Entwicklercamp 2011

Lotus knows.

Smarter software for a Smarter Planet.

Track 4 Session 8

Add-ons für Client/Designer auf Basis neuer Java APIs von Lotus Notes 8.5.1 und höher!

Karsten Lehmann | Geschäftsführer | Mindoo GmbH

NOTES & DOMINO ENTWICKLERCAMP

Agenda

- Einführung
- Lotus Notes mit Java erweitern
- Domino Designer mit Java erweitern
- Mehrwert für das klassische Lotus Notes und XPages
- Best Practices
- Q&A

Über uns

- Mindoo ist IBM Business Partner und Notes/Domino Design Partner
- Wir konzentrieren uns auf die “neuen” Aspekte der IBM Lotus Notes-Entwicklung
  - Eclipse/Expeditor-Plugins und Rich-Client-Anwendungen
  - XPages-Anwendungen und -Controls
  - Composite Application Architekturen
  - LiveText Erweiterungen
- Karsten Lehmann und Tammo Riedinger
  - Gründer der Mindoo GmbH
  - Seit 2004 Entwickler der Applikation MindPlan® für die Haus Weilgut GmbH, Mindmapping und Projekt-Management für Lotