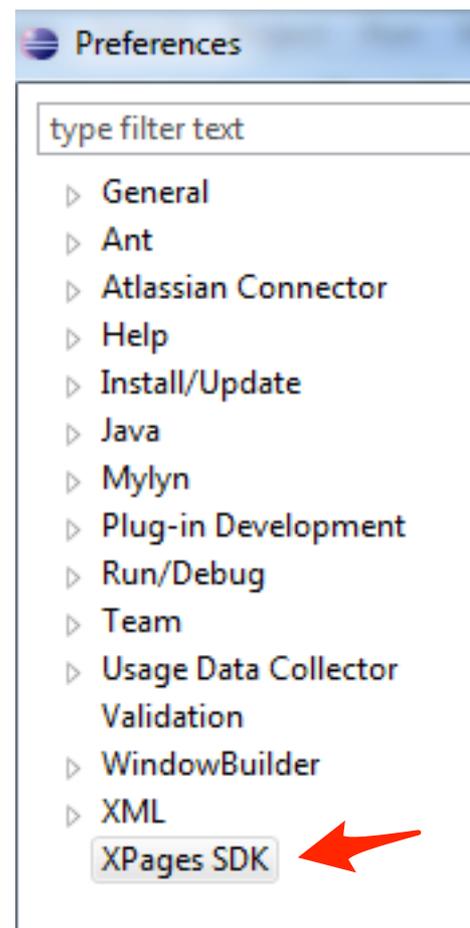
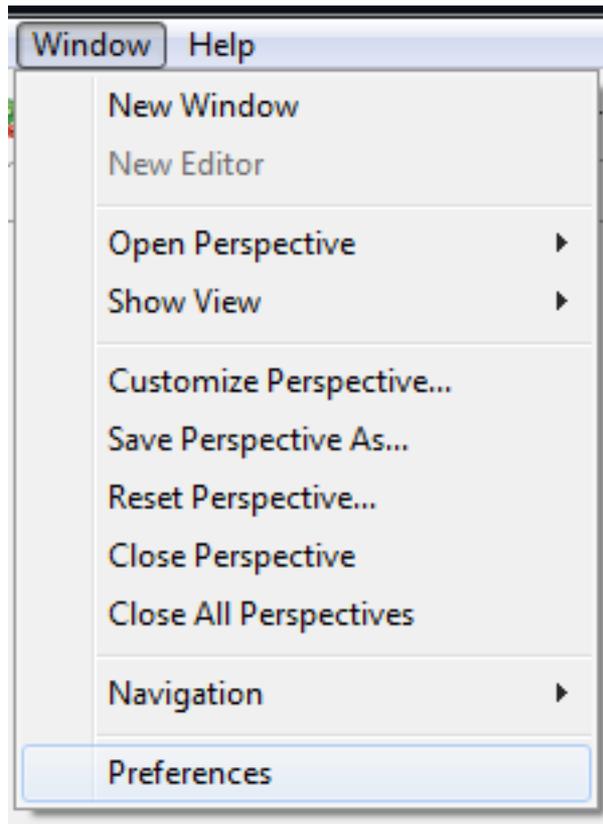
- Installation des XPages SDK (IV)
  - Unsigniert ist ok – es bedeutet, dass die Plug-Ins nicht digital signiert wurden



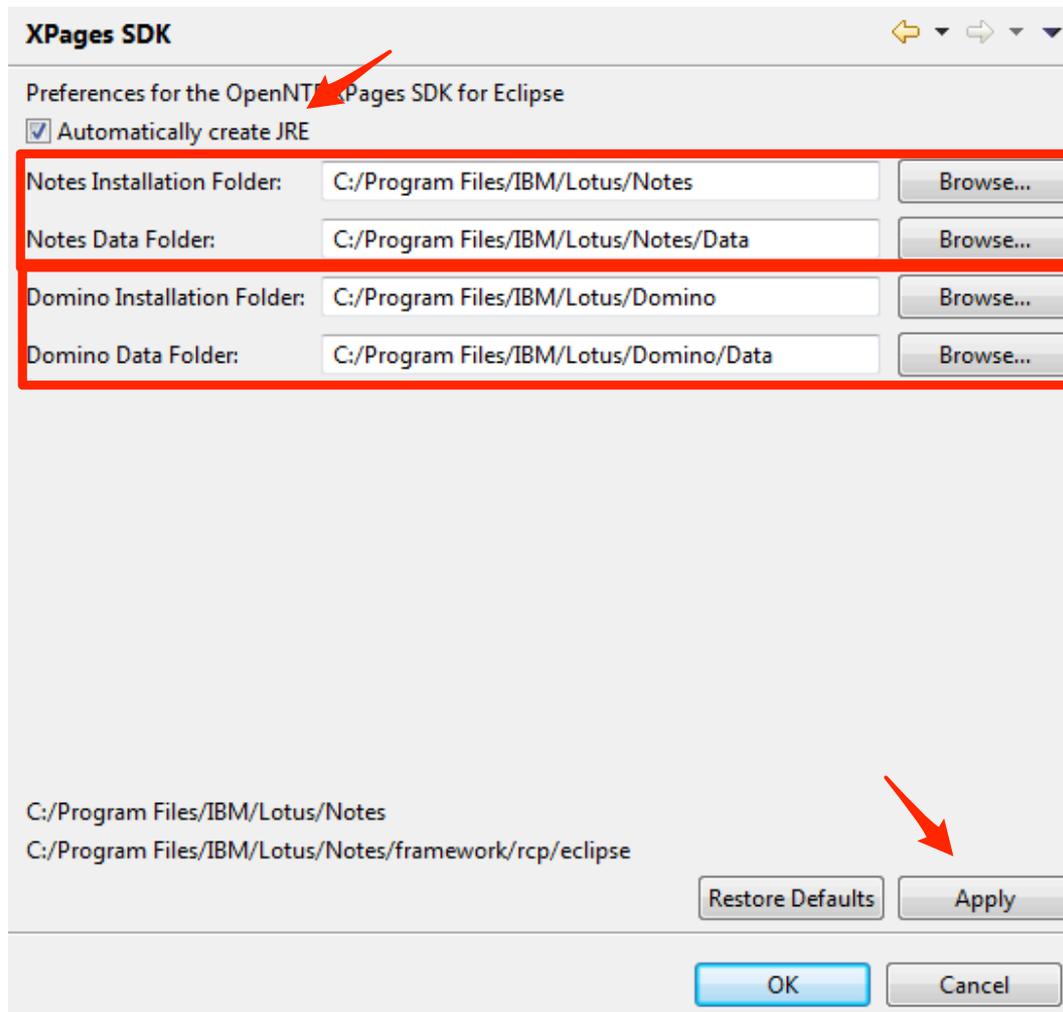
- Installation des XPages SDK (V)



- Konfiguration des XPages SDK (I)



- Konfiguration des XPages SDK (II)



# Setup der Entwicklungsumgebung

- Konfiguration des XPages SDK (III)

type filter text

- General
- Ant
- Atlassian Connector
- Help
- Install/Update
- Java**
- Appearance
- Build Path
- Code Style
- Compiler
- Debug
- Editor
- Installed JREs**
- Execution Environments
- JUnit
- Properties Files Editor
- Mylyn

### Installed JREs

Add, remove or edit JRE definitions. By default, the checked JRE is added to the build path of newly created Java projects.

Installed JREs:

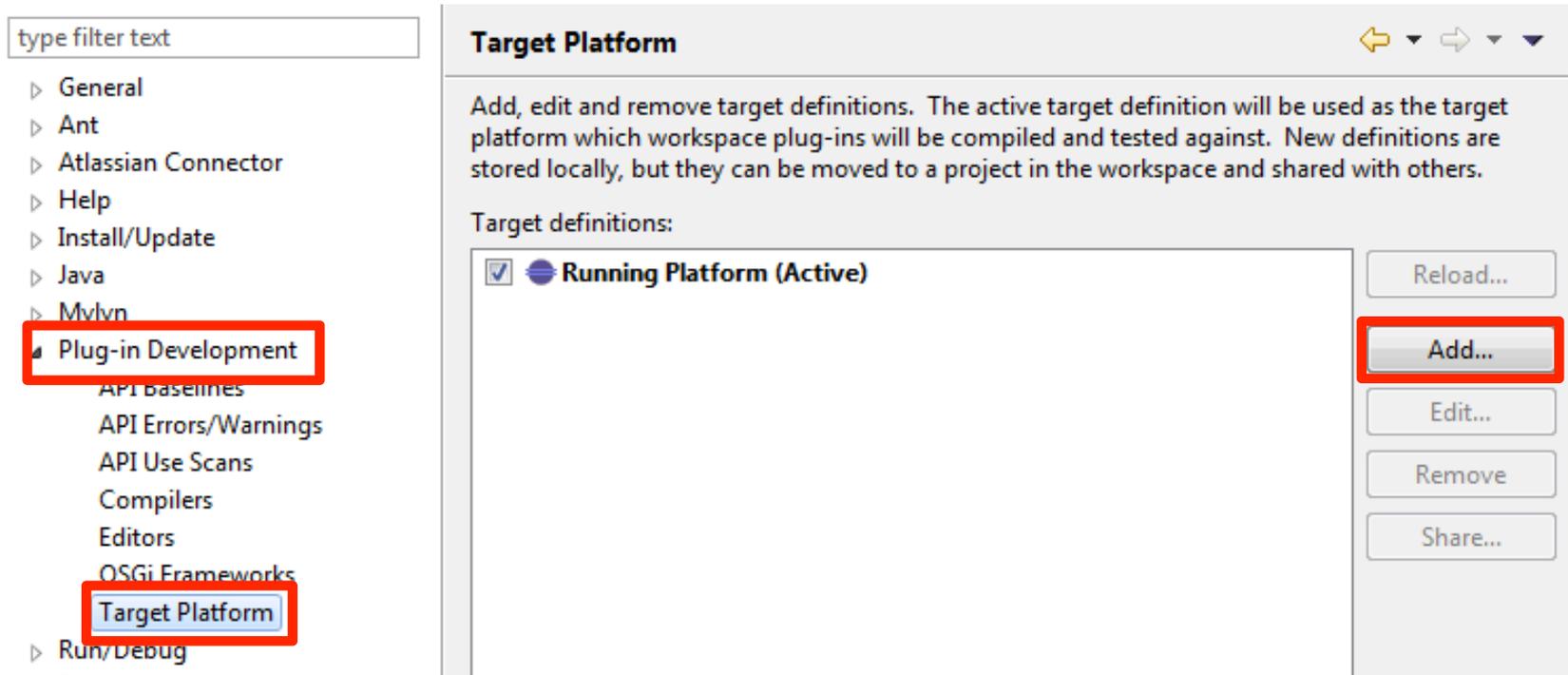
Name	Location	Type
<input type="checkbox"/> jre7	C:\Program Files\Java\jre7	Standard VM
<input checked="" type="checkbox"/> XPages Domino JRE	C:\Program Files\IBM\Lotus\Domino\jvm	Standard VM
<input type="checkbox"/> XPages Notes JRE	C:\Program Files\IBM\Lotus\Notes\jvm	Standard VM

Buttons: Add..., Edit..., Duplicate, Remove, Search.



# Setup der Entwicklungsumgebung

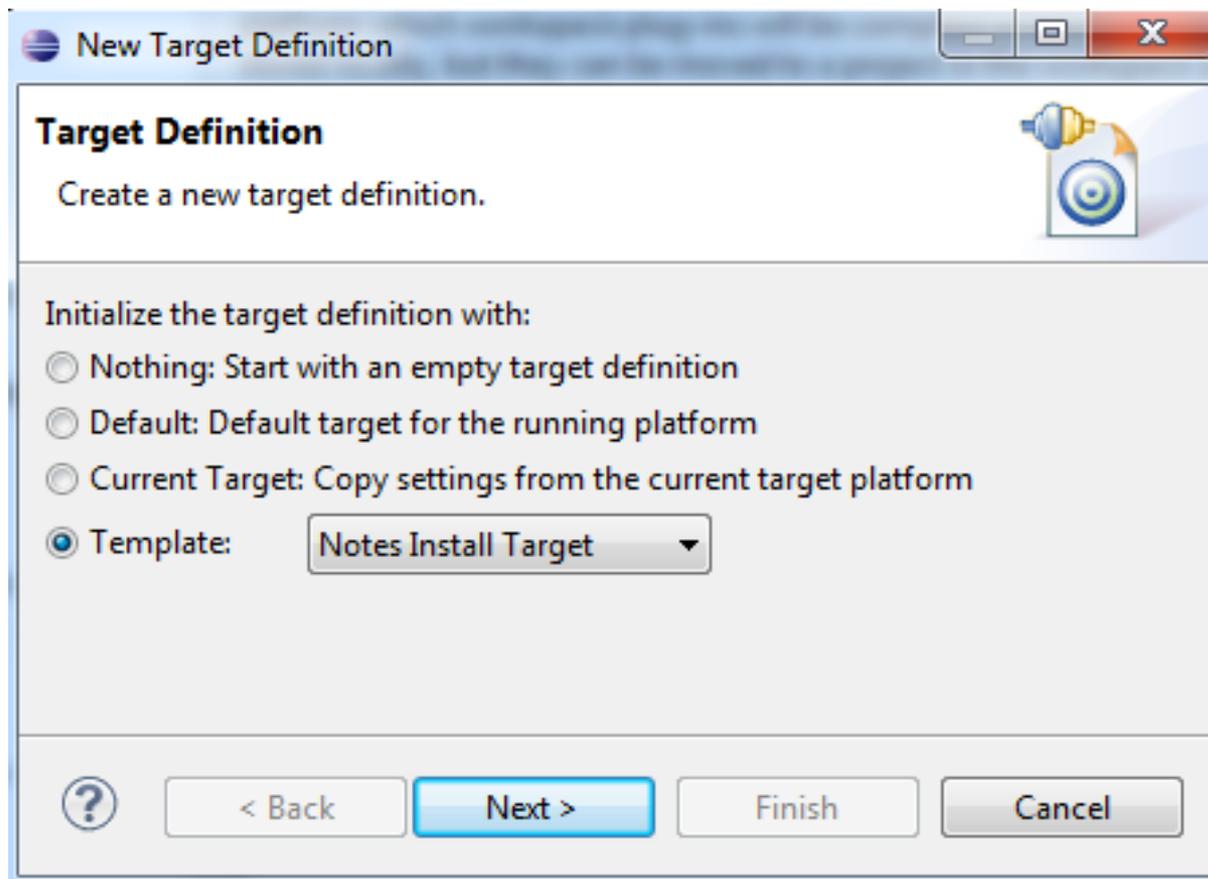
- Konfiguration des XPages SDK (IV)



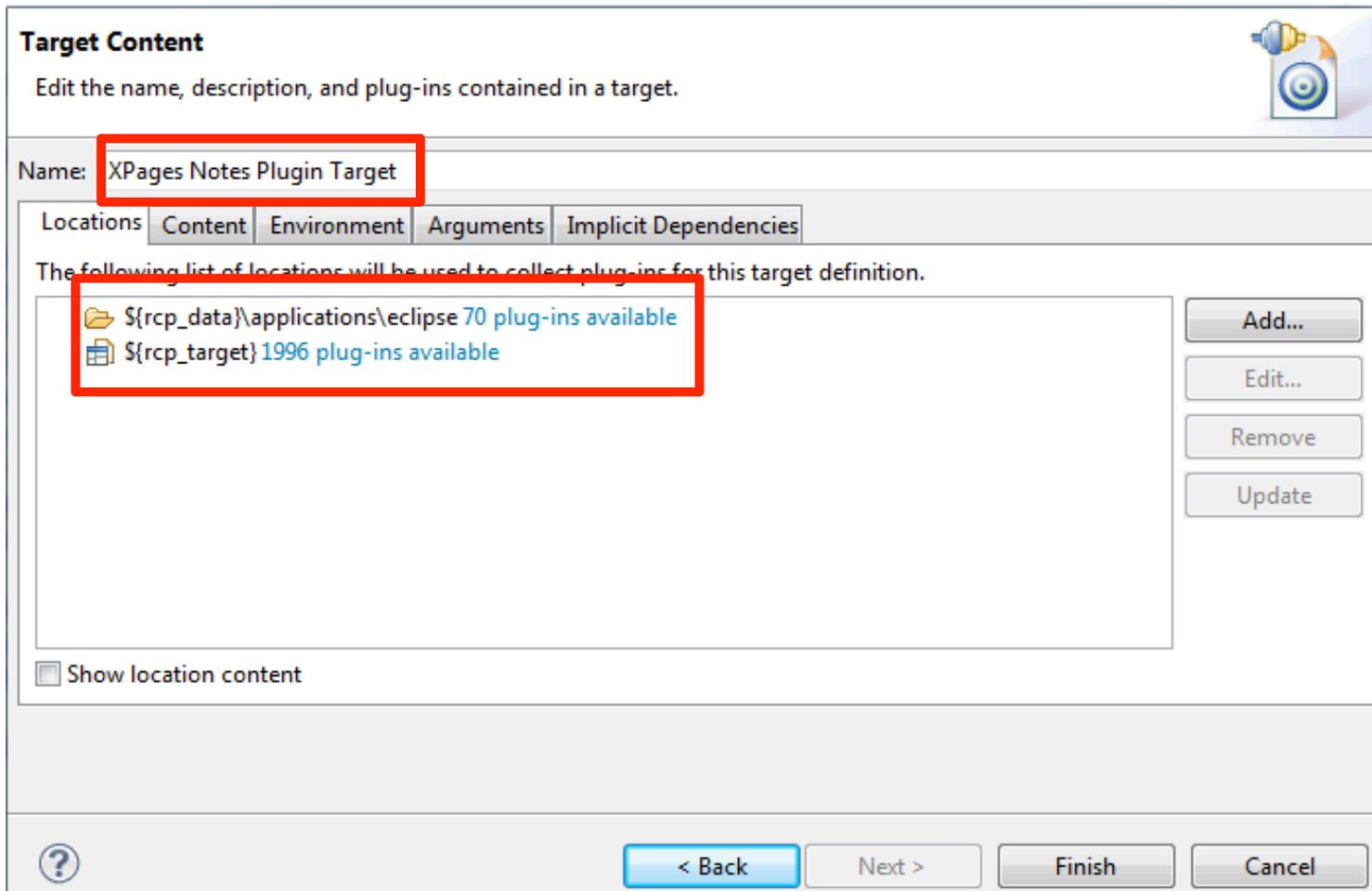
The screenshot shows the Eclipse IDE interface. On the left, a tree view displays the 'Plug-in Development' category, with 'Target Platform' selected and highlighted by a red box. The main area shows the 'Target Platform' dialog box. The dialog contains the following text: 'Add, edit and remove target definitions. The active target definition will be used as the target platform which workspace plug-ins will be compiled and tested against. New definitions are stored locally, but they can be moved to a project in the workspace and shared with others.' Below this text, a list of 'Target definitions' shows 'Running Platform (Active)' with a checked checkbox and a globe icon. To the right of the list are several buttons: 'Reload...', 'Add...' (highlighted with a red box), 'Edit...', 'Remove', and 'Share...'.



