midpoints
purify it

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

midpoints
purify it

Worüber wir heute sprechen werden

- Unterschiede XPages, Extension Library und Extensibility API

- Vorgehensweise Plug-In-Development

- Codebeispiele

- Deployment

midpoints
purify it

Worüber wir heute sprechen werden

- Unterschiede XPages, Extension Library und Extensibility API

- Vorgehensweise Plug-In-Development

- Codebeispiele

- Deployment

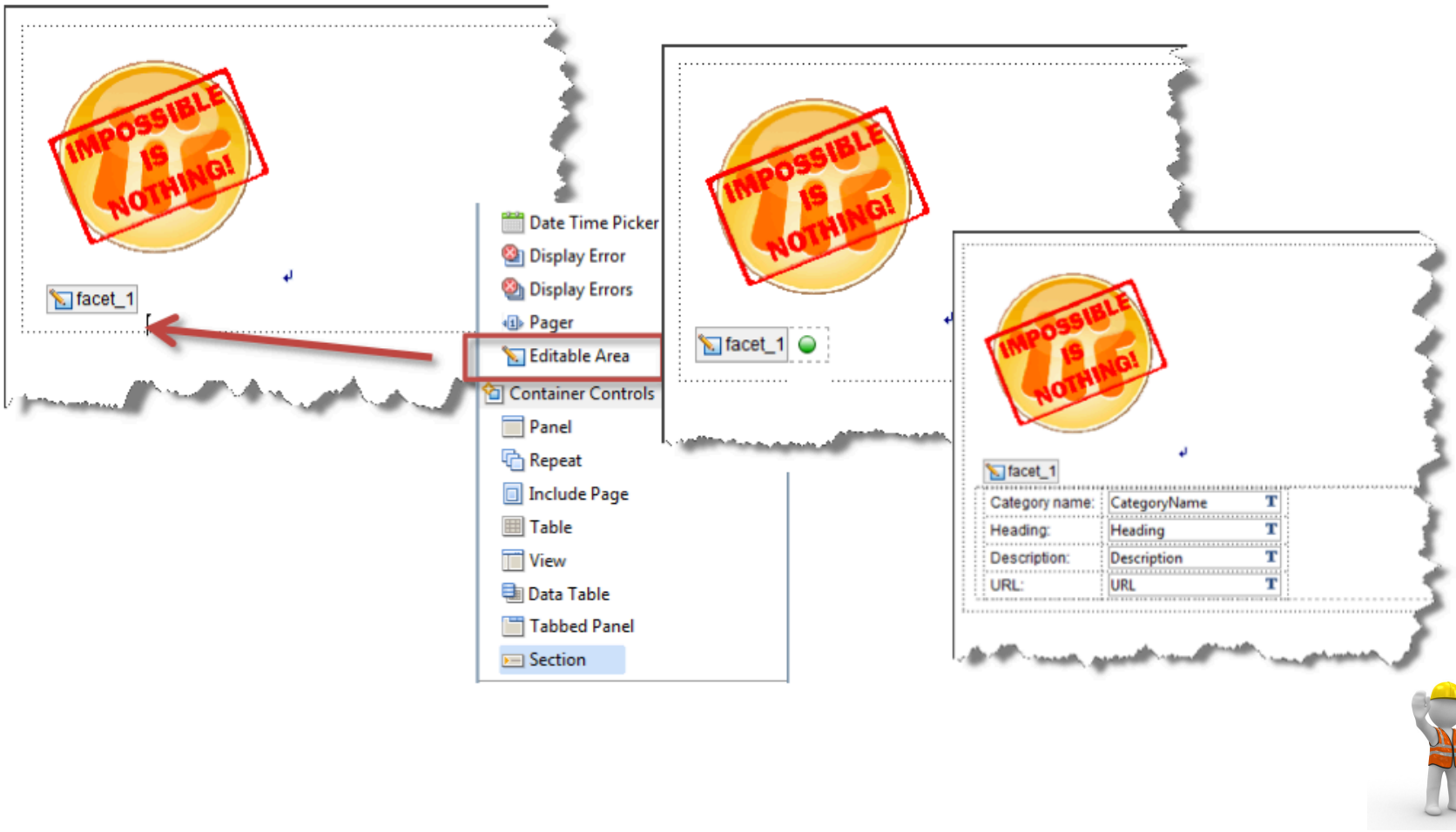
XPages

- 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.

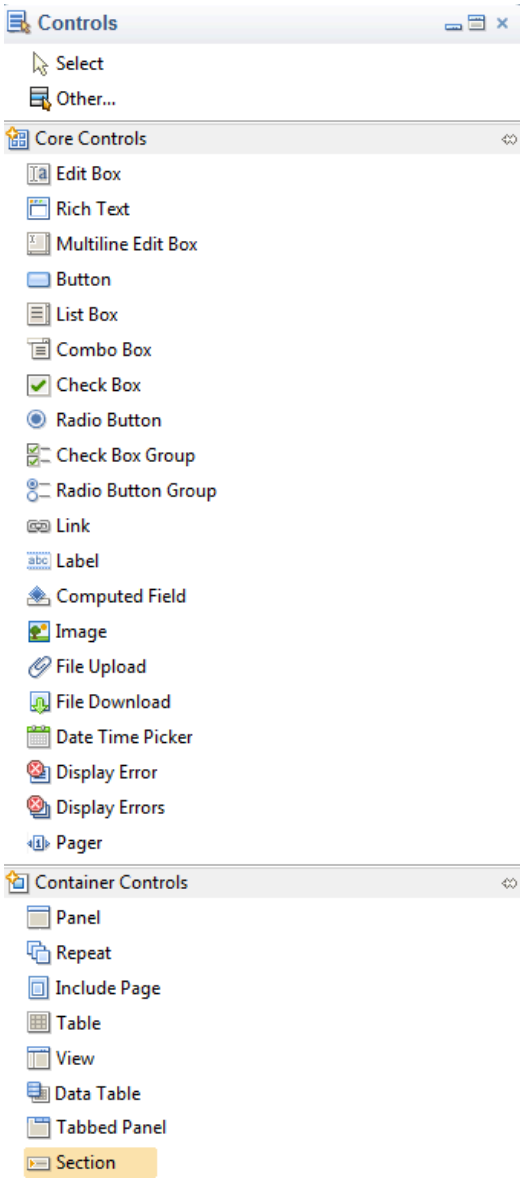


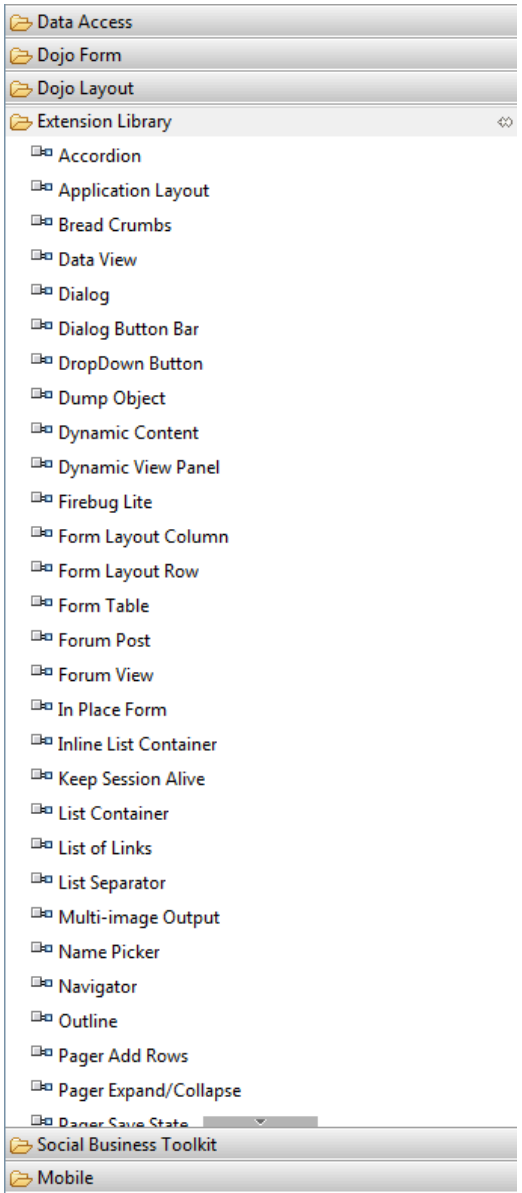
midpoints
purify it

- 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

midpoints
purify it





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

midpoints purify it

- Data Access
  - JDBC Connection Manager
  - Remote Service
  - REST Service
- Dojo Form
  - Dojo Button
  - Dojo Check Box
  - Dojo Combo Box
  - Dojo Currency Text Box
  - Dojo Date Text Box
  - Dojo Filtering Select
  - Dojo Horizontal Slider
  - Dojo Number Spinner
  - Dojo Number Text Box
  - Dojo Radio Button
  - Dojo Simple Text Area
  - Dojo Slider Rule
  - Dojo Slider Rule Labels
  - Dojo Text Area
  - Dojo Text Box
  - Dojo Time Text Box
  - Dojo Toggle Button
  - Dojo Validation Text Box
  - Dojo Vertical Slider
  - Dojo Image Select
  - Dojo Link Select
  - Dojo List Text Box
  - Dojo Name Text Box
- Dojo Layout
  - Dojo Accordion Container
  - Dojo Accordion Pane
  - Dojo Border Container
  - Dojo Border Pane
  - Dojo Content Pane
  - Dojo Data Grid
  - Dojo Data Grid Column
  - Dojo Data Grid Row
  - Dojo Stack Container
  - Dojo Stack Pane
  - Dojo Tab Container
  - Dojo Tab Pane
- Extension Library
  - Accordion
  - Application Layout
  - Bread Crumbs
  - Data View
  - Dialog
  - Dialog Button Bar
  - DropDown Button
  - Dump Object
  - Dynamic Content
  - Dynamic View Panel
  - Firebug Lite
  - Form Layout Column
  - Form Layout Row
  - Form Table
  - Forum Post
  - Forum View
  - In Place Form
  - Inline List Container
  - Keep Session Alive
  - List Container
  - List of Links
  - List Separator
  - Multi-image Output
  - Name Picker
  - Navigator
  - Outline
  - Pager Add Rows
  - Pager Expand/Collapse
  - Pager Save State
  - Pager Show/Hide Details
  - Pager Sizes
  - Pop-up Menu
  - Sort Links
  - Switch
  - Tag Cloud
  - Toolbar
  - Tooltip
  - Tooltip Dialog
  - Value Picker
  - Widget Container
  - Dialog Content
- Social Business Toolkit
  - Connections Client
  - Connections Widget
  - Sametime Client
  - Sametime Widget
- Mobile
  - Mobile Page
  - Mobile Switch
  - Page Heading
  - Rounded List
  - Single Page Application
  - Static Line Item
  - Tab Bar
  - Tab Bar Button

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	ExtensionLibraryOpenNTF-853.20120126-0415.zip
Dec 21, 2011	Philippe Riand	853.20111221-0529b	1,318	ExtensionLibraryOpenNTF-853.20111221-0529b.zip
Dec 16, 2011	Philippe Riand	853-20111215-0914c	331	ExtensionLibraryOpenNTF-853.20111215-0914c.zip
Nov 19, 2011	Philippe Riand	853-20111117-1058	1,298	ExtensionLibraryOpenNTF-853-20111117-1058.zip
Oct 28, 2011	Philippe Riand	853.20111027-1245a	1,028	ExtensionLibraryOpenNTF-853.20111027-1245a.zip
Oct 28, 2011	Philippe Riand	8.5.2.201110260152NTF	683	XPagesExtensionLibrary-852.zip
Oct 28, 2011	Philippe Riand	8.5.2.201110260152NTF	683	XPagesExtensionLibrary-852.zip

XPages Extension Library

midpoints
purify it

- 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

midpoints purify it

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

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

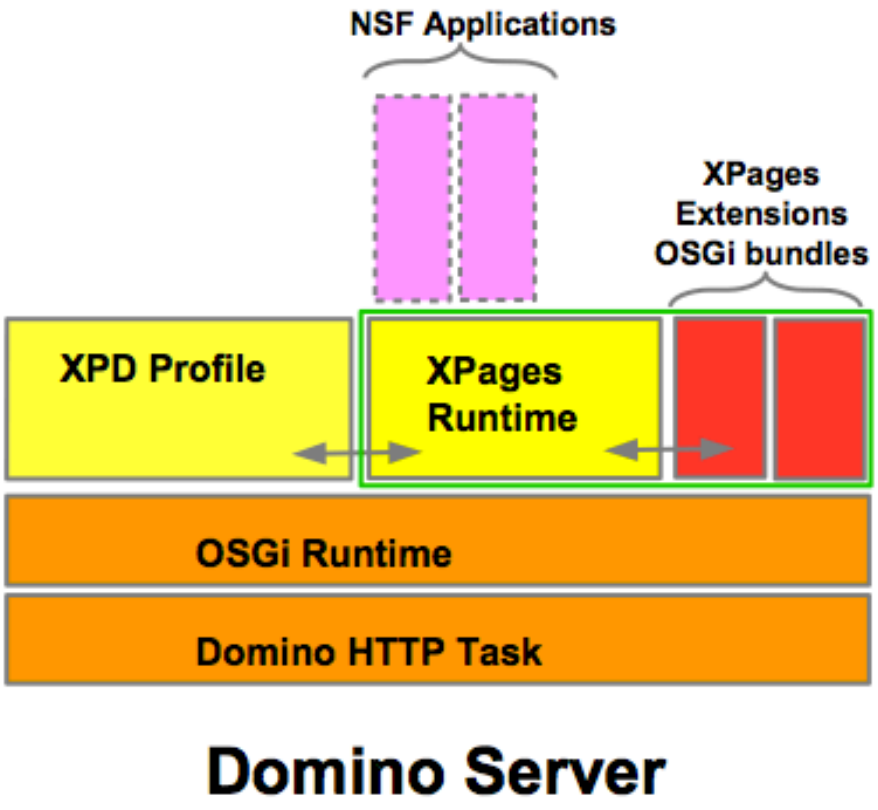
XPages Extensibility API

midpoints
purify it

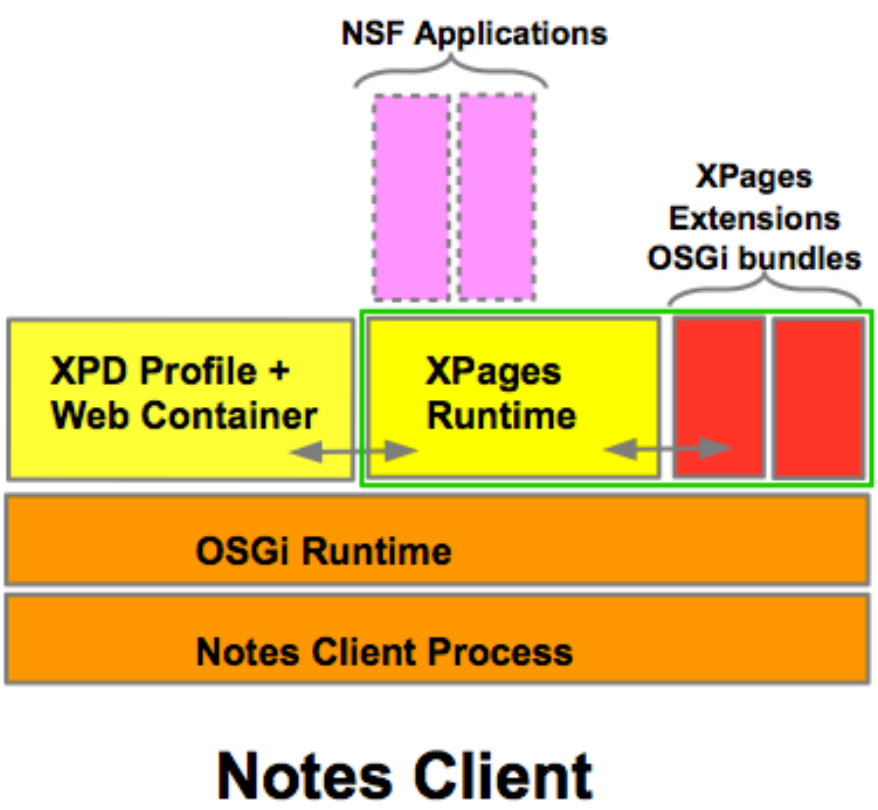
- So genannte „Artefakte“ könne über die vorhandene API zur Runtime hinzugefügt werden.
- JSF ist ein „offener Standard“ dessen APIs öffentlich verfügbar sind.
- Seit Domino 8.5.2 sind die XPages APIs veröffentlicht worden.
- Artefakte können sein:
  - UI Controls
  - Converters
  - Validators
  - Data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