- Konfiguration des XPages SDK (V)



- Konfiguration des XPages SDK (VI)



**Target Content**  
Edit the name, description, and plug-ins contained in a target.

Name: XPages Notes Plugin Target

Locations | Content | Environment | Arguments | Implicit Dependencies

The following list of locations will be used to collect plug-ins for this target definition.

- Folder icon: `${rcp_data}\applications\eclipse 70 plug-ins available`
- Folder icon: `${rcp_target} 1996 plug-ins available`

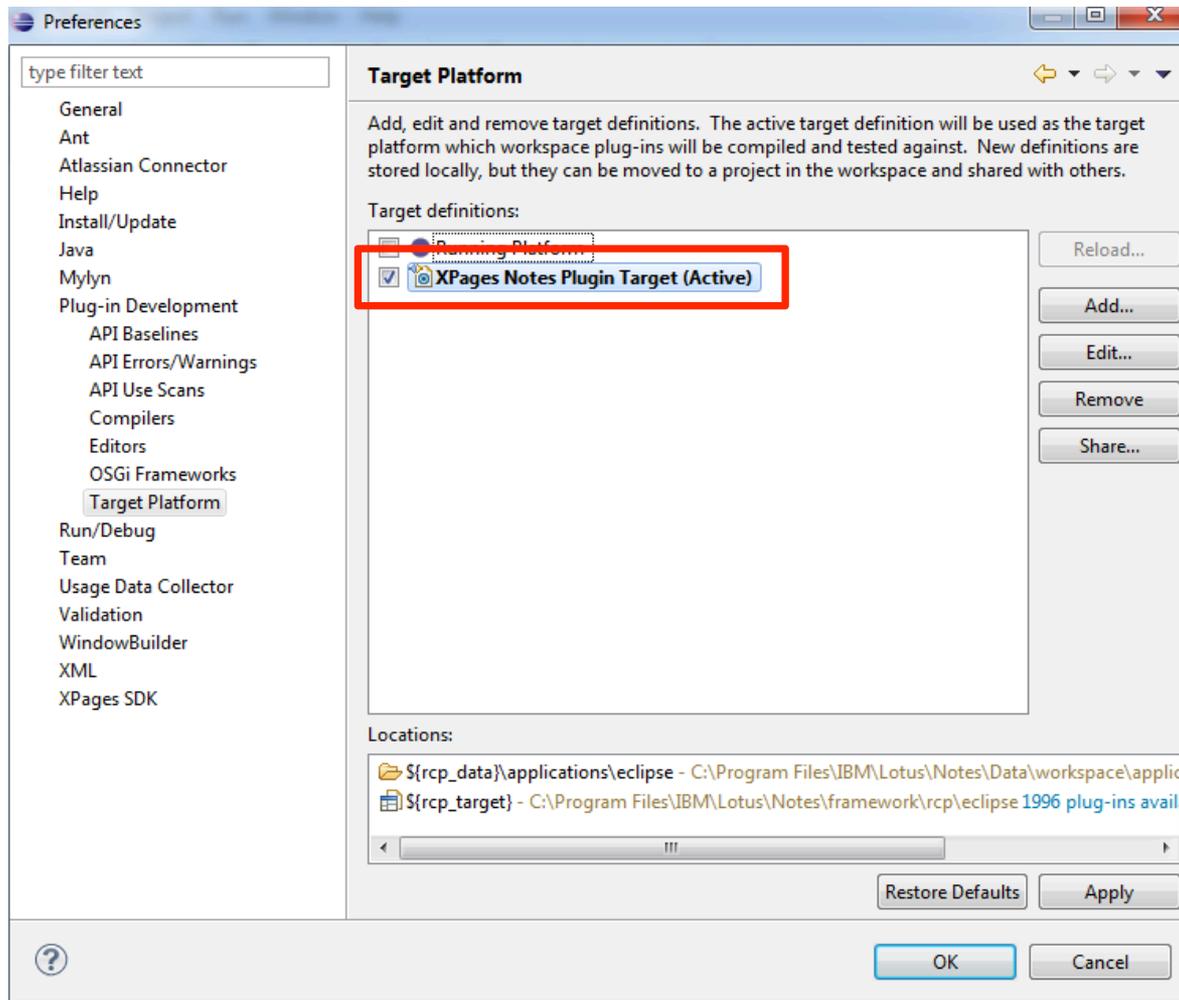
Buttons: Add..., Edit..., Remove, Update

Show location content

Navigation: < Back, Next >, Finish, Cancel



- Konfiguration des XPages SDK (VII)

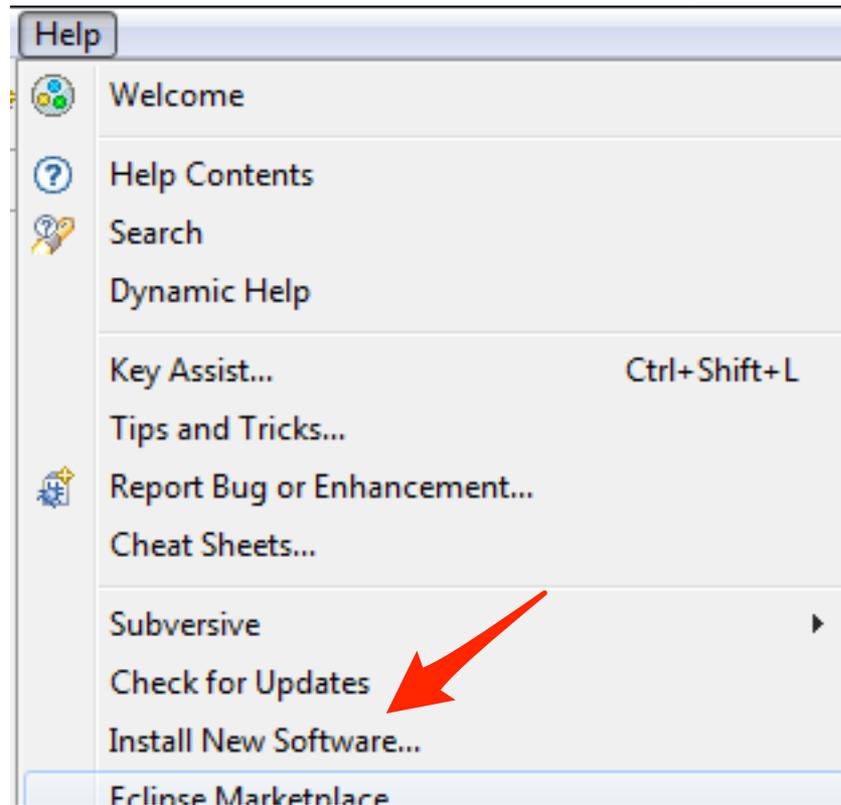


## Setup der Entwicklungsumgebung

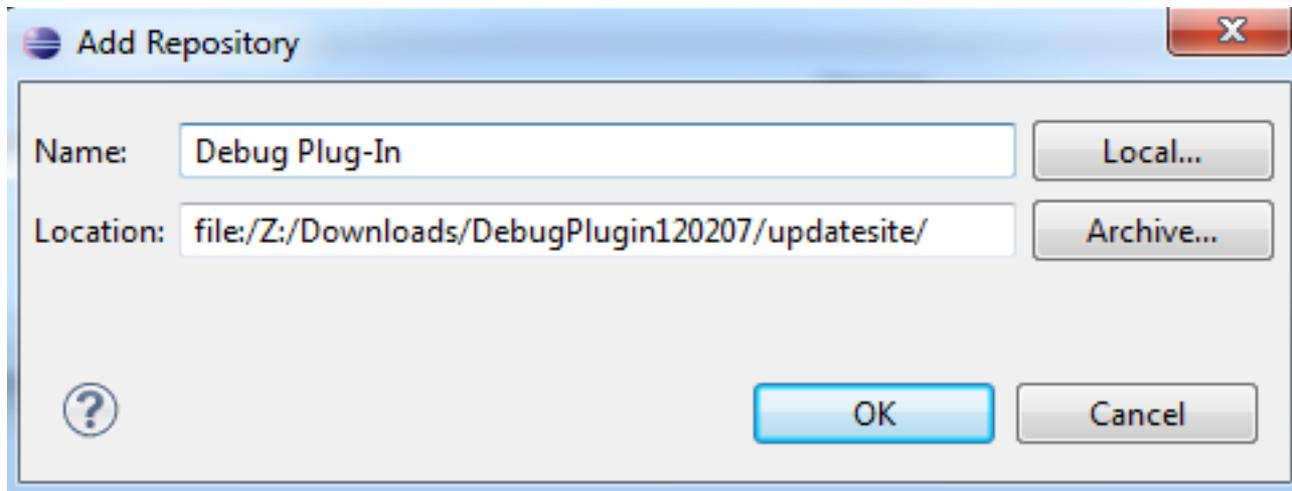
- Ohne Debugging sollte man nicht entwickeln – also ist noch ein Debugger zu installieren.
- Wir setzen hier das Domino Debug Plug-In von OpenNTF ein.
  - <http://www.openntf.org/internal/home.nsf/release.xsp?documentId=CBF874E9C4607B4C8625799D00287B8C&action=openDocument>



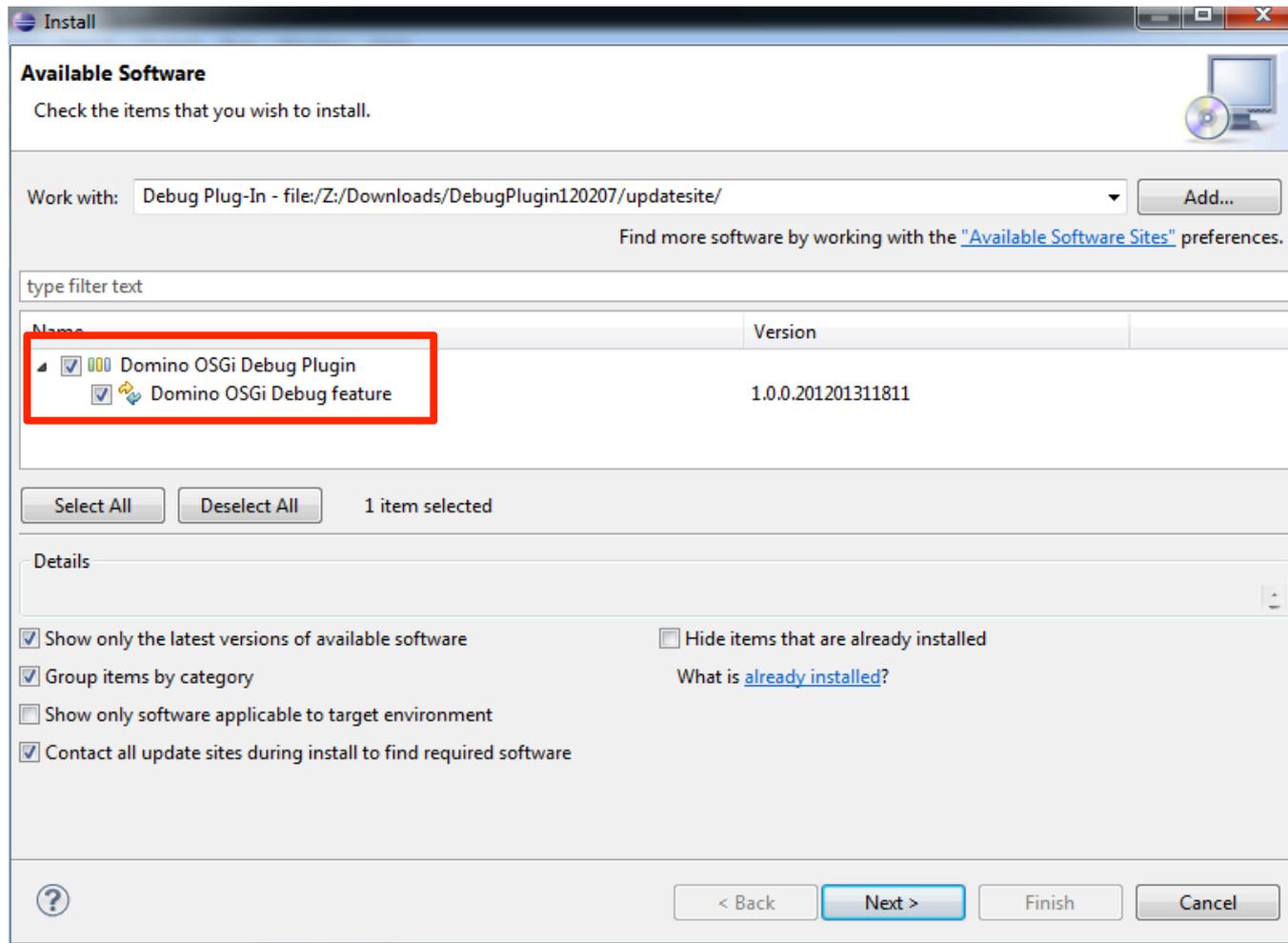
- Installation des Debug-Plug-In (I)



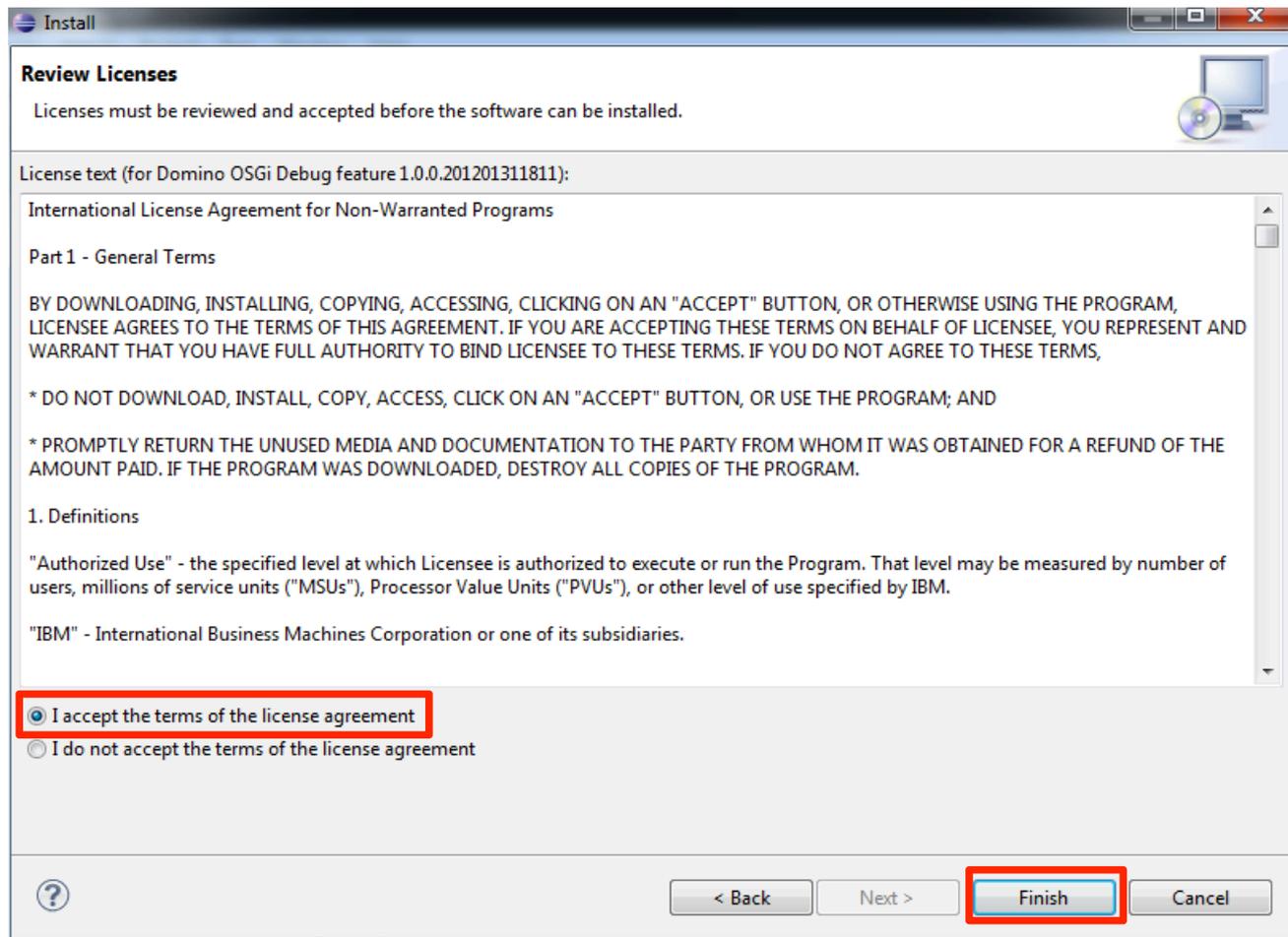
- Installation des Debug-Plug-In (II)