XPages Architektur

midpoints
purify it



Domino Server



Notes Client

XPages API Konzepte

midpoints
purify it

- 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.

midpoints
purify it

Worüber wir heute sprechen werden

- Unterschiede XPages, Extension Library und Extensibility API

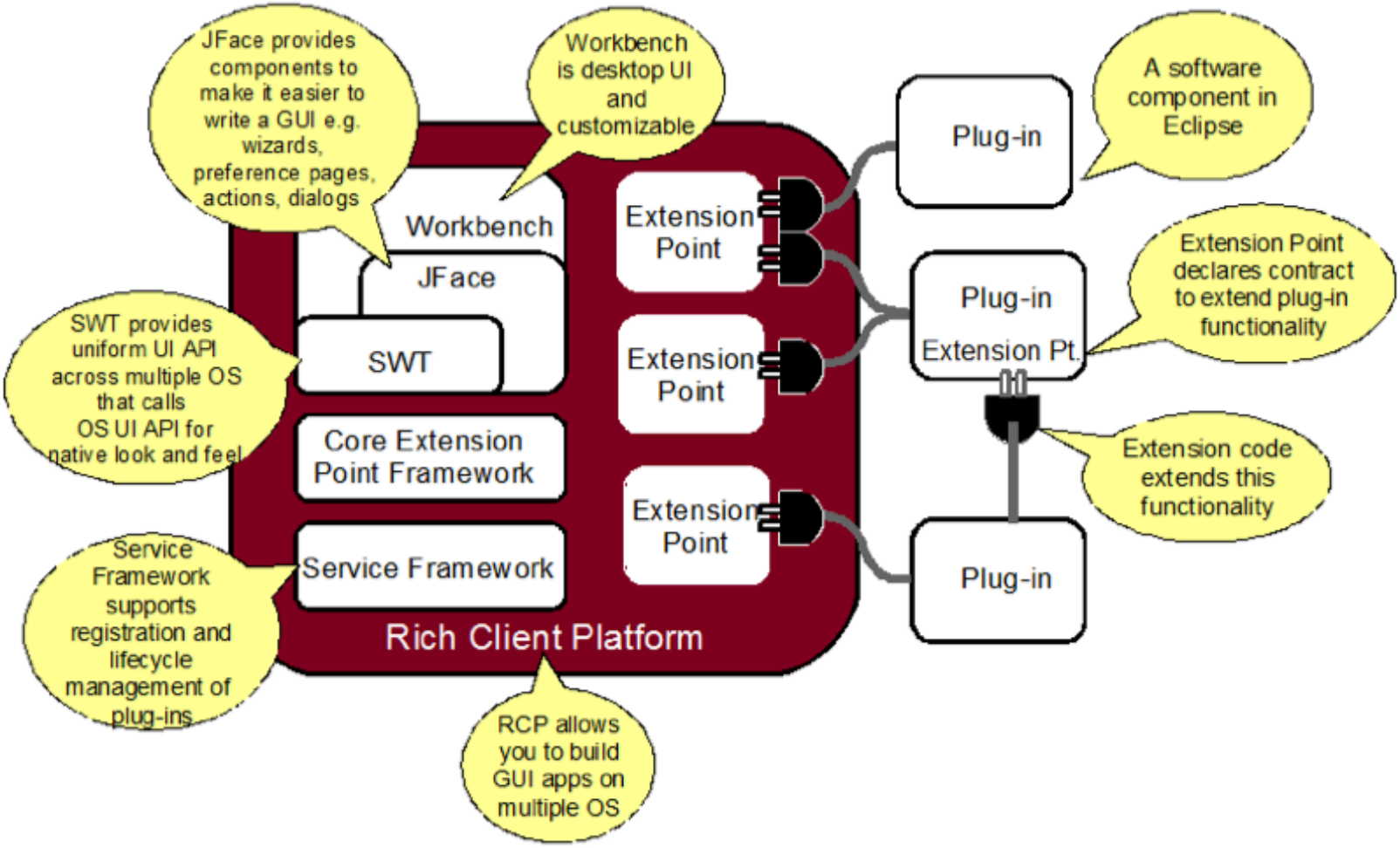
- Vorgehensweise Plug-In-Development

- Codebeispiele

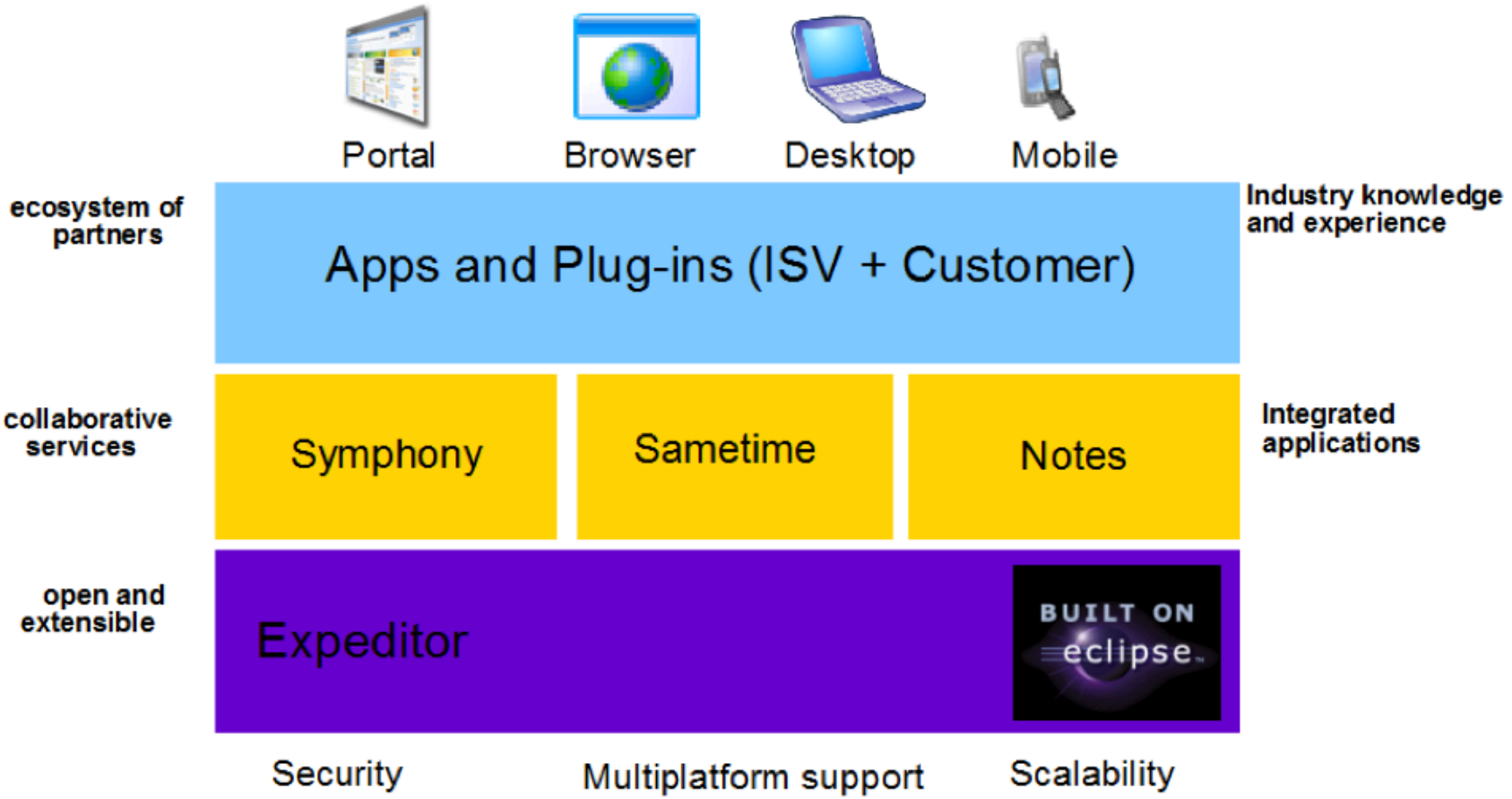
- Deployment

Powered By Eclipse

Eclipse Plug-In-Architektur

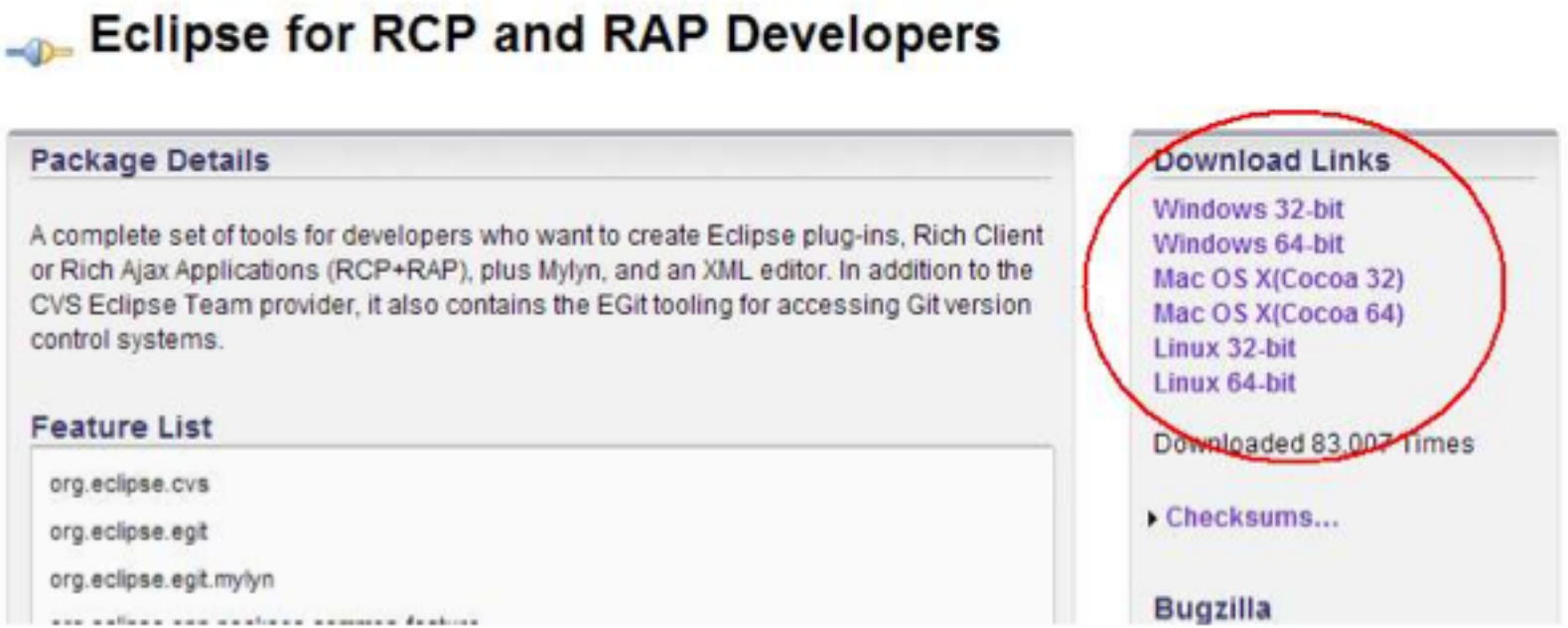


IBM Clients/Server auf Basis von Eclipse



Setup der Entwicklungsumgebung

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



Setup der Entwicklungsumgebung

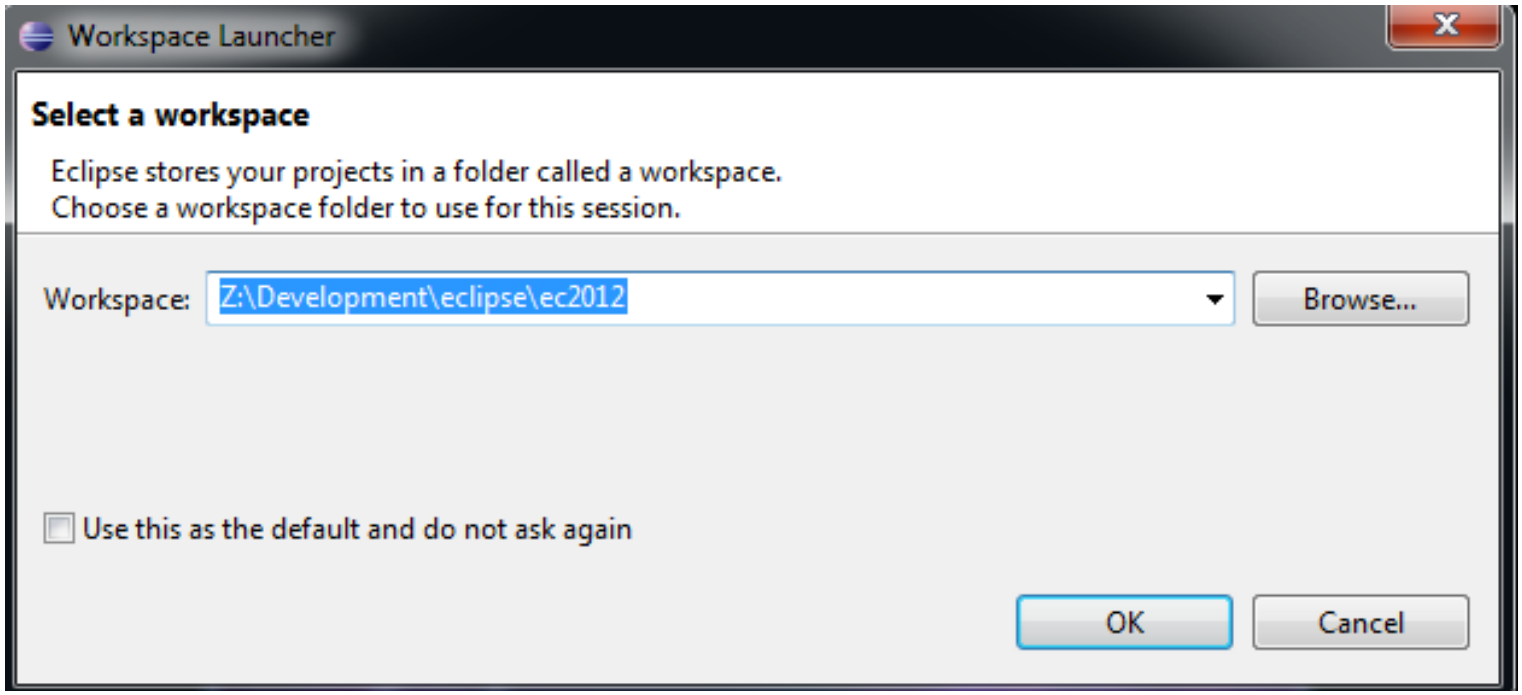
- Zur Vereinfachung setzen wir das „XPages SDK for Eclipse RCP“ ein.
  - Bereitgestellt auf OpenNTF durch Nathan T. Freeman: http://www.openntf.org/internal/home.nsf/project.xsp?action=openDocument&name=XPages%20SDK%20for%20Eclipse%20RCP

XPages SDK for Eclipse RCP

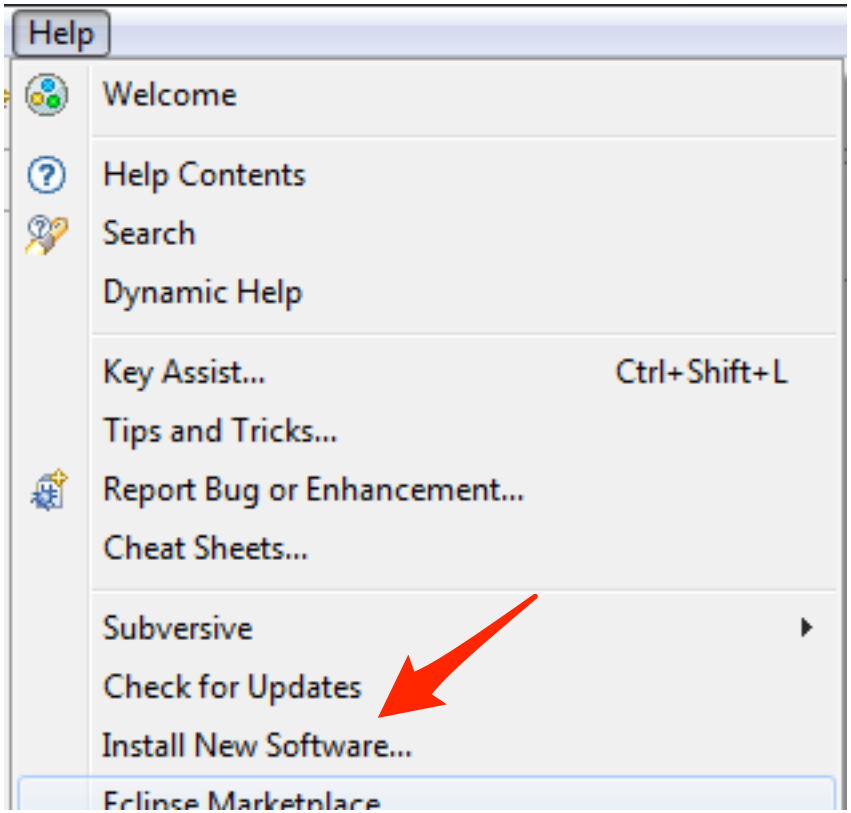
Owners	Nathan T Freeman, Niklas Heidloff	Category	Admin Utilities
Contributors	Niklas Heidloff	Platform	R 8.5.2
Downloads	256 Download latest release	Last Release	Feb 7, 2012
Rating	☆☆☆☆☆ (0 ratings)	Project Creation	Jan 10, 2012
Status	Active	Short URL	Not defined
Description	XPages SDK for Eclipse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

Setup der Entwicklungsumgebung

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

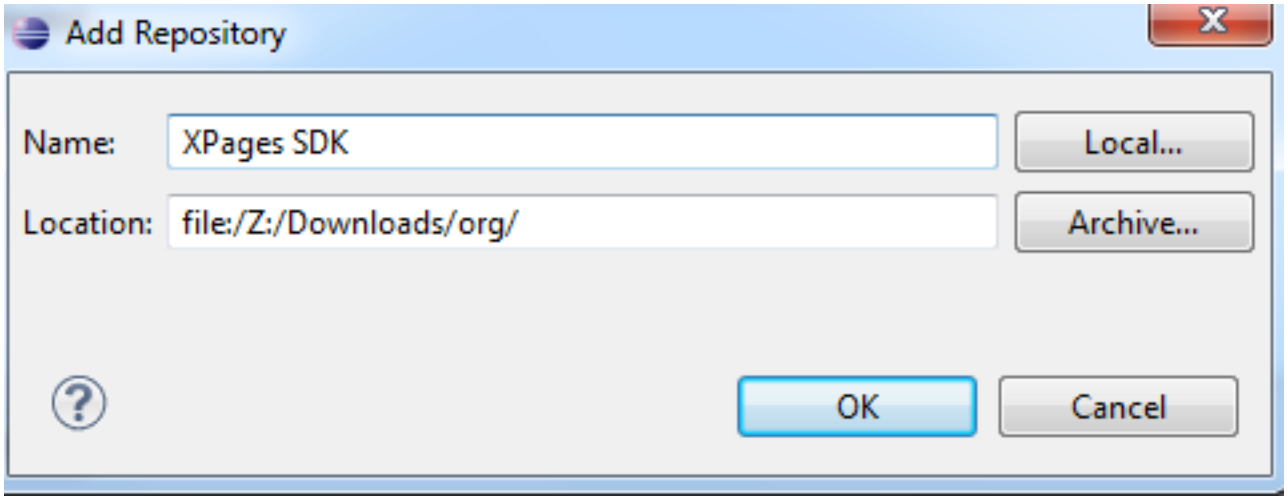


- Installation neuer Software innerhalb von Eclipse

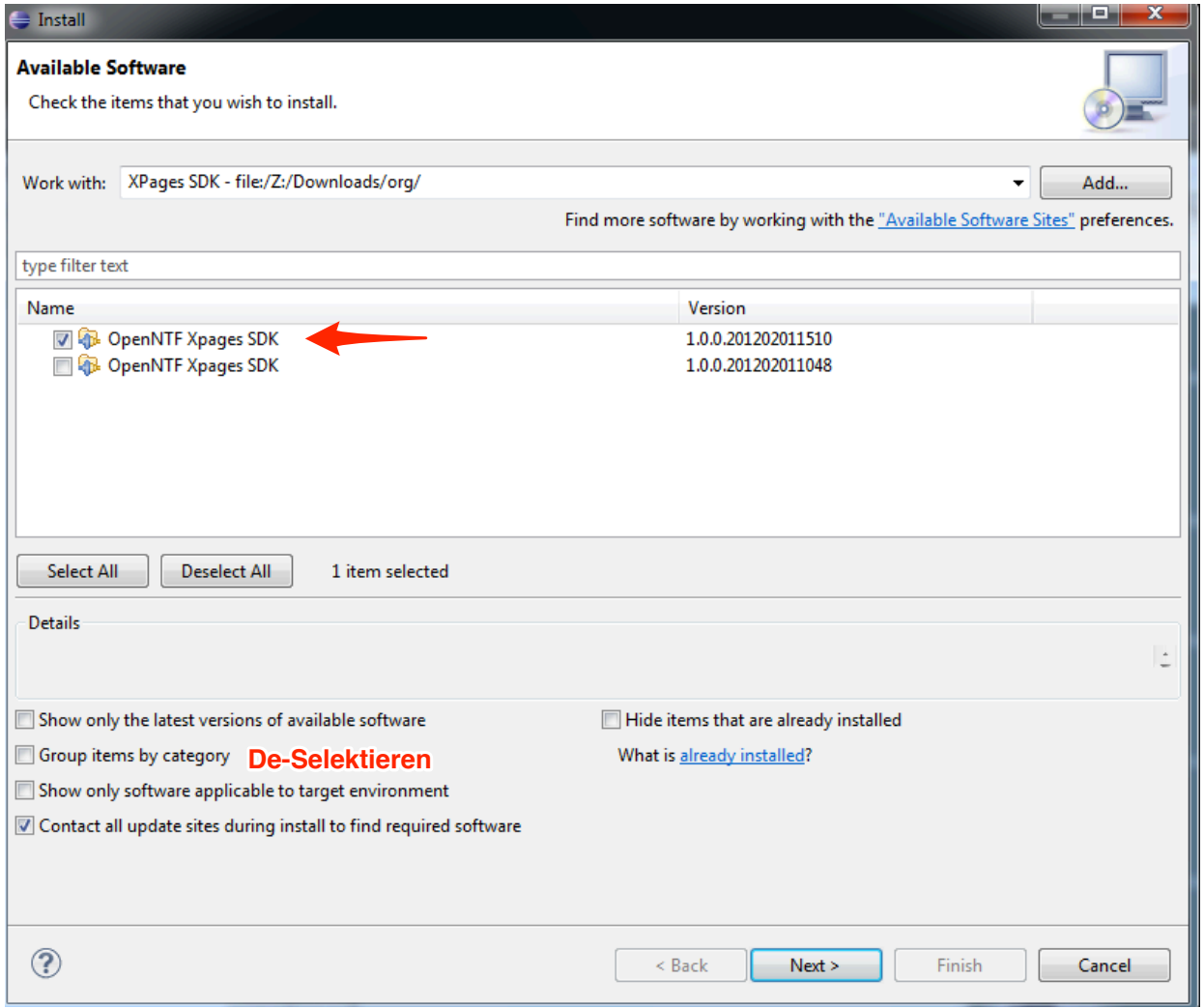


Setup der Entwicklungsumgebung

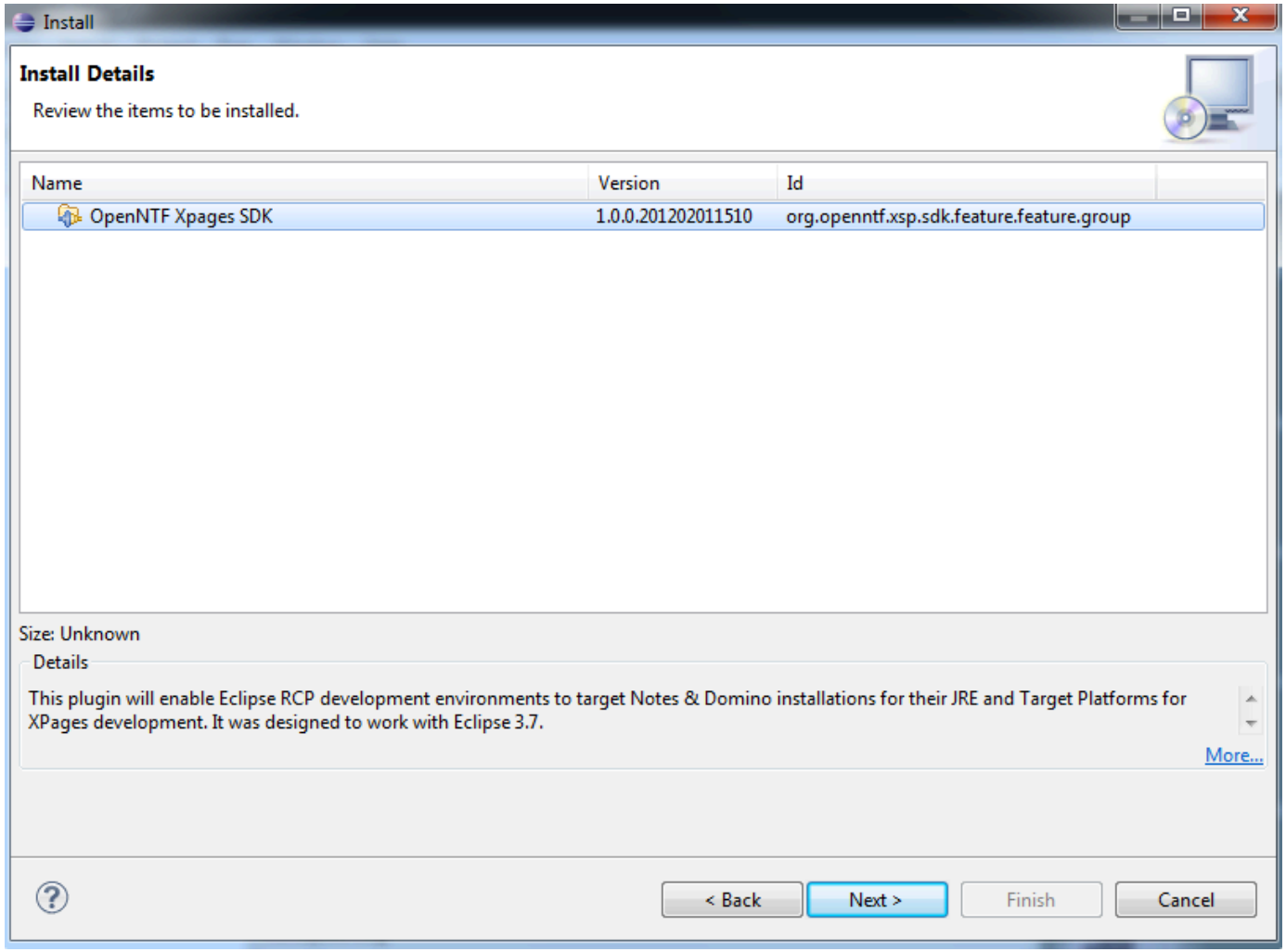
- Auswahl des XPages SDK



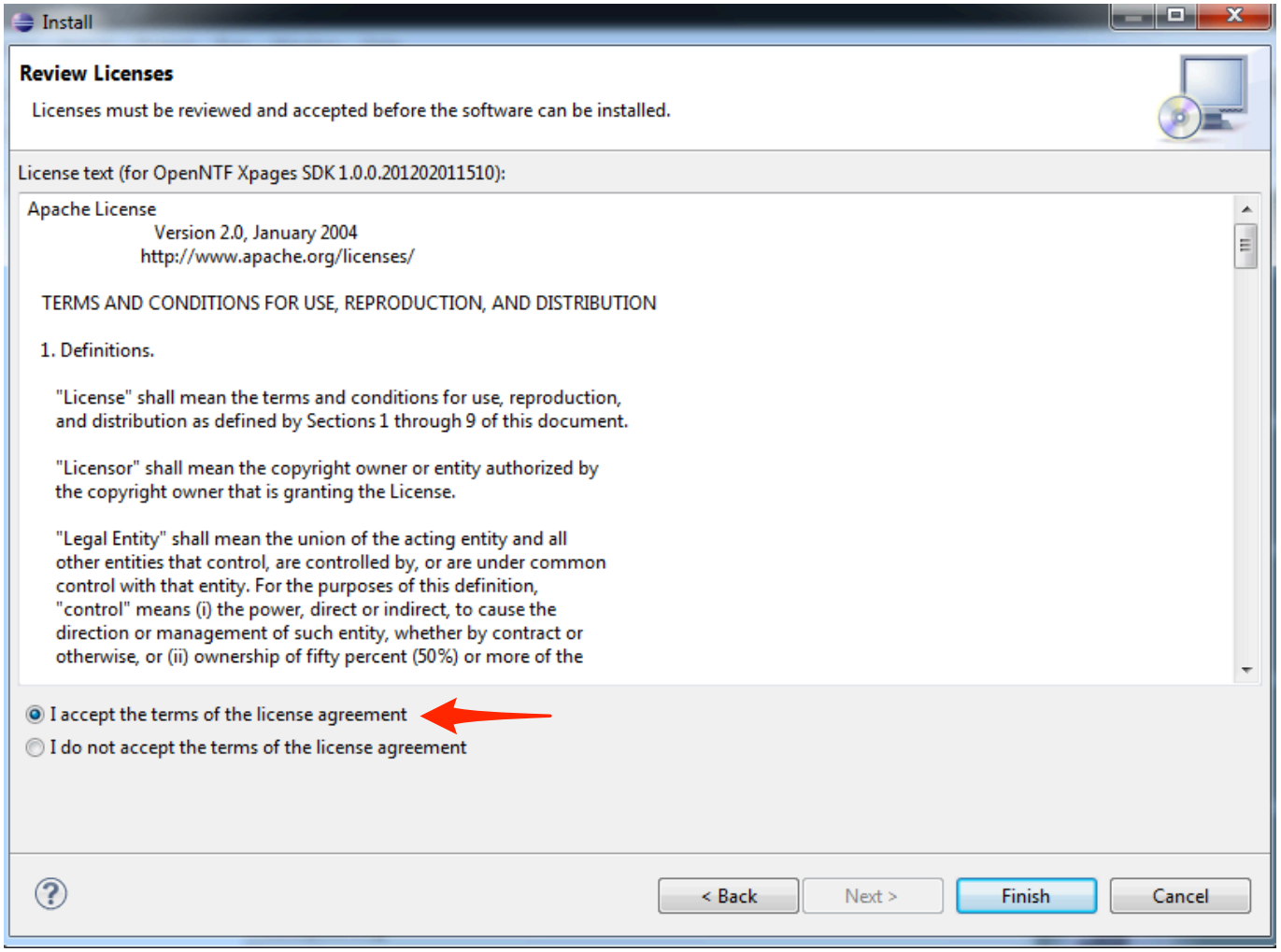
- Installation des XPages SDK (I)