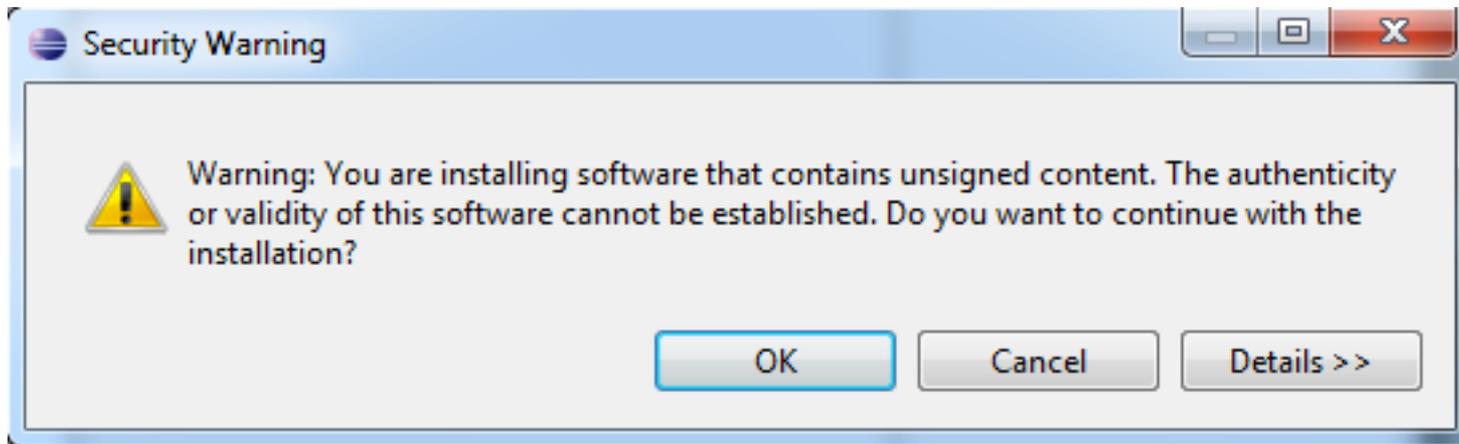
## ■ Installation des Debug-Plug-In (III)



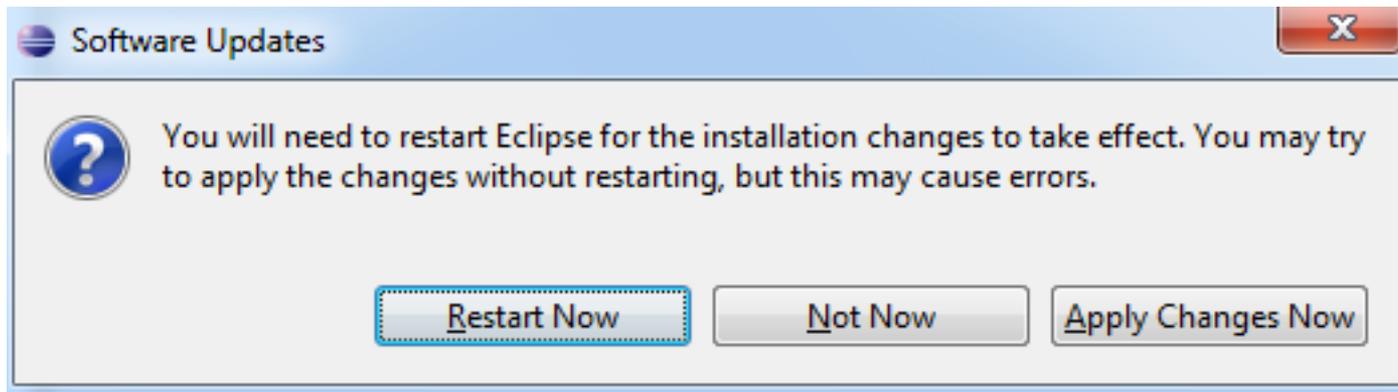
## ■ Installation des Debug-Plug-In (IV)



- Installation des Debug-Plug-In (V)

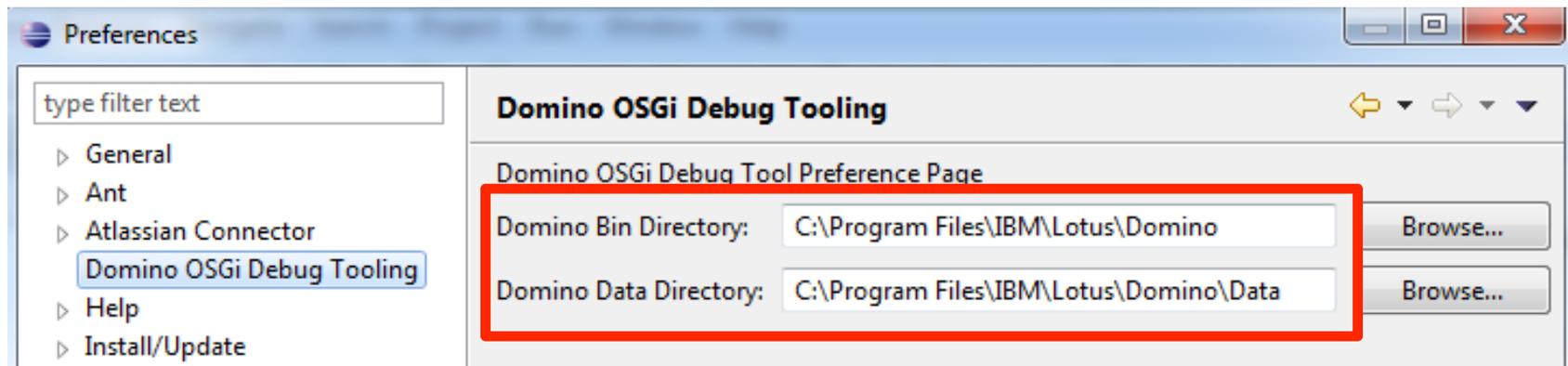


- Installation des Debug-Plug-In (VI)

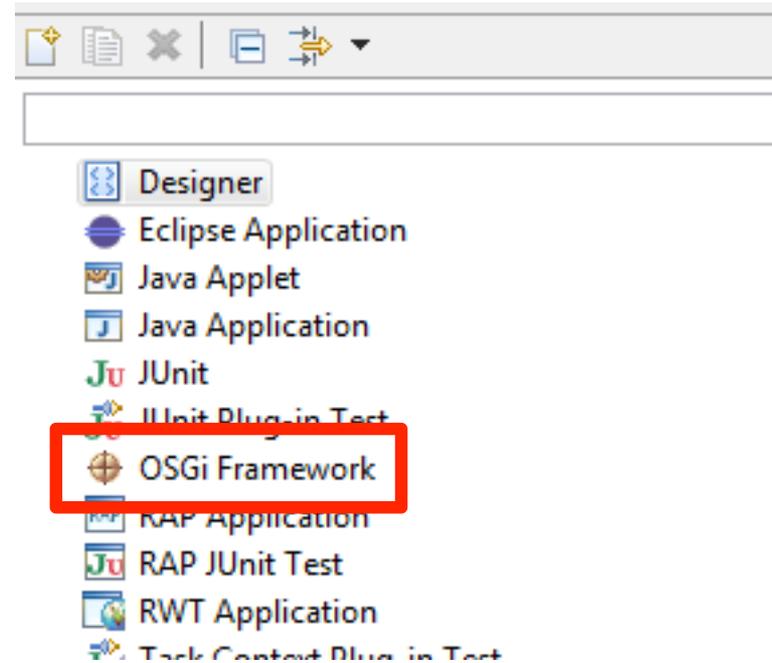
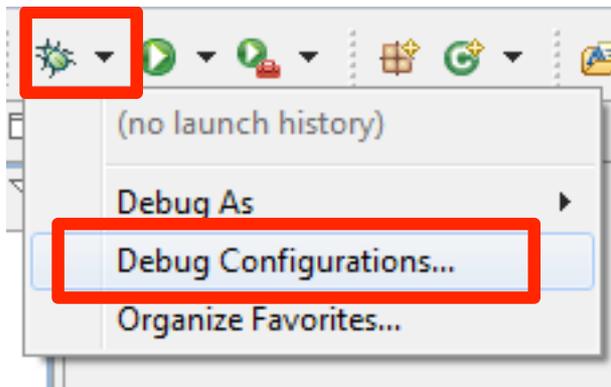


## Setup der Entwicklungsumgebung

- Konfiguration des Debug-Plug-In
  - NUR lokale Domino-Server werden (aktuell) unterstützt

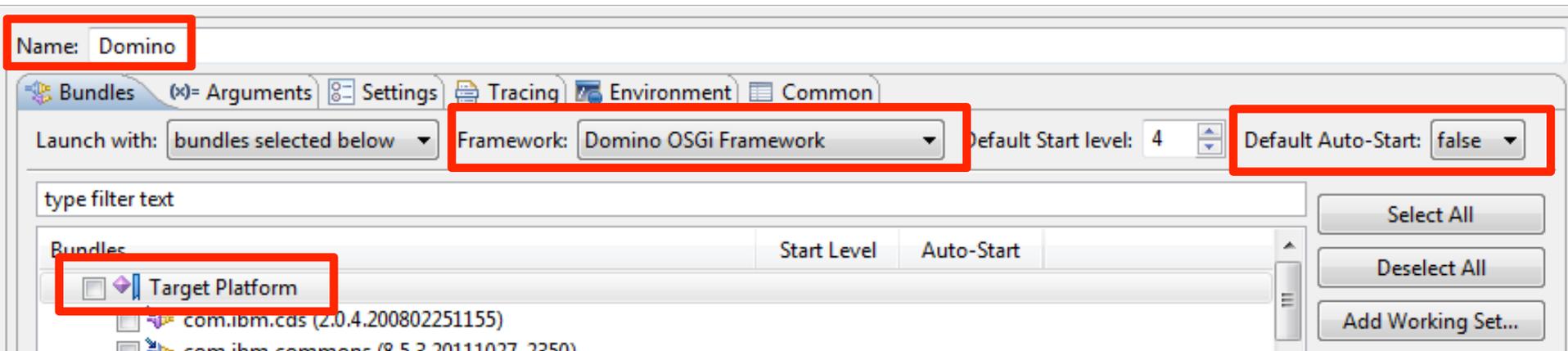


- Erstellung der Debug-Umgebung (I)



## Setup der Entwicklungsumgebung

- Erstellung der Debug-Umgebung (II)



The screenshot shows the Eclipse IDE interface for configuring a bundle. The 'Name' field is set to 'Domino'. The 'Framework' is set to 'Domino OSGi Framework'. The 'Default Start level' is set to 4, and the 'Default Auto-Start' is set to false. The 'Bundles' list shows 'Target Platform' selected, along with other bundles like 'com.ibm.cds' and 'com.ibm.commons'.