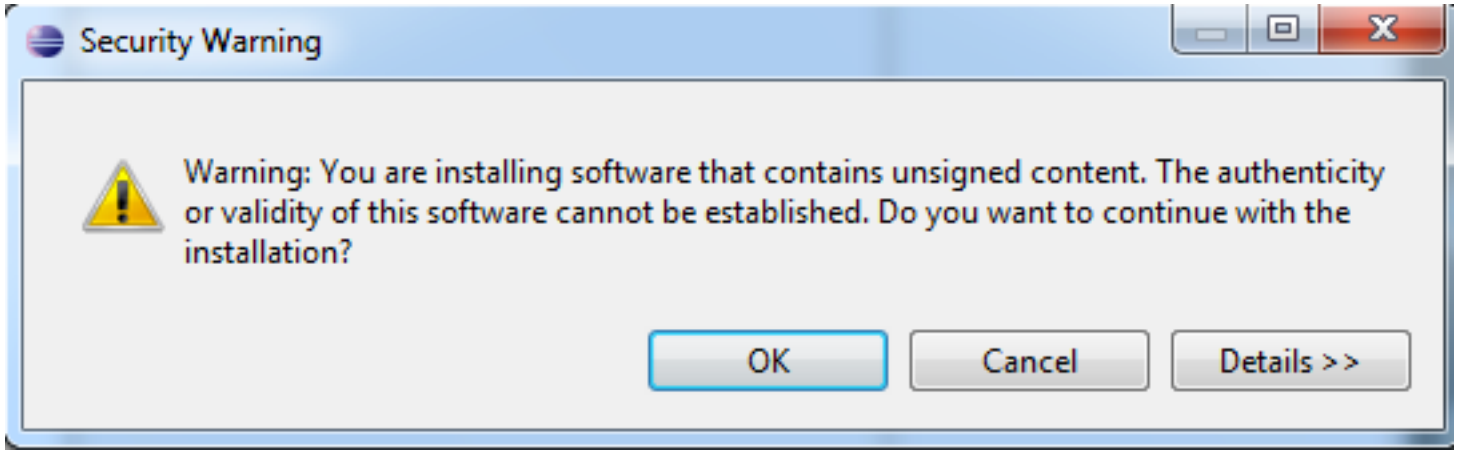
- Installation des XPages SDK (II)



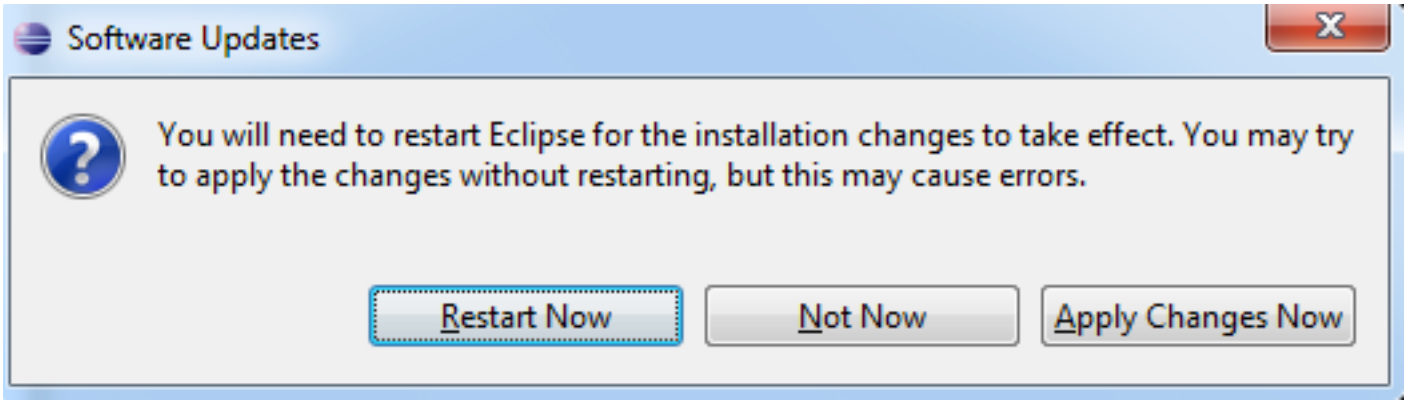
- Installation des XPages SDK (III)



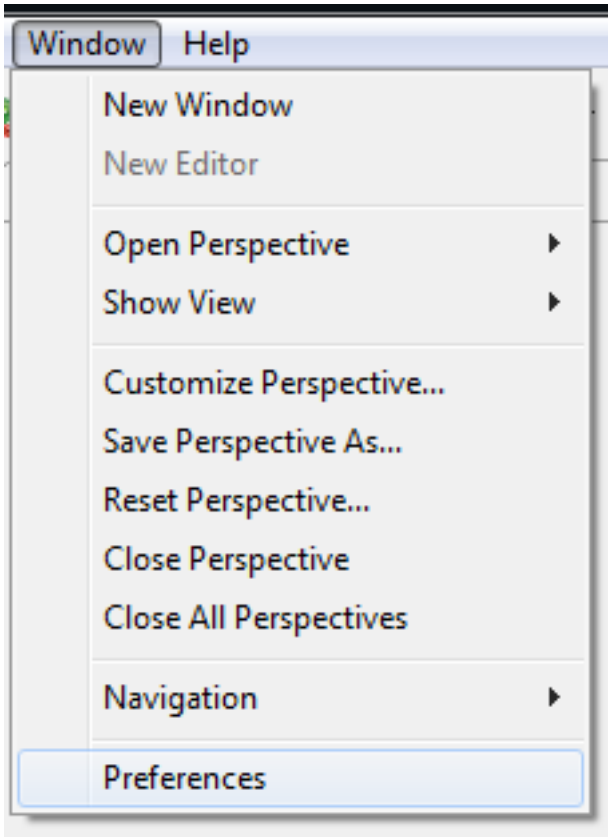
- Installation des XPages SDK (IV)
  - Unsigniert ist ok – es bedeutet, dass die Plug-Ins nicht digital signiert wurden

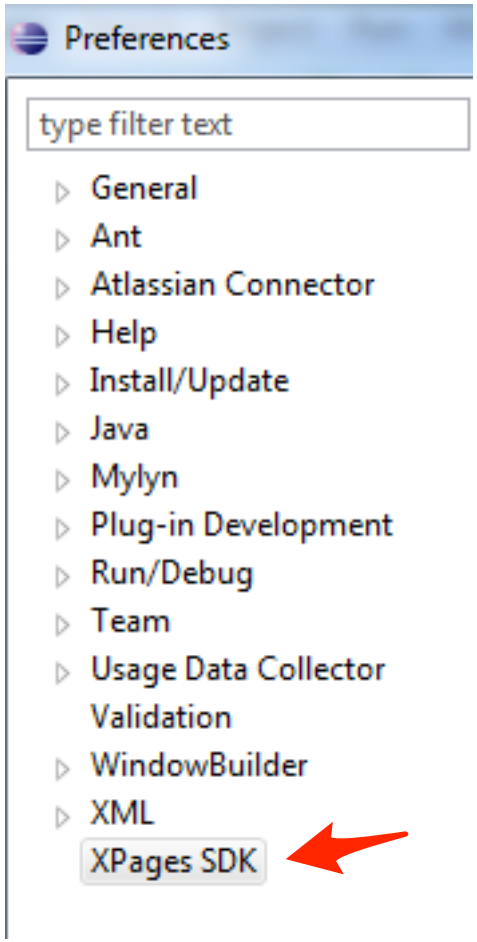


- Installation des XPages SDK (V)

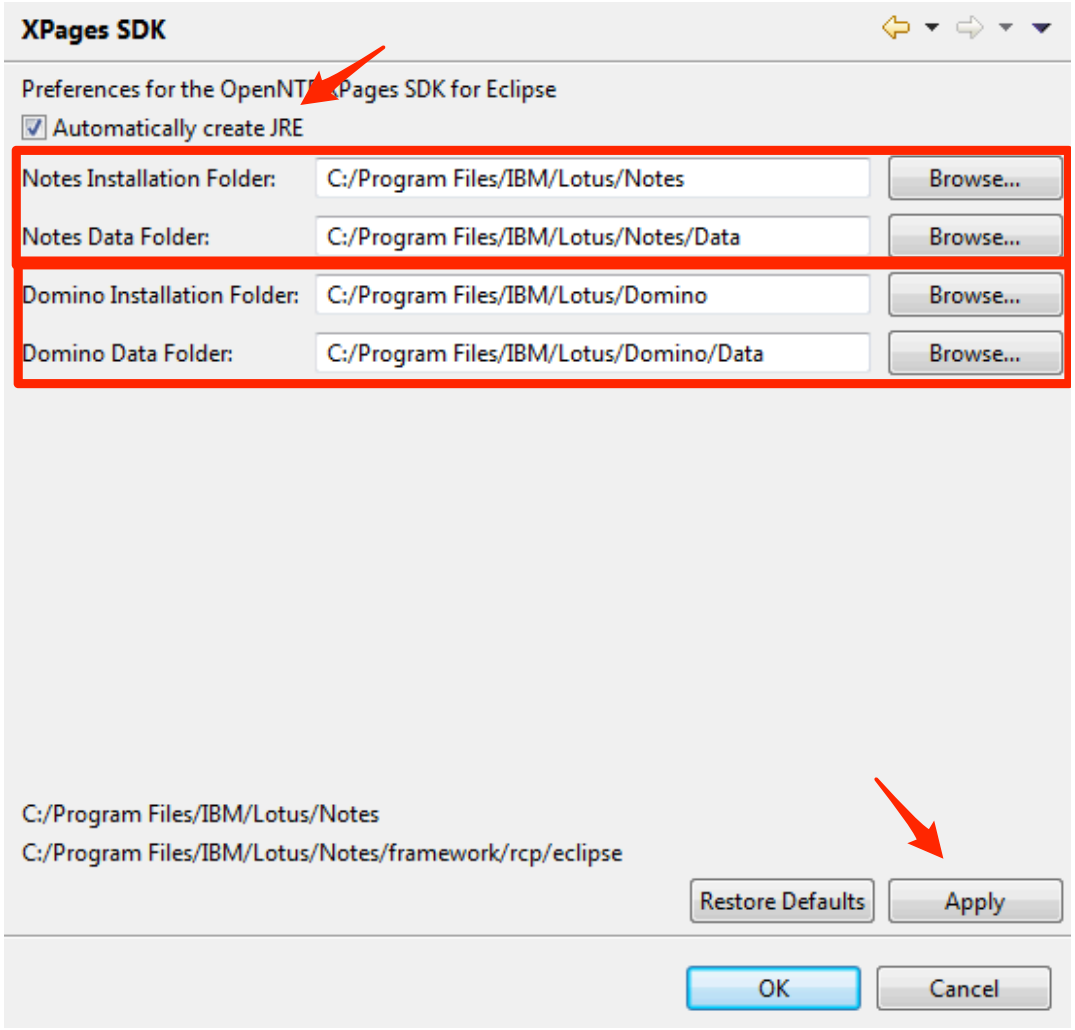


- Konfiguration des XPages SDK (I)

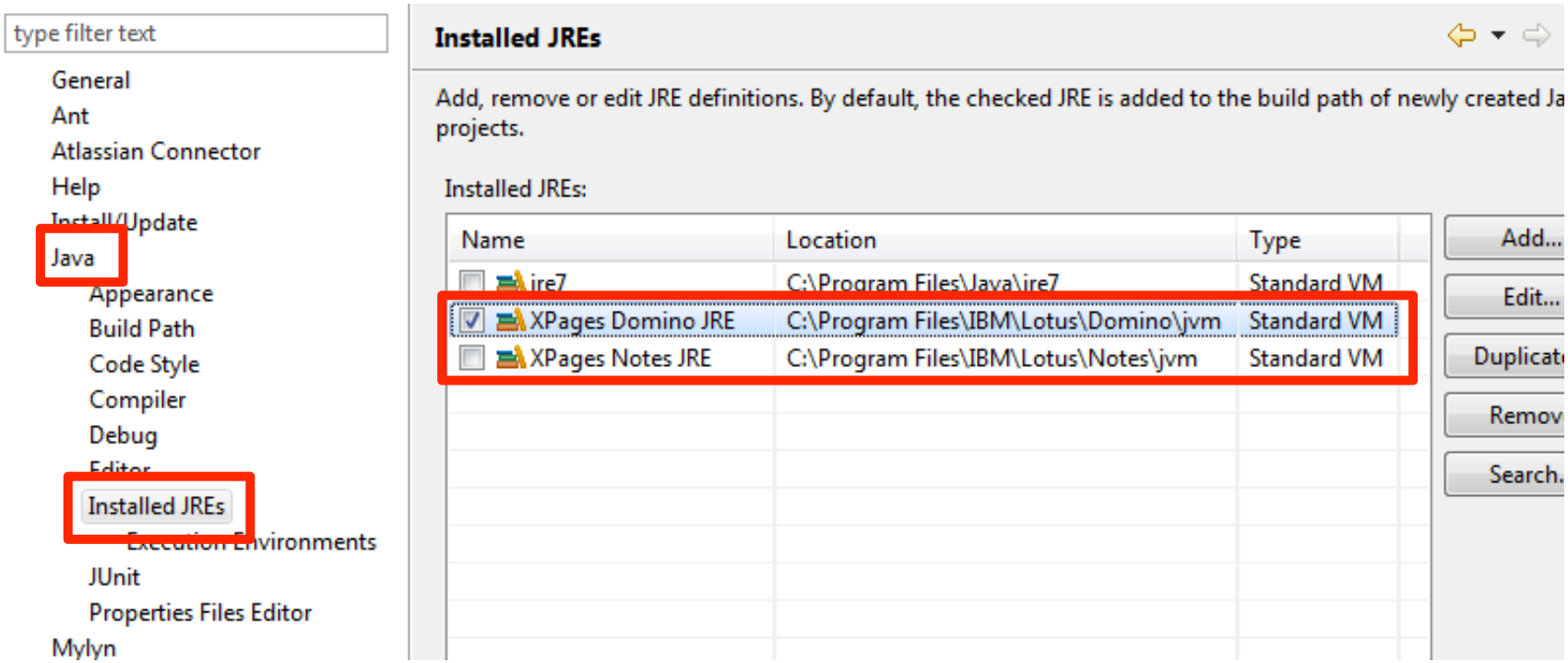




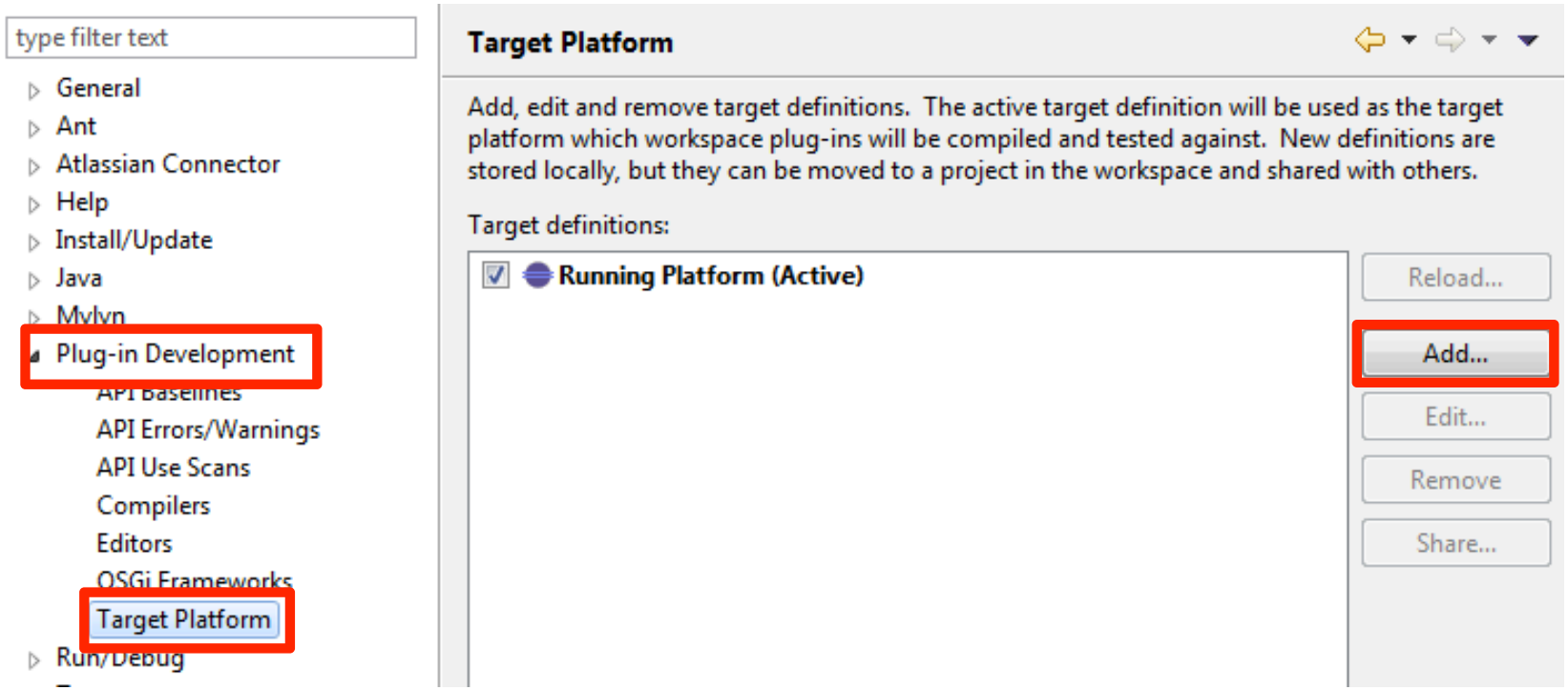
- Konfiguration des XPages SDK (II)