Name: Domino

Launch with: bundles selected below

Framework: Domino OSGi Framework

Default Start level: 4

Default Auto-Start: false

type filter text

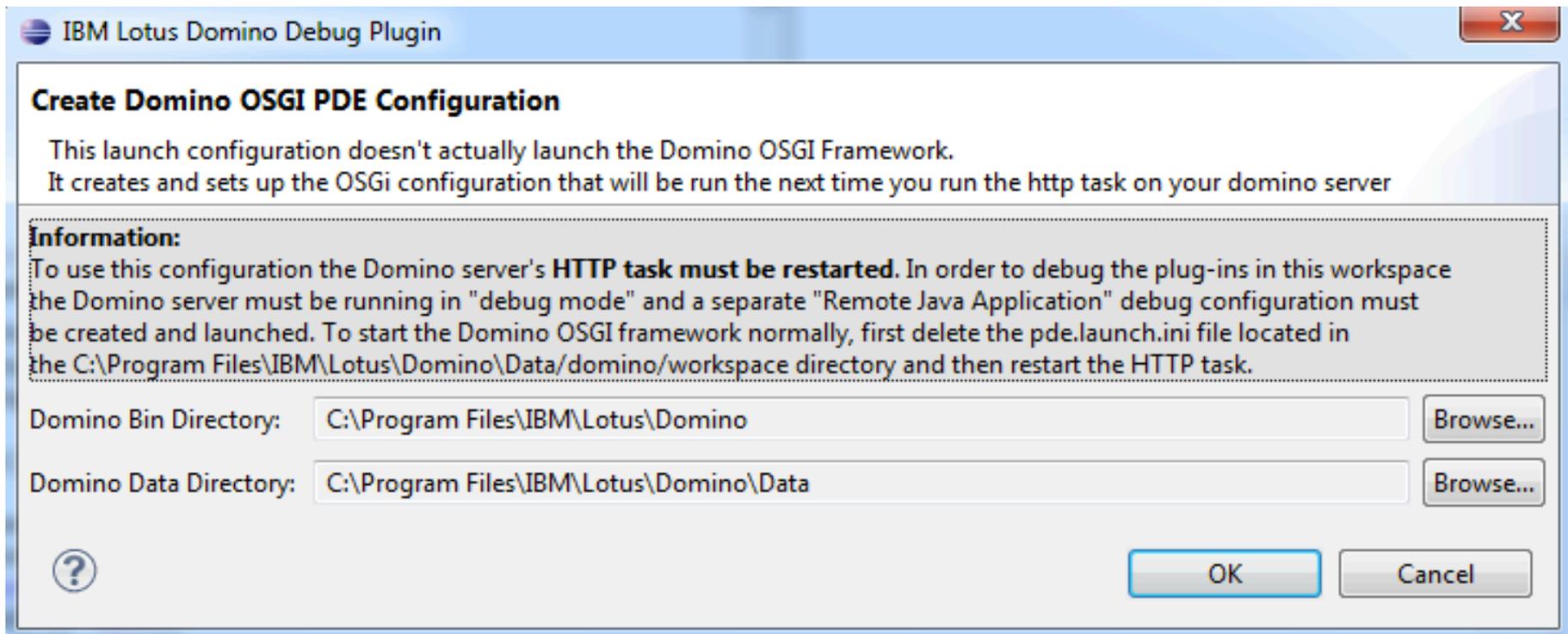
Bundles	Start Level	Auto-Start
<input checked="" type="checkbox"/> Target Platform		
<input type="checkbox"/> com.ibm.cds (2.0.4.200802251155)		
<input type="checkbox"/> com.ibm.commons (8.5.3.20111027-2350)		

Select All  
Deselect All  
Add Working Set...

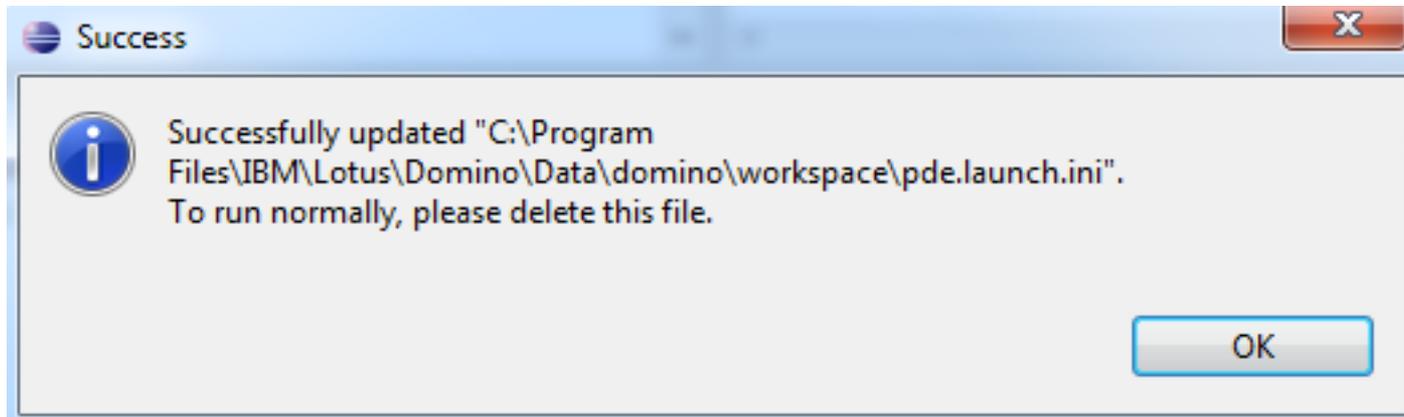


# Setup der Entwicklungsumgebung

- Erstellung der Debug-Umgebung (II)



- Erstellung der Debug-Umgebung (III)



## Setup der Entwicklungsumgebung

- Damit der Domino Remote-Debug akzeptiert, sind noch zwei notes.ini-Parameter zu setzen.
  - JAVADEBUGOPTIONS=transport=dt\_socket,server=y,suspend=n,address=8000
  - JAVAENABLEDEBUG=1
  
- Und wir sind noch nicht ganz fertig – später dazu mehr...



# Demo

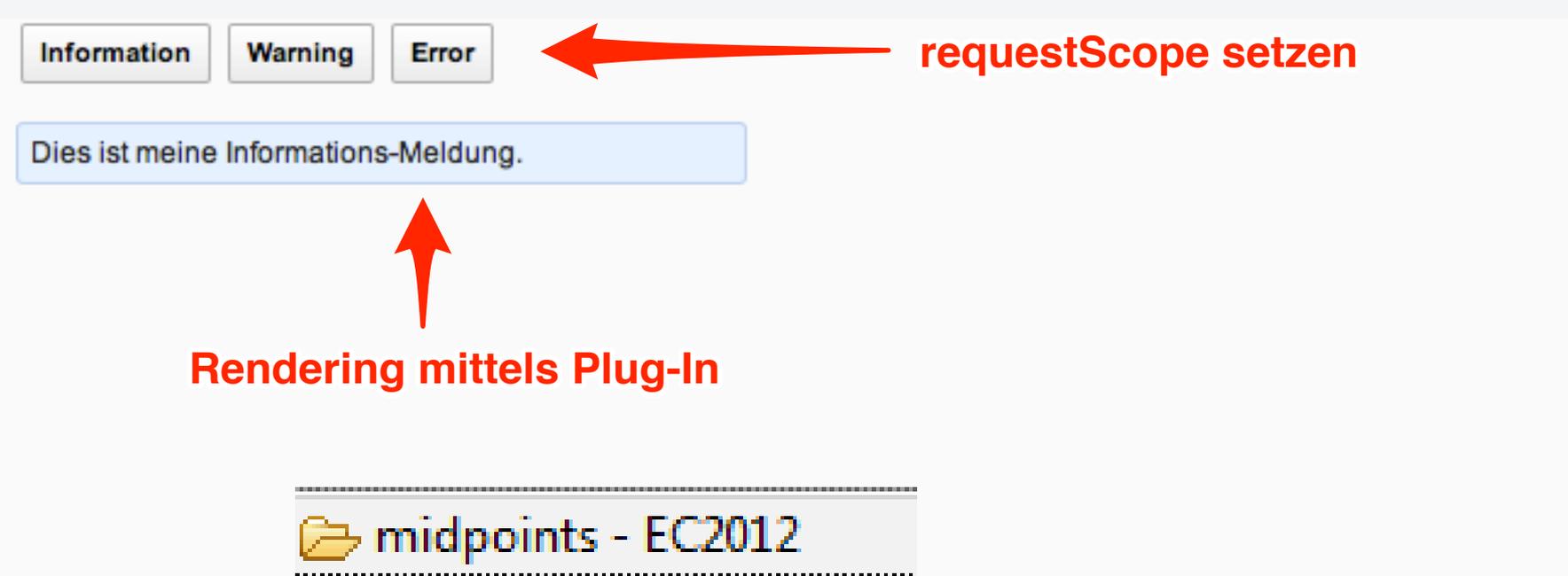


## **Worüber wir heute sprechen werden**

- Unterschiede XPages, Extension Library und Extensibility API
- Vorgehensweise Plug-In-Development
- Codebeispiele
- Deployment

## Codebeispiele

- XPages Extensibility API am Beispiel einer einfach UIComponent.