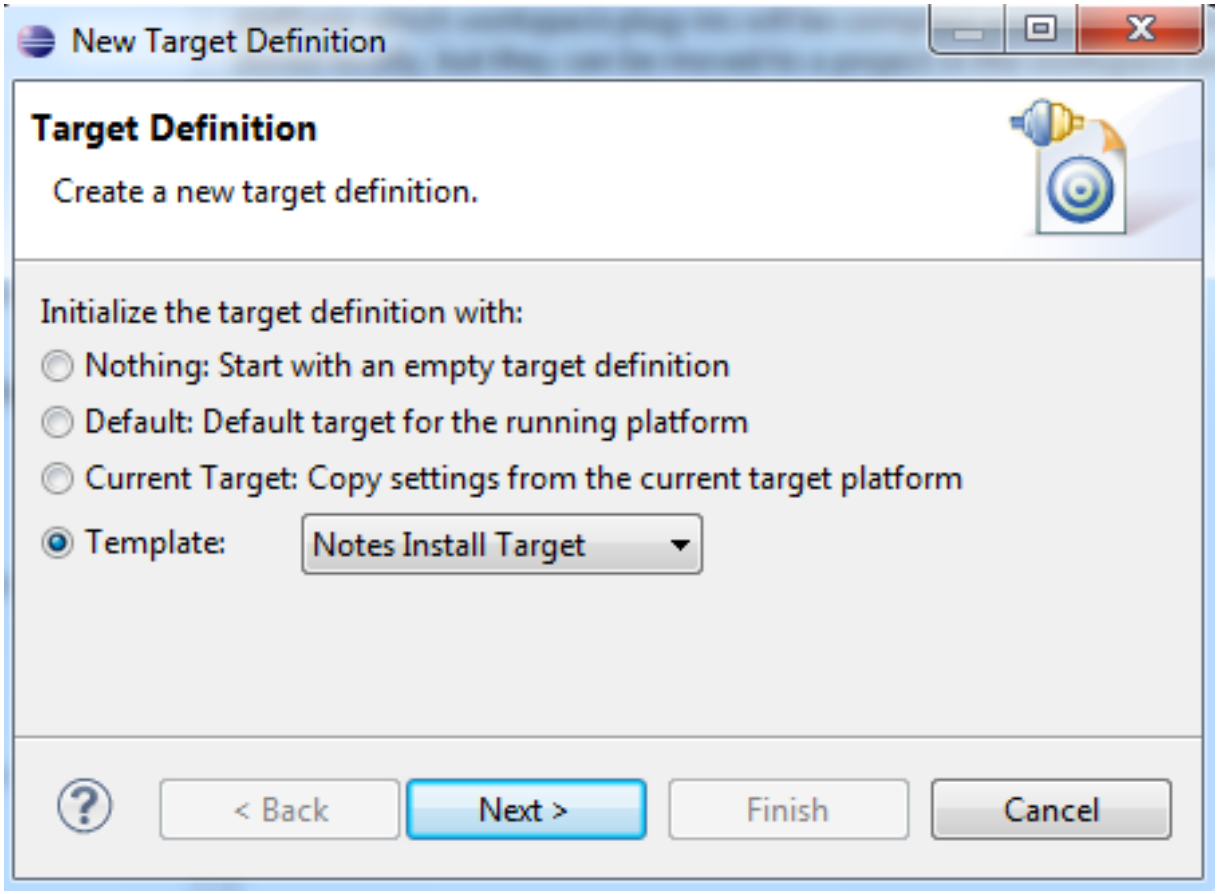
- Konfiguration des XPages SDK (III)



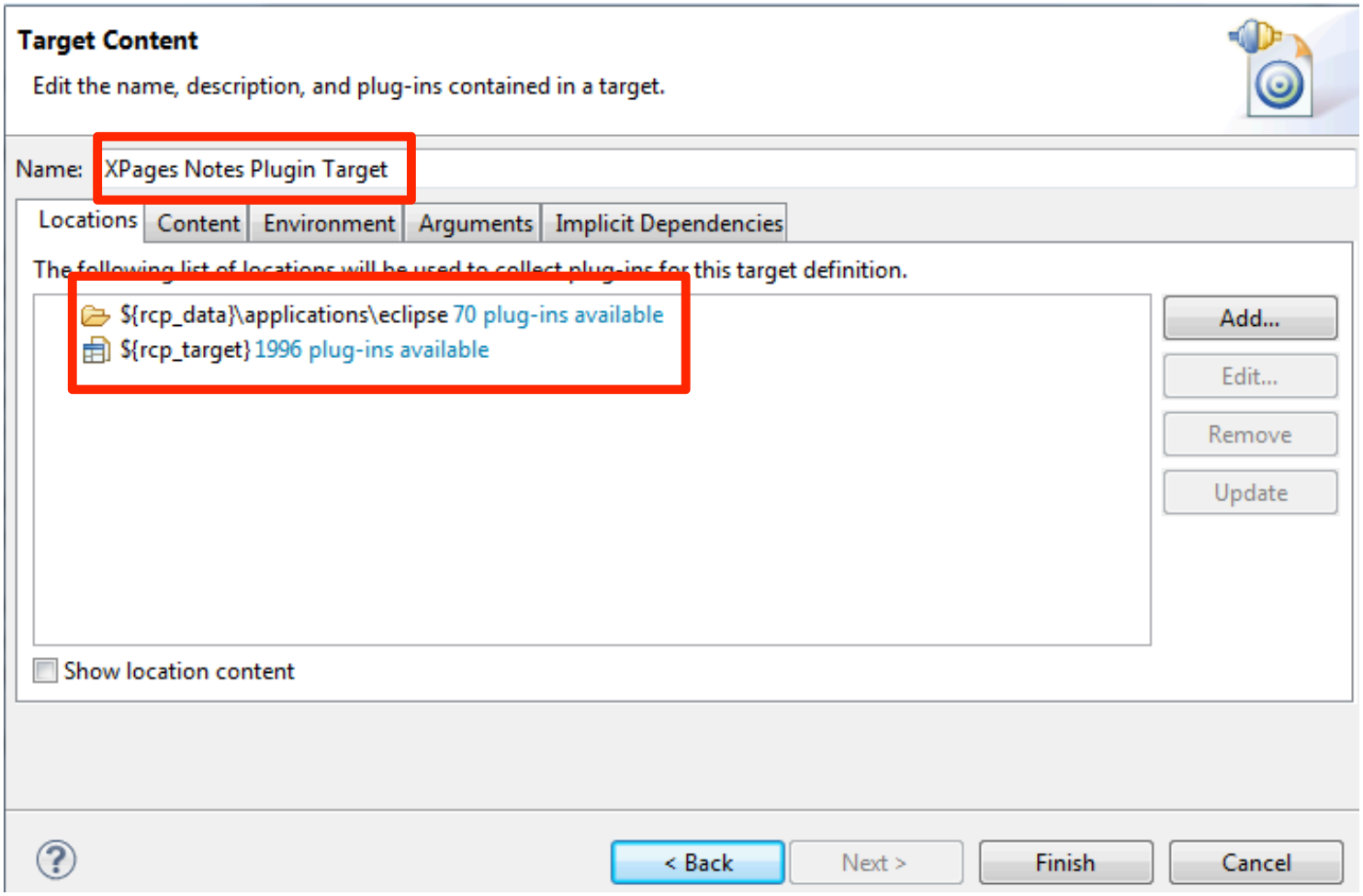
- Konfiguration des XPages SDK (IV)



- Konfiguration des XPages SDK (V)

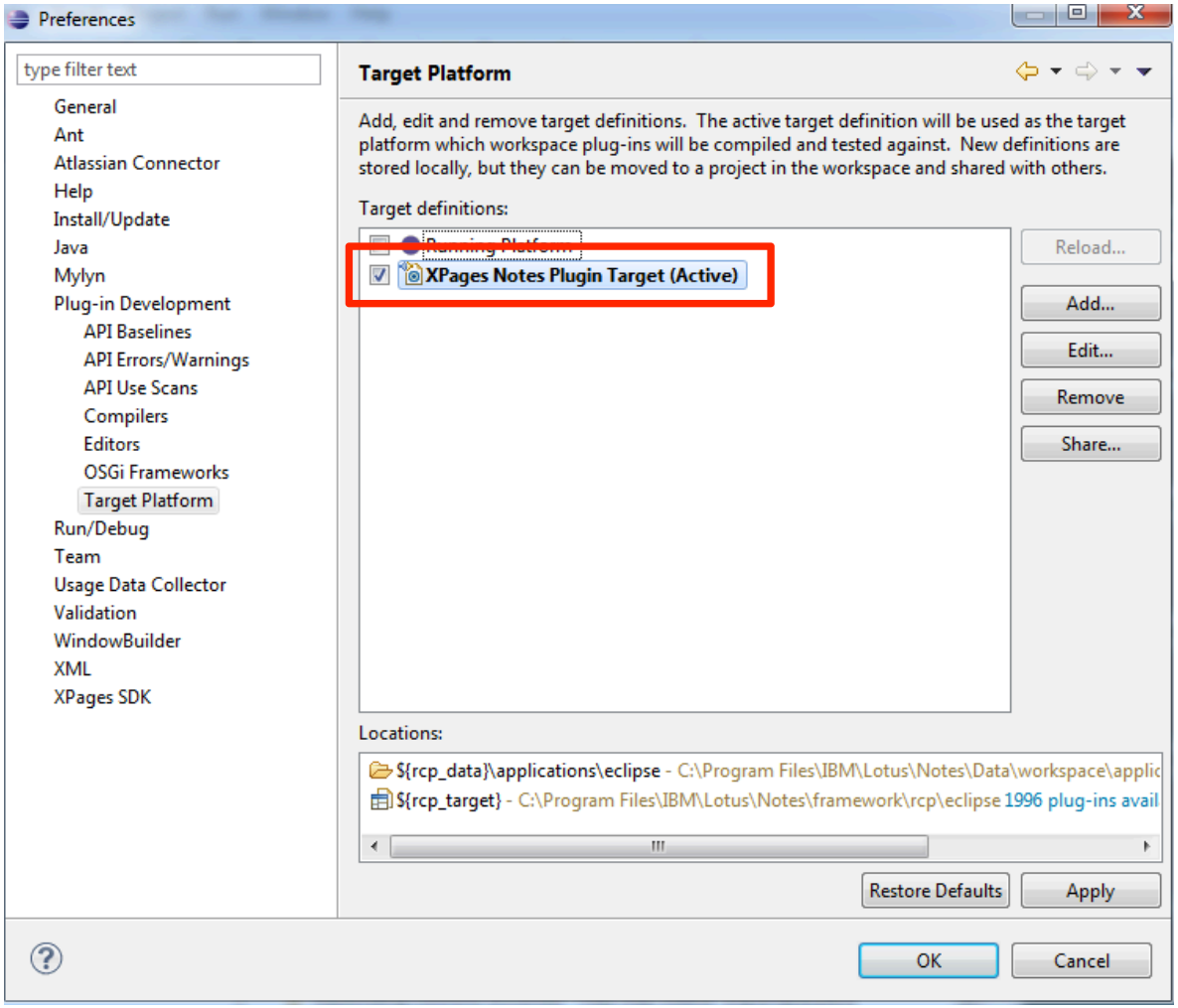


- Konfiguration des XPages SDK (VI)



Setup der Entwicklungsumgebung

- Konfiguration des XPages SDK (VII)

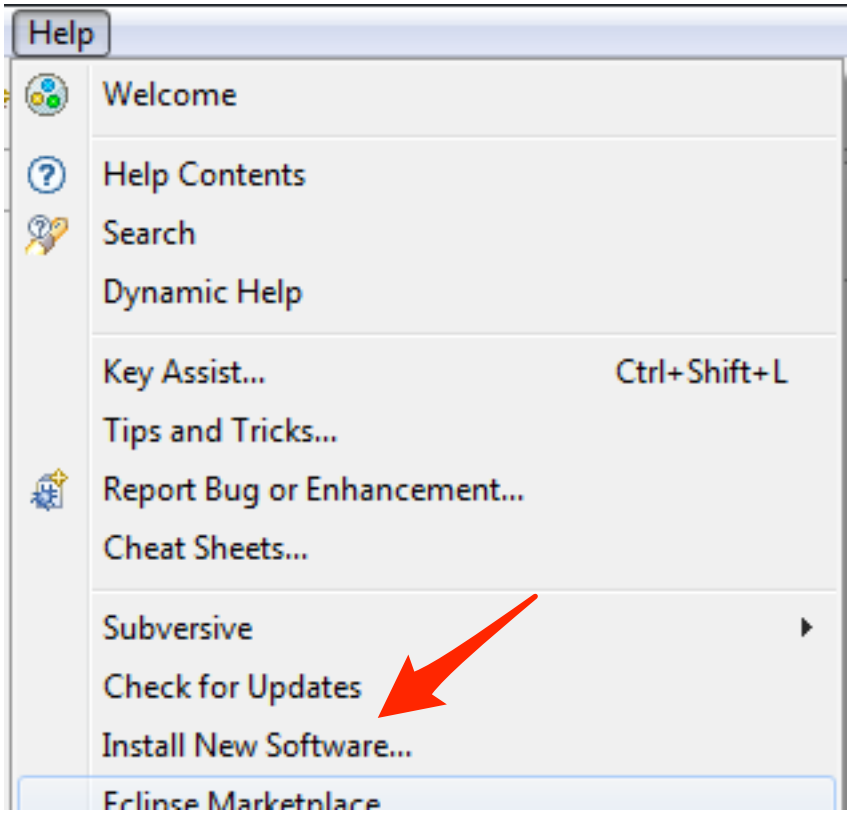


Setup der Entwicklungsumgebung

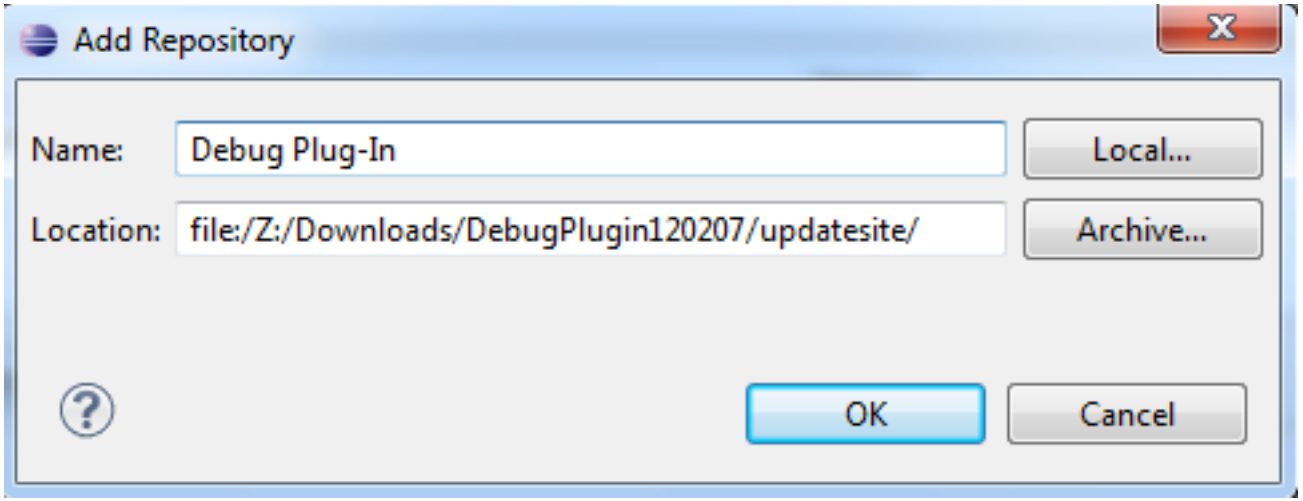
- Ohne Debugging sollte man nicht entwickeln – also ist noch ein Debugger zu installieren.