The screenshot shows a UI component with three buttons labeled "Information", "Warning", and "Error". A red arrow points from the text "requestScope setzen" to the "Error" button. Below the buttons is a blue message box containing the text "Dies ist meine Informations-Meldung.". A red arrow points from the text "Rendering mittels Plug-In" to this message box. At the bottom, a folder icon is next to the text "midpoints - EC2012", with "Message Dialogs" written below it.

Information Warning Error ← requestScope setzen

Dies ist meine Informations-Meldung.

Rendering mittels Plug-In

midpoints - EC2012

Message Dialogs

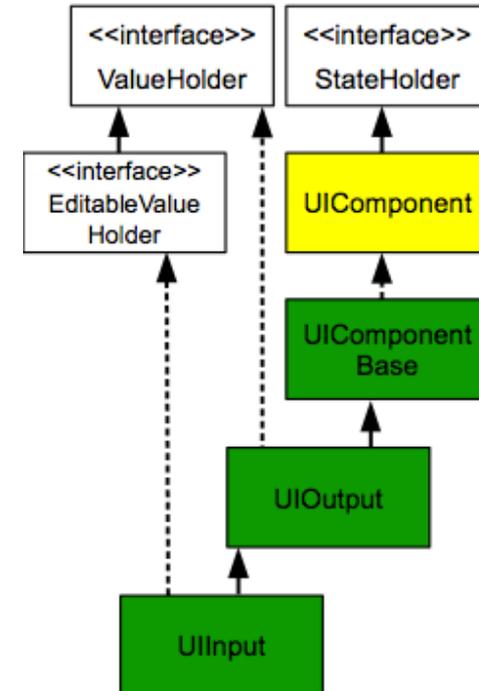


# Demo



## Generelle Struktur einer Library

- Java-Dateien
  - Controls
    - javax.faces.component.UIComponent
    - Basis für alle UI Komponenten
  - Renderers
    - javax.faces.render.Renderer
    - Schreibt den Output zum Browser
- Konfigurationsdateien
  - faces-config.xml
    - Runtime JSF Konfiguration, definiert z. B. den Renderer
  - .xsp-config
    - Definiert die Controls, wird benötigt zur Anzeige im Designer und zum Kompilieren der XPages
  - plugin.xml
    - Andocken an die Erweiterungspunkte (extensions)



## Projektstruktur

- de.midpoints.xsp.messageDialogs
  - JRE System Library [J2SE-1.5]
  - Plug-in Dependencies
  - src
    - de.midpoints.xsp.javascript
    - de.midpoints.xsp.messageDialogs.component
      - MessagesDialog.java
    - de.midpoints.xsp.messageDialogs.library
      - Library.java
    - de.midpoints.xsp.messageDialogs.render
      - MessagesRenderer.java
    - META-INF
      - messages-faces-config.xml
      - messages.xsp-config
  - META-INF
    - MANIFEST.MF
  - build.properties
  - plugin.xml



## MessagesDialog (UIComponent für den Designer)

```
package de.midpoints.xsp.messageDialogs.component;

import javax.faces.component.UIComponentBase;
import javax.faces.context.FacesContext;
import javax.faces.el.ValueBinding;

public class MessagesDialog extends UIComponentBase {

    public MessagesDialog() {
        super();
        setRendererType("de.midpoints.xsp.MessageDialogs");
    }

    @Override
    public String getFamily() {
        return "de.midpoints.xsp.MessageDialogs";
    }
}
```

**Dient der Identifikation.**



# MessagesDialog (UIComponent für den Designer)

```
/*
 * Nur zu Anschauungszwecken vorhanden (Zugriff auf Variable), wird
 * im Projekt nicht genutzt.
 */
private java.lang.String message = null;

public java.lang.String getMessage() {

    ValueBinding _vb = getValueBinding("message"); //$NON-NLS-1$

    if (_vb != null) {
        java.lang.String val = (java.lang.String) _vb.getValue(FacesContext.getCurrentInstance());
        if (val!=null) {
            if (!val.equals("")) {
                message = val;
            }
        }
    }

    return message;
}

public void setMessage(java.lang.String message) {
    this.message = message;
}
```



# MessagesRenderer (Output in der XPage)

```
package de.midpoints.xsp.messageDialogs.render;

import java.io.IOException;
import java.util.Map;

import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.context.ResponseWriter;
import javax.faces.render.Renderer;

import de.midpoints.xsp.messageDialogs.component.MessagesDialog;

public class MessagesRenderer extends Renderer { ←

    public void encodeBegin (FacesContext context, UIComponent component) throws IOException {
        ResponseWriter writer = context.getResponseWriter(); ←
        MessagesDialog dialog = (MessagesDialog)component;
    }
}
```



## MessagesRenderer (Output in der XPage)

```
public void encodeBegin (FacesContext context, UIComponent component)
    throws IOException {
    ResponseWriter writer = context.getResponseWriter();
    MessagesDialog dialog = (MessagesDialog)component;

    String strOutput = getOutputFromScope(); Helfer-Methode
    if (!strOutput.equals("")) {
        writer.startElement("div", component);
        String[] strOutputSplitted = strOutput.split("#");
        writer.writeAttribute("class",
            "lotusMessage %REPLACE%".replace("%REPLACE%",
                strOutputSplitted[0]), null);
        writer.writeText(strOutputSplitted[1], null);
        writer.endElement("div");
    }
}
```



## MessagesRenderer (Output in der XPage)

```
public void encodeEnd (FacesContext context, UIComponent component)
    throws IOException {
    ResponseWriter writer = context.getResponseWriter();
    writer.endElement("div");
}
```



## MessagesRenderer (Output in der XPage)

```
private String getOutputFromScope() {
    String strMessage = "";
    FacesContext context = FacesContext.getCurrentInstance();
    Map viewScopeMap = (Map)context.getApplication().
        getVariableResolver().resolveVariable(context, "requestScope");

    if (viewScopeMap.containsKey("messageInformation")) {
        strMessage = "lotusInfo#" + viewScopeMap.get("messageInformation").toString();
    } else if (viewScopeMap.containsKey("messageWarning")) {
        strMessage = "lotusWarning#" + viewScopeMap.get("messageWarning").toString();
    } else if (viewScopeMap.containsKey("messageError")) {
        strMessage = "lotusError#" + viewScopeMap.get("messageError").toString();
    }

    return strMessage;
}
```



```
package de.midpoints.xsp.messageDialogs.library;

import com.ibm.xsp.library.AbstractXspLibrary;

public class Library extends AbstractXspLibrary {

    public String getLibraryId() {
        return "de.midpoints.xsp.messageDialogs.library";
    }

    public String getPluginId() {
        return "de.midpoints.xsp.messageDialogs";
    }

    @Override
    public String[] getFacesConfigFiles() {
        return new String[]{
            "META-INF/messages-faces-config.xml",
        };
    }

    @Override
    public String[] getXspConfigFiles() {
        return new String[]{
            "META-INF/messages.xsp-config",
        };
    }
}
```

Case-sensitive

Designer  
(UIComponent)

XPages  
(Renderer)



## Dependencies ▶ ⚙️ 🔍 ?

### Required Plug-ins ↓ a z

Specify the list of plug-ins required for the operation of this plug-in.

-  com.ibm.commons
-  com.ibm.xsp.extsn
-  com.ibm.xsp.core

Add...

Remove

Up

Down

Properties...

Total: 3

### Imported Packages ↓ a z

Specify packages on which this plug-in depends without explicitly identifying their originating plug-in.

-  com.ibm.designer.runtime.extensions

(optional, für JSAddin)

Add...

Remove

Properties...