- Wir setzen hier das Domino Debug Plug-In von OpenNTF ein.
  - http://www.openntf.org/internal/home.nsf/release.xsp?documentId=CBF874E9C4607B4C8625799D00287B8C&action=openDocument

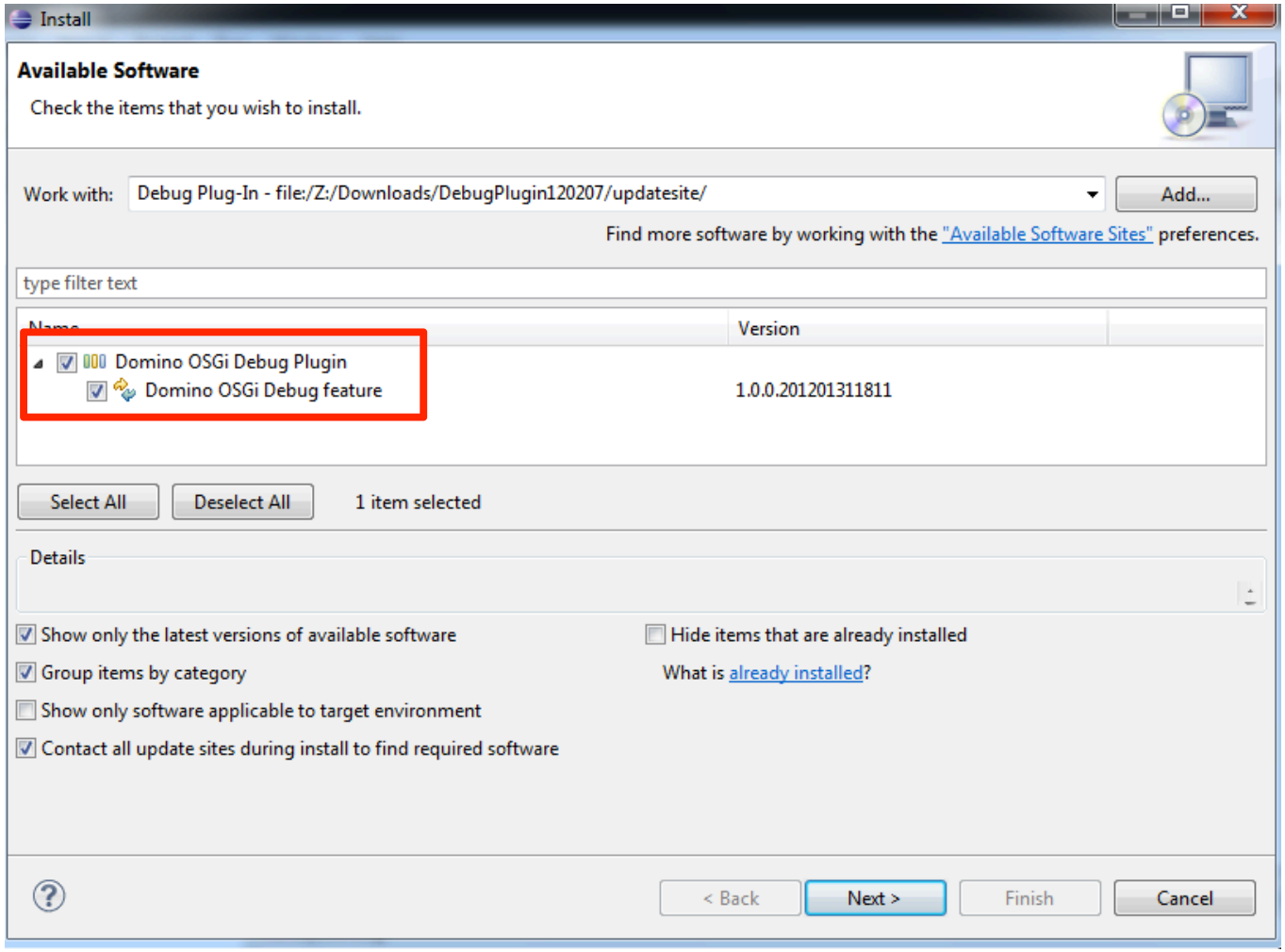
- Installation des Debug-Plug-In (I)



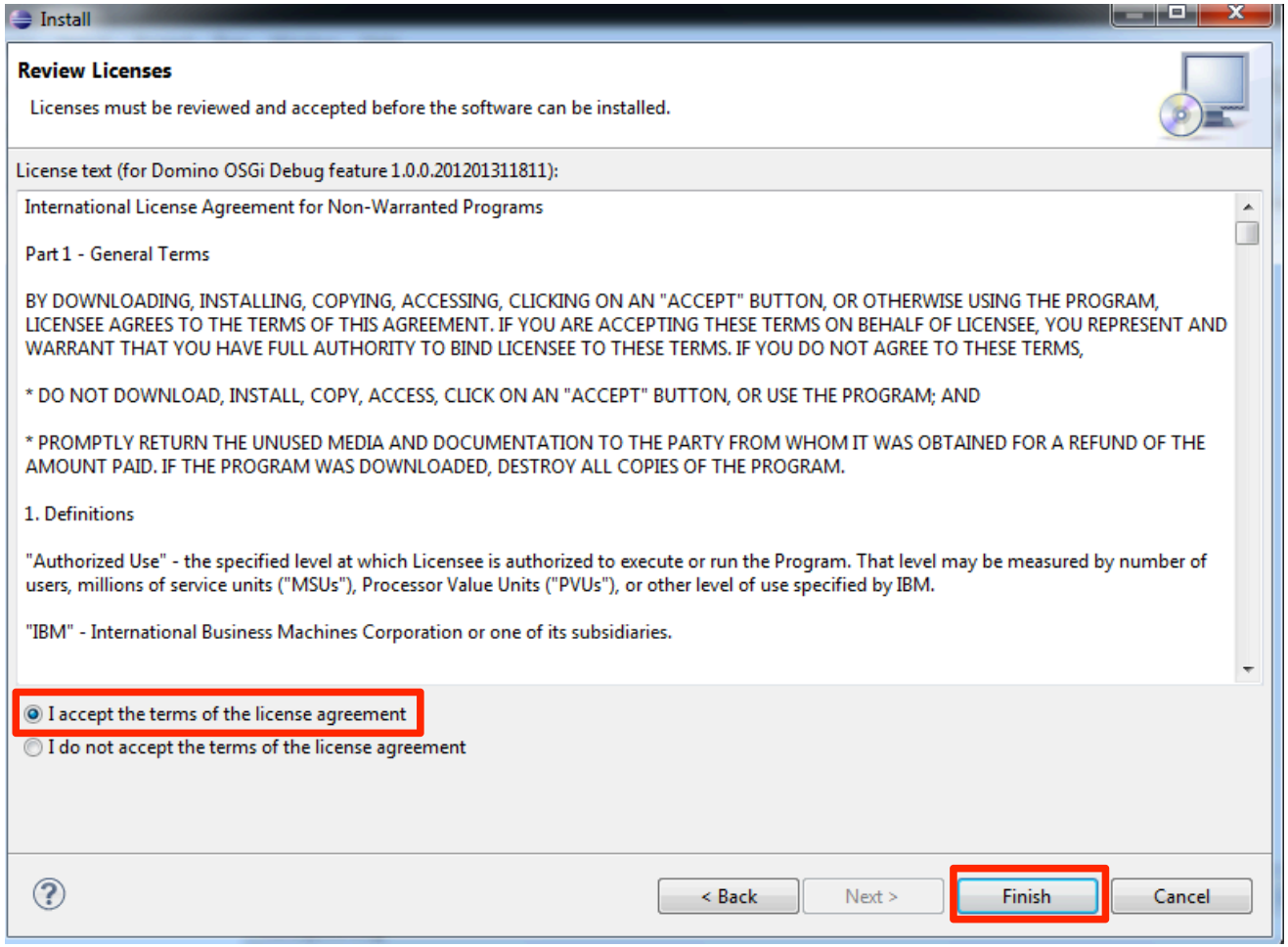
- Installation des Debug-Plug-In (II)



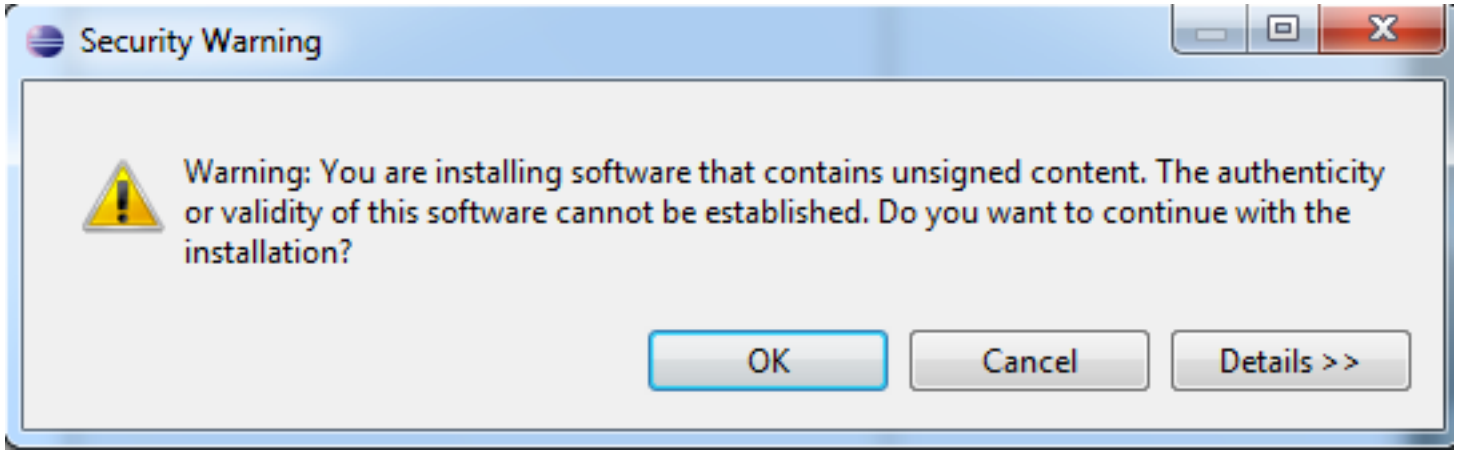
- Installation des Debug-Plug-In (III)



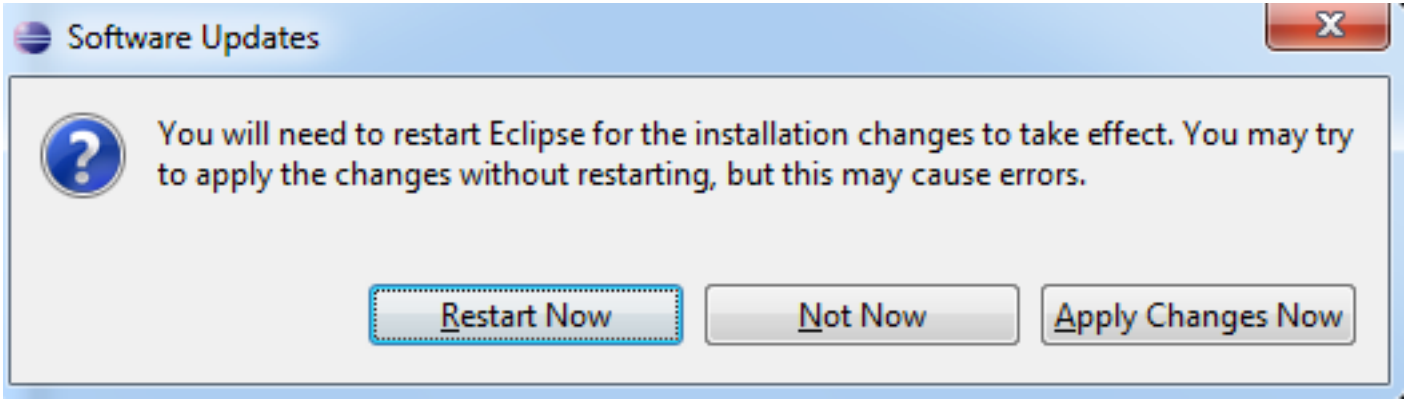
- Installation des Debug-Plug-In (IV)



- Installation des Debug-Plug-In (V)

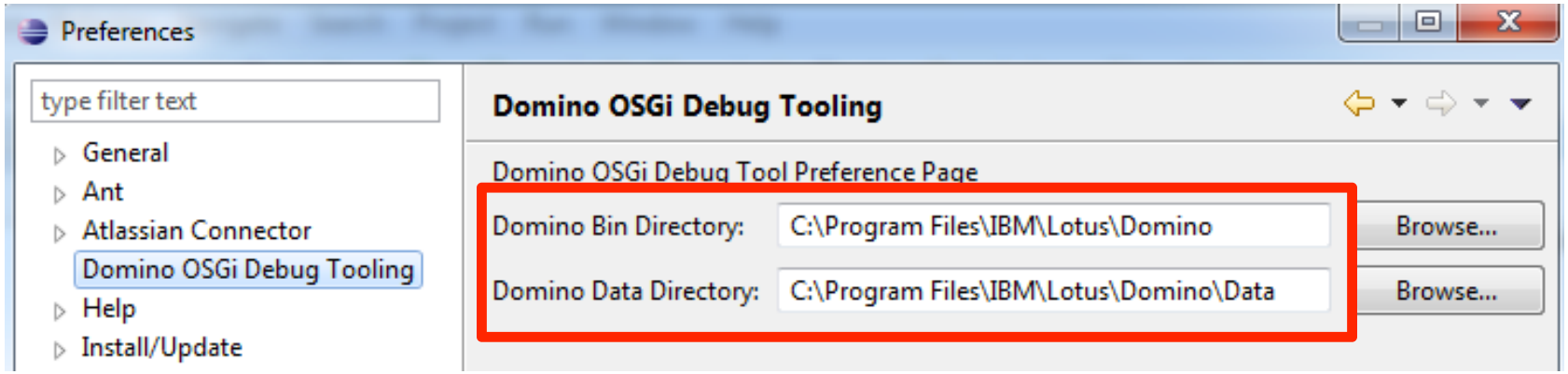


- Installation des Debug-Plug-In (VI)

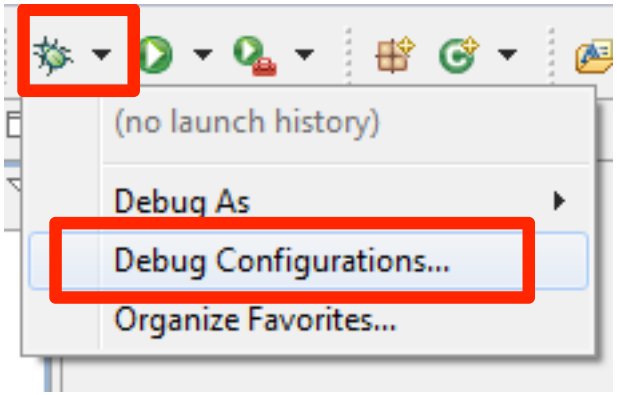


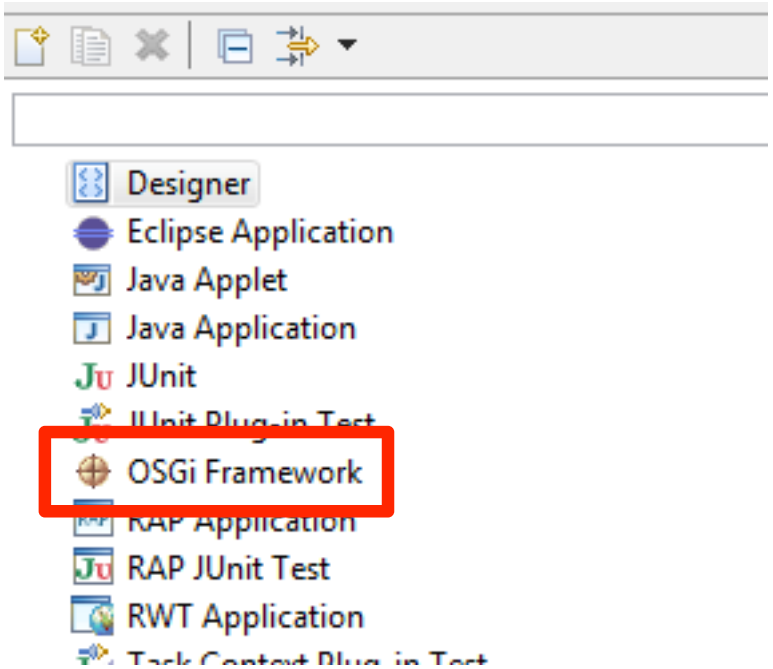
Setup der Entwicklungsumgebung

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



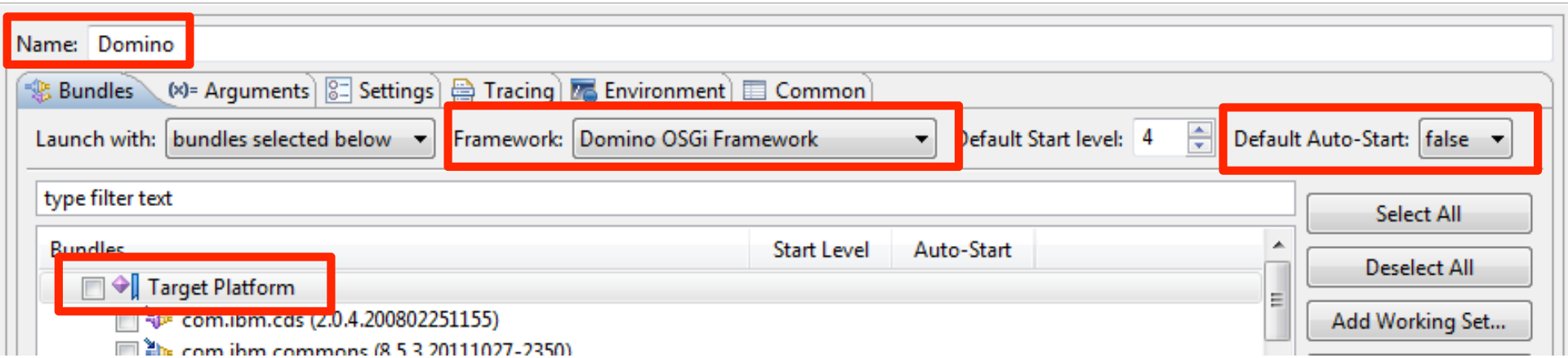
- Erstellung der Debug-Umgebung (I)



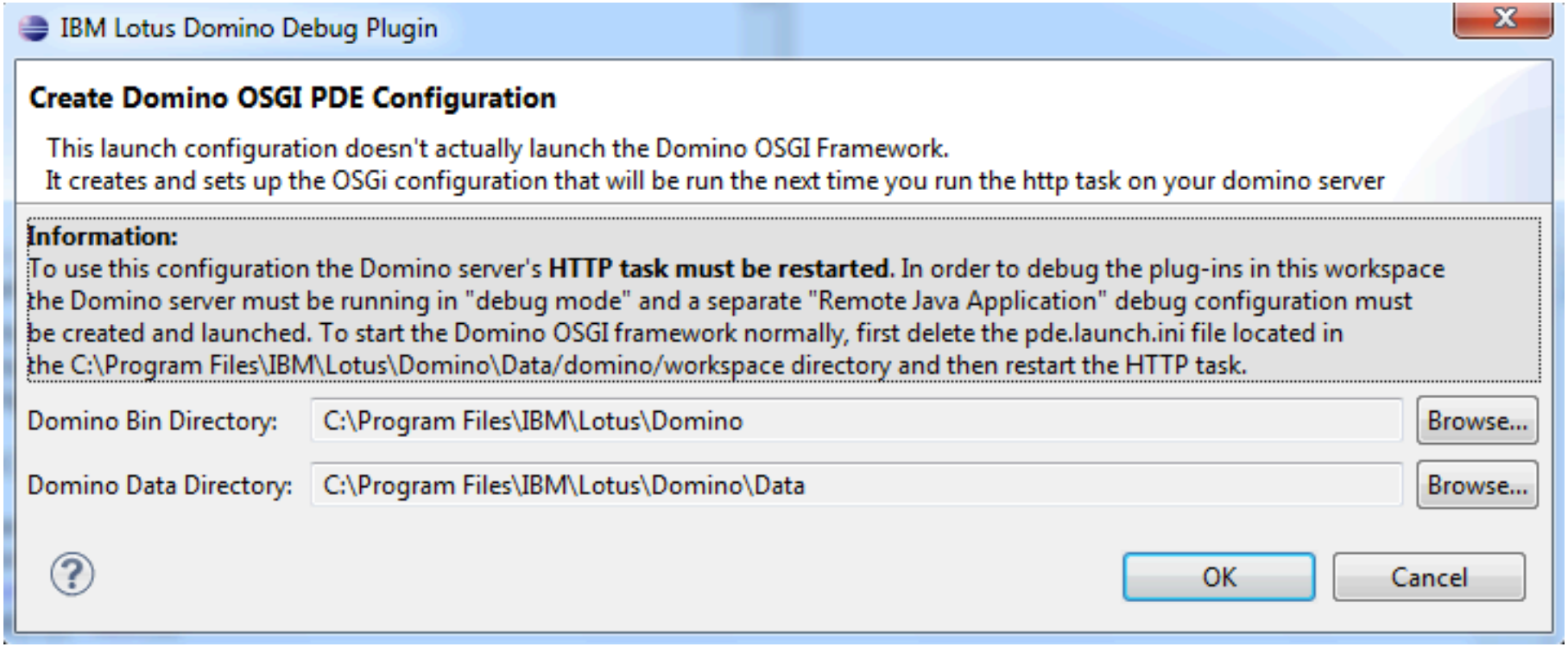


Setup der Entwicklungsumgebung

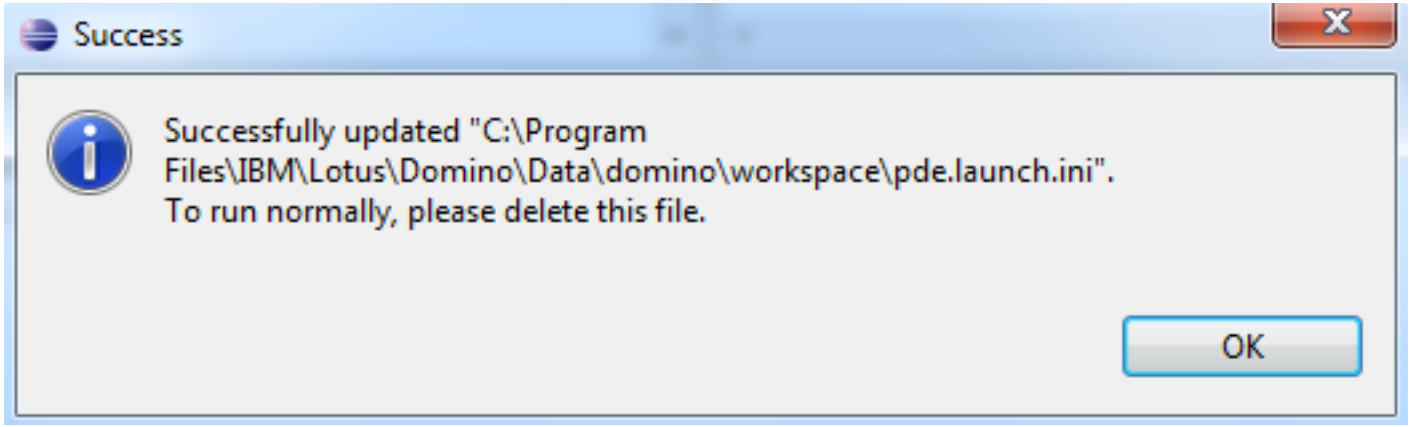
- Erstellung der Debug-Umgebung (II)



- Erstellung der Debug-Umgebung (II)



- Erstellung der Debug-Umgebung (III)



Setup der Entwicklungsumgebung

midpoints
purify it

- 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...

midpoints
purify it

Demo

midpoints
purify it

Worüber wir heute sprechen werden

- Unterschiede XPages, Extension Library und Extensibility API

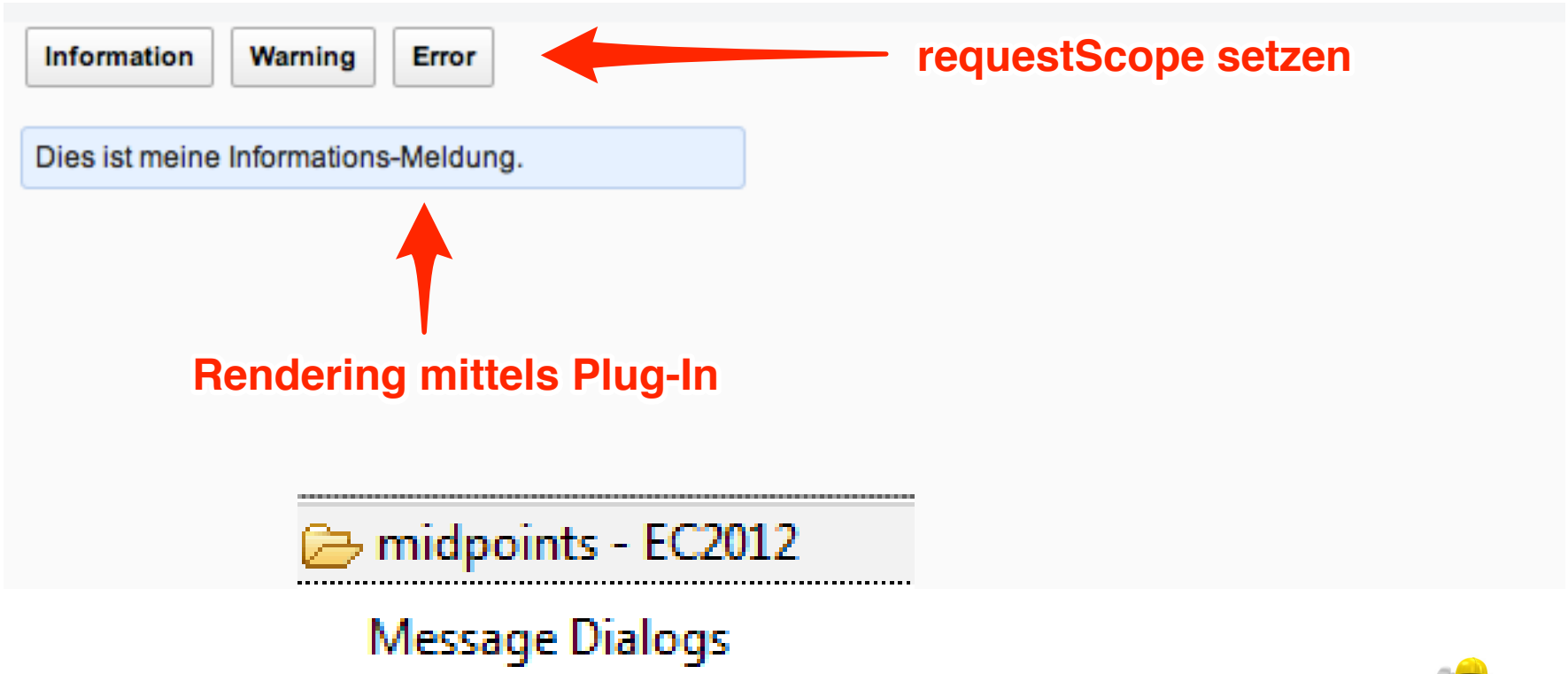
- Vorgehensweise Plug-In-Development

- Codebeispiele

- Deployment

Codebeispiele

- XPages Extensibility API am Beispiel einer einfach UIComponent.

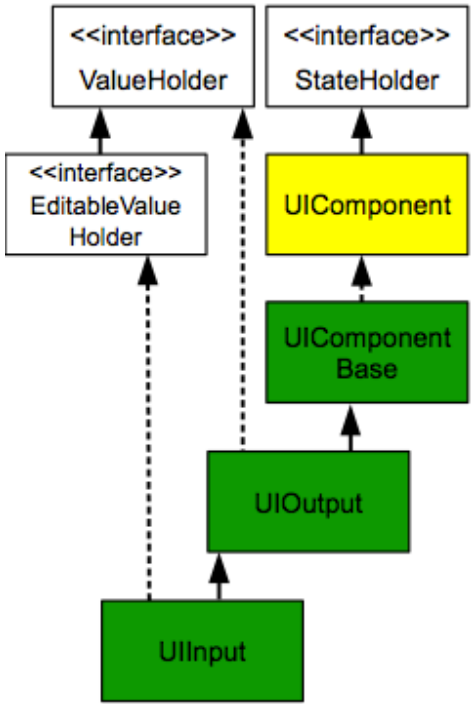


midpoints
purify it

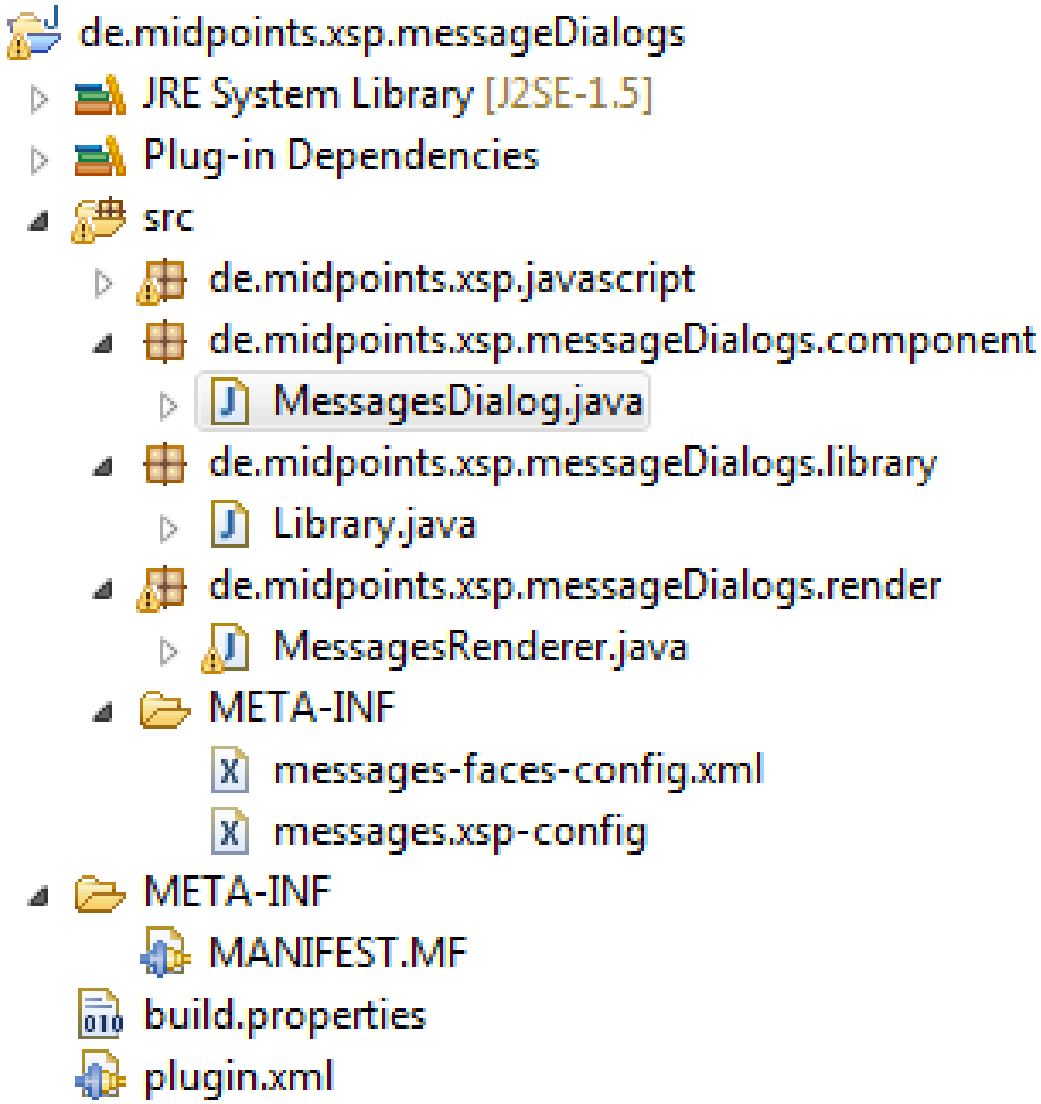
Demo

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



MessagesDialog (UIComponent für den Designer)

```java
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 Identifkation.

MessagesDialog (UIComponent für den Designer)

midpoints
purify it