Total: 1

### ▶ Automated Management of Dependencies ↓ a z

### ▶ Dependency Analysis



de.midpoints.xsp.messageDialogs

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
<plugin>
  <extension
    point="com.ibm.commons.Extension">
    <service
      class="de.midpoints.xsp.messageDialogs.library.Library"
      type="com.ibm.xsp.Library">
    </service>
  </extension>
</plugin>
```



**Library-Klasse des Plug-Ins**



# Konfigurationsdateien

- Bis hierhin ist es noch relativ einfach gewesen...oder?
- Nun müssen die Konfigurationsdateien erstellt und bearbeitet werden. Die verwendeten Bezeichnungen sollten sich am vorhandenen Standard orientieren.

**Designer  
(UIComponent)**



```
@Override  
public String[] getFacesConfigFiles() {  
    return new String[]{  
        "META-INF/messages-faces-config.xml",  
    };  
}
```

**XPages  
(Renderer)**



```
@Override  
public String[] getXspConfigFiles() {  
    return new String[]{  
        "META-INF/messages.xsp-config",  
    };  
}
```



# messages-faces-config.xml (XPage-Rendering)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE faces-config PUBLIC
    "-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.0//EN"
    "http://java.sun.com/dtd/web-facesconfig_1_0.dtd">
<faces-config>
  <render-kit>
    <renderer>
      <component-family>de.midpoints.xsp.MessageDialogs
      </component-family>
      <renderer-type>de.midpoints.xsp.MessageDialogs
      </renderer-type>
      <renderer-class>de.midpoints.xsp.messageDialogs.render.MessagesRenderer
      </renderer-class>
    </renderer>
  </render-kit>
</faces-config>
```

**Component-Info!!**

**von Renderer abgeleitete Klasse**

```
public MessagesDialog() {
    super();
    setRendererType("de.midpoints.xsp.MessageDialogs");
}

@Override
public String getFamily() {
    return "de.midpoints.xsp.MessageDialogs";
}
```



## messages.xsp-config

```
<faces-config>  
<faces-config-extension>  
  <namespace-uri>http://www.midpoints.de/xsp/messageDialogs</namespace-uri>  
  <default-prefix>mp</default-prefix>  
</faces-config-extension>
```



mp:messageDialog    /mp:messageDialog



## messages.xsp-config

```
<component>  
  <description>Display custom messages (info, error, message)  
</description>  
  <display-name>Message Dialogs  
</display-name>  
  <component-type>de.midpoints.MessageDialog  
</component-type>  
  <component-class>de.midpoints.xsp.messageDialogs.component.MessagesDialog  
</component-class>
```



midpoints - EC2012

Message Dialogs

Display custom messages  
(info, error, message).



# messages.xsp-config

```

<property>
  <description>The message dialog.</description>
  <display-name>Message</display-name>
  <property-name>message</property-name>
  <property-class>java.lang.String</property-class>
  <property-extension>
    <designer-extension>
      <category>basics</category>
    </designer-extension>
  </property-extension>
</property>

```

Message Dialogs

**All Properties**

Property	Value
▲ <b>basics</b>	
binding	
id	messageDialog1
loaded	
message	
rendered	
rendererType	
▲ <b>styling</b>	
disableTheme	
themeId	

**Message**

---

The message dialog.

From library de.midpoints.xsp.messageDialogs.library  
Since initial version



## messages.xsp-config

```
<component-extension>
  <base-component-type>javax.faces.Component</base-component-type>
  <component-family>de.midpoints.xsp.MessageDialogs</component-family>
  <renderer-type>de.midpoints.xsp.MessageDialogs</renderer-type>
  <tag-name>messageDialog</tag-name>
  <designer-extension>
    <in-palette>true</in-palette>
    <category>midpoints - EC2012</category>
  </designer-extension>
</component-extension>
'component>
```

 midpoints - EC2012

Message Dialogs

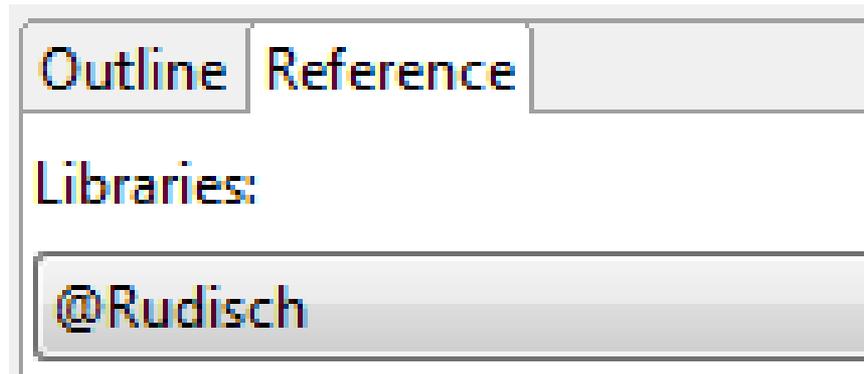


# Demo



## plugin.xml für eigene @Commands

```
<extension point="com.ibm.commons.Extension">  
  <service type="com.ibm.designer.runtime.extensions.JavaScriptProvider"  
    class="de.midpoints.xsp.javascript.JSFunctions" />  
</extension>
```



## Globale Java-Definition für eigene @Commands

```
package de.midpoints.xsp.javascript;  
  
+ import com.ibm.designer.runtime.extensions.JavaScriptProvider;  
  
@SuppressWarnings("restriction")  
  
public class JSFunctions implements JavaScriptProvider {  
-   public void registerWrappers(JSContext jsContext) {  
       jsContext.getRegistry()  
       .registerGlobalPrototype("@Rudisch", new RudischFunctions(jsContext));  
   }  
}
```

Outline Reference

Libraries:

@Rudisch



## Detail-Klasse für eigene @Commands

```
public RudischFunctions(JSContext jsContext) {
    super(jsContext, null, false);
    addFunction(1, "@TranslateToRudisch", "(sentenceToTranslate:T)");
    addFunction(2, "@TranslateFromRudisch", "(sentenceToTranslate:T)");
}

private void addFunction(int index, String functionName, String... params) {
    createMethod(functionName, FBSObject.P_NODELETE | FBSObject.P_READONLY,
        new NotesFunction(getJSContext(), index, functionName, params));
}
```

- @TranslateFromRudisch(sentenceToTranslate:string)
- @TranslateToRudisch(sentenceToTranslate:string)

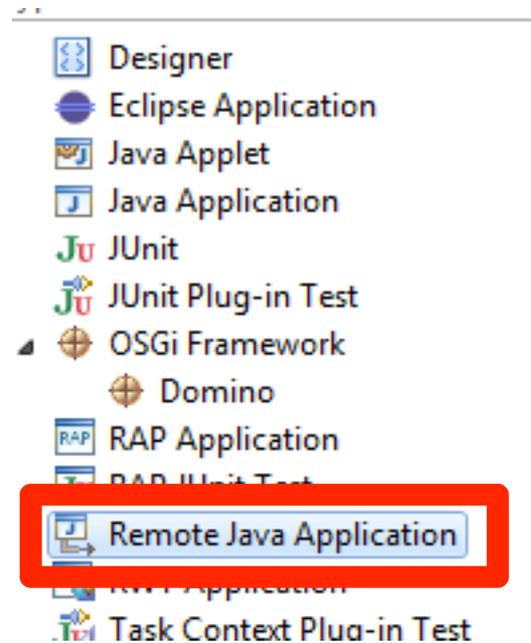
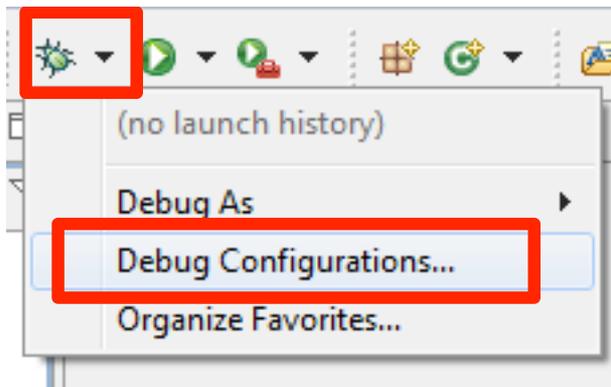


# Demo



# D - E - B - U - G - G - I - N - G

- Einrichtung einer Remote Java Application zum Debugging.



# D - E - B - U - G - G - I - N - G

Name: RemoteDebug

Connect Source Common

Project:  
de.midpoints.xsp.messageDialogs Browse...

Connection Type:  
Standard (Socket Attach)

Connection Properties:  
Host: localhost  
Port: 8000

Allow termination of remote VM

Apply Revert

Debug Close



# Demo



## **Worüber wir heute sprechen werden**

- Unterschiede XPages, Extension Library und Extensibility API
- Vorgehensweise Plug-In-Development
- Codebeispiele
- Deployment

- Für das Deployment von Plug-Ins in den Notes-Client möchte ich auf die EC2012-Slides meines Kollegen Detlev Pöttgen verweisen.



## Deployment Domino-Server

- Auf Basis des Updatesite-Templates ist eine neue Datenbank zu erstellen bzw. kann eine vorhandene mitgenutzt werden. Für reine Server-Plug-Ins empfehle ich eine separate Datenbank!
- Die Plug-Ins sind die Updatesite-Datenbank zu importieren.
- Die Datenbank ist auf alle erforderlichen Server zu replizieren.
  - Plug-Ins an sich kann man nicht replizieren, da es sich um „normales“ File-System handelt.
- Der notes.ini-Parameter `OSGI_HTTP_DYNAMIC_BUNDLES=<datenbank.nsf>` ist auf allen Domino-Server zu setzen.
- Fertig. Die Plug-Ins werden bei HTTP-Start automatisch installiert.



# Demo



Vielen Dank für Ihre Aufmerksamkeit!



René Winkelmeier

Skype/Twitter/LinkedIn/Facebook: muenzpraeger

<http://blog.winkelmeier.com>

[http://www.xing.de/Rene\\_Winkelmeier](http://www.xing.de/Rene_Winkelmeier)

[rene.winkelmeier@midpoints.de](mailto:rene.winkelmeier@midpoints.de) / [mail@winkelmeier.com](mailto:mail@winkelmeier.com)

midpoints | purify it

<http://www.midpoints.de>

[info@midpoints.de](mailto:info@midpoints.de)