```
/*
 * 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)

midpoints
purify it

```
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();
    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");
    }
}
```

Helfer-Methode

MessagesRenderer (Output in der XPage)

midpoints
purify it

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

MessagesRenderer (Output in der XPage)

midpoints purify it

```
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;
}
```

Library

midpoints
purify it

```
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",
        };
    }
}
```

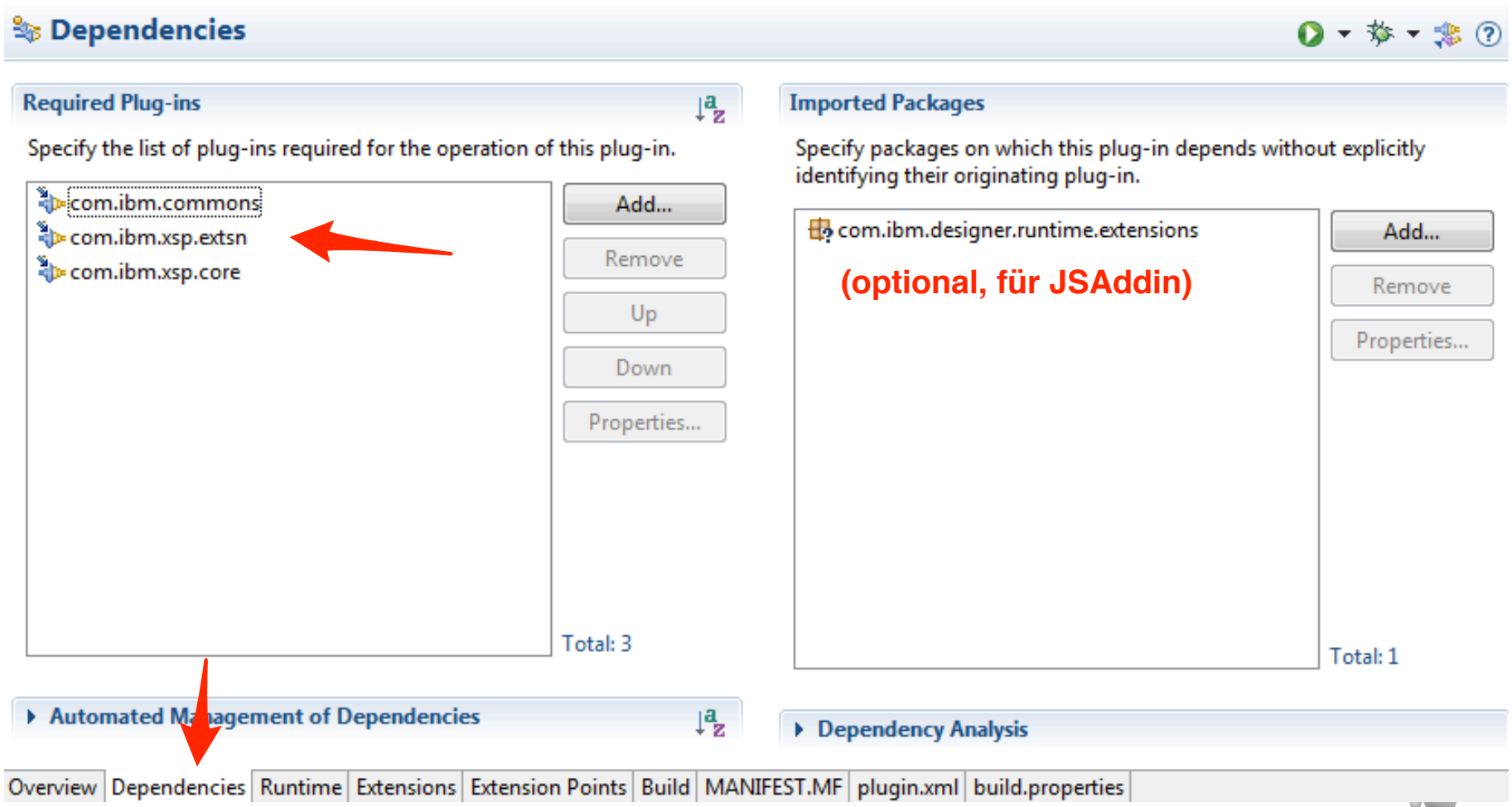
Case-sensitive

Designer (UIComponent)

XPages (Renderer)

manifest.mf

midpoints purify it



plugin.xml

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

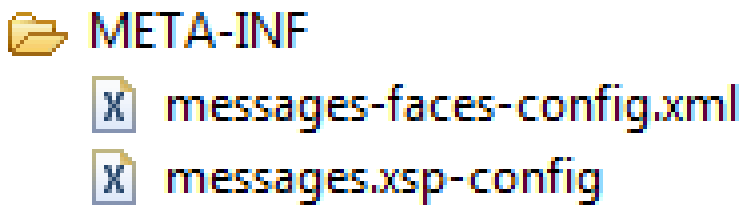
- Bis hierhin ist es noch relativ einfach gewesen...oder?
- Nun müssen die Konfigurationsdateien erstellt und bearbeitet werden. Die verwendeten Bezeichnungen sollten sich am vorhandenen Standard orientieren.

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)

midpoints purify it

```
<?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

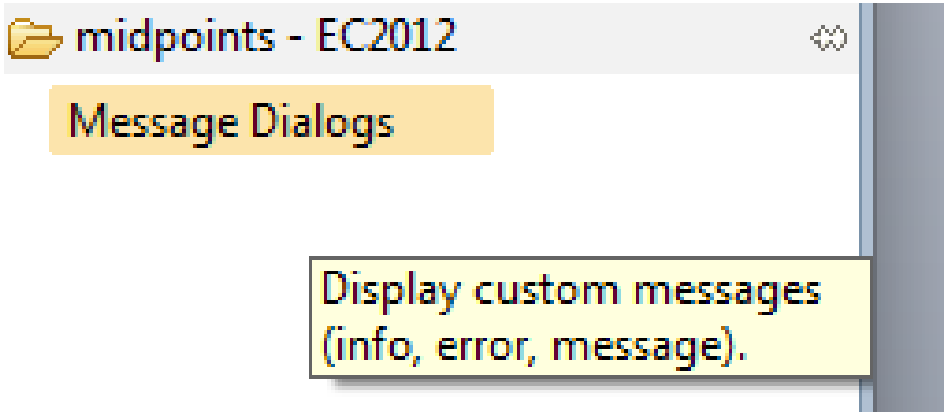
```
public MessagesDialog() {
    super();
    setRendererType("de.midpoints.xsp.MessageDialogs");
}

@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>
```

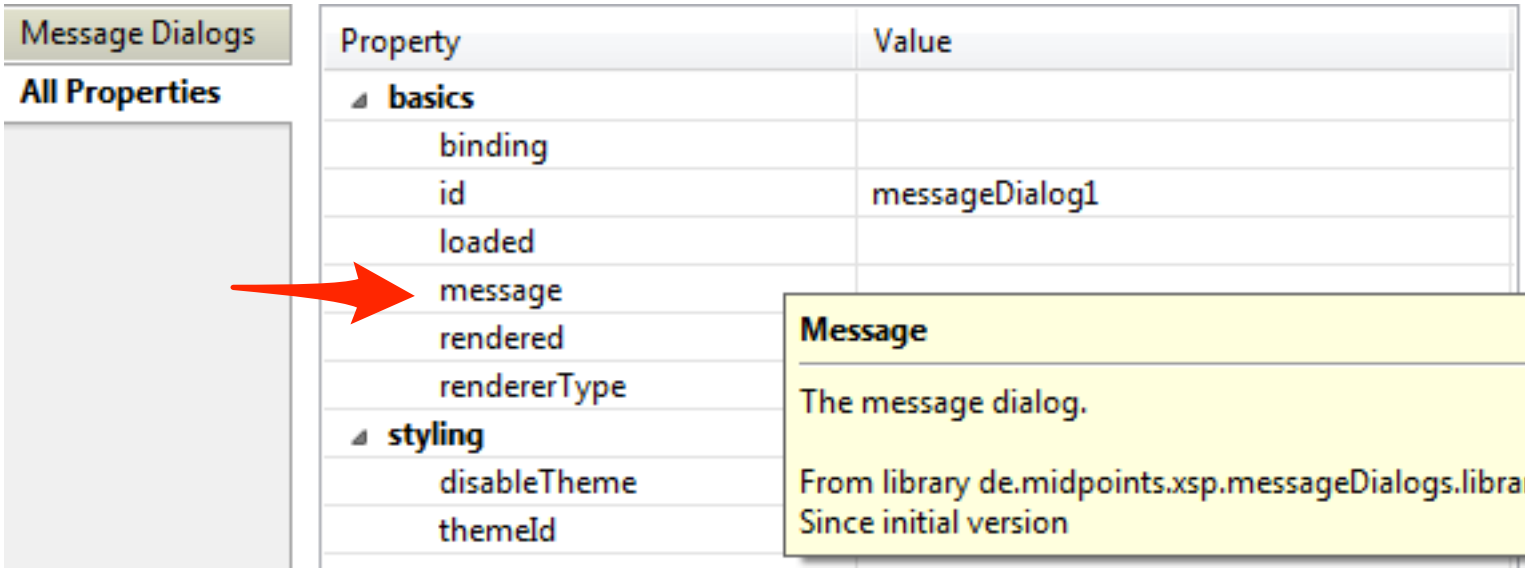
messages.xsp-config

```
<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>
```



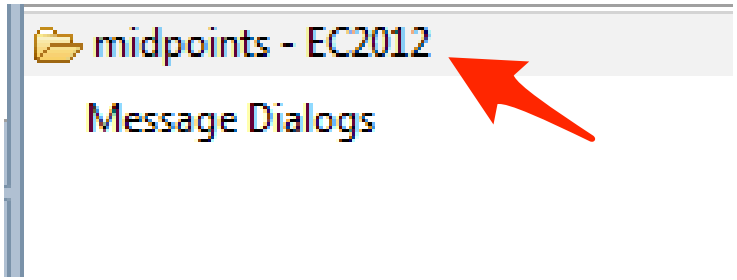
messages.xsp-config

```
<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>
```



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>
```

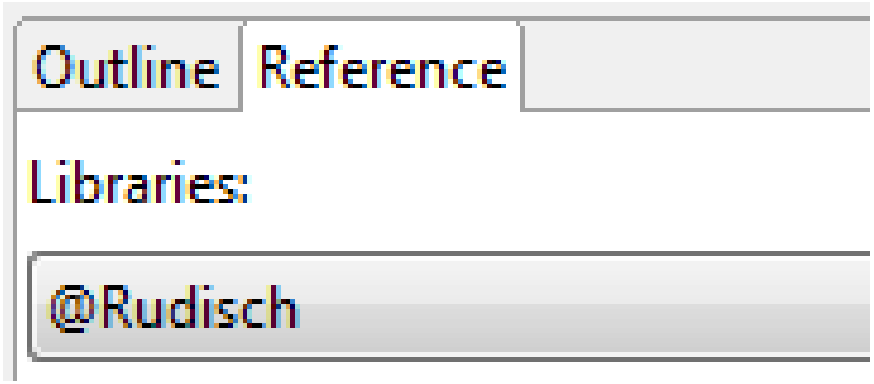


midpoints
purify it

Demo

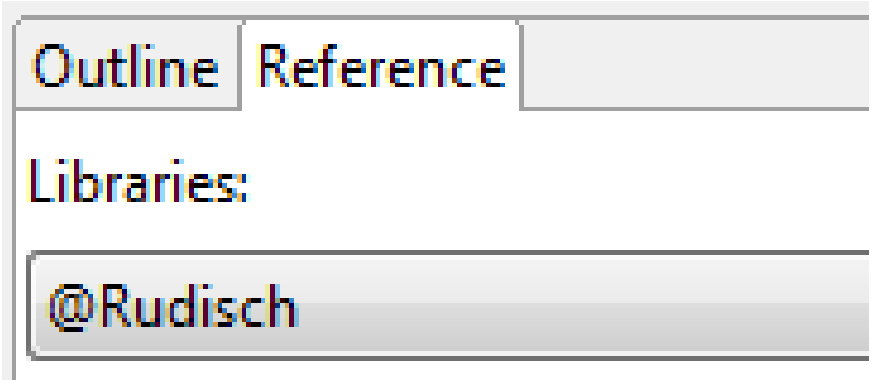
plugin.xml für eigene @Commands

```
<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

```java
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));

    }

}
```



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));
}
```

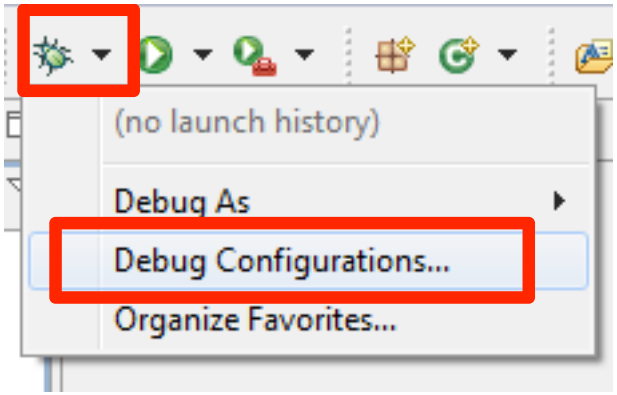
- @TranslateFromRudisch(sentenceToTranslate:string)
- @TranslateToRudisch(sentenceToTranslate:string)

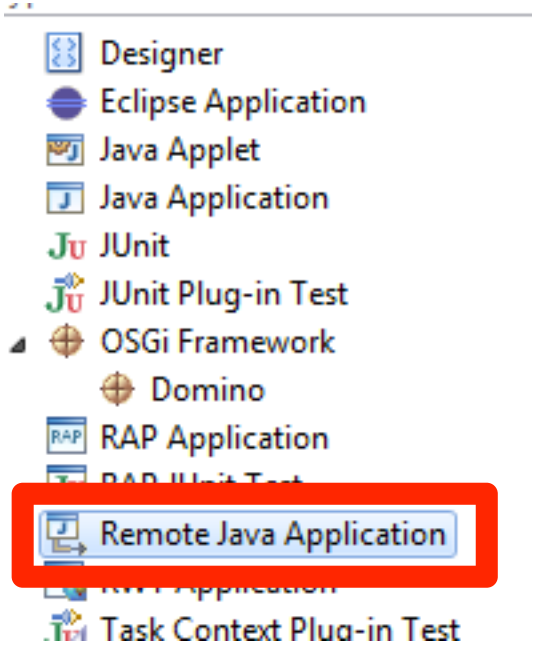
midpoints
purify it

Demo

midpoints
purify it

- Einrichtung einer Remote Java Application zum Debugging.





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

midpoints
purify it

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

midpoints
purify it

Demo

midpoints
purify it

Worüber wir heute sprechen werden

- Unterschiede XPages, Extension Library und Extensibility API

- Vorgehensweise Plug-In-Development

- Codebeispiele

- Deployment

Deployment Notes-Client

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

Deployment Domino-Server

midpoints
purify it

- 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.

midpoints
purify it

Demo

Vielen Dank für Ihre Aufmerksamkeit!

midpoints
purify it

René Winkelmeyer
Skype/Twitter/LinkedIn/Facebook: muenzpraeger
http://blog.winkelmeyer.com
http://www.xing.de/Rene_Winkelmeyer
rene.winkelmeyer@midpoints.de / mail@winkelmeyer.com

midpoints | purify it
http://www.midpoints.de
info@midpoints.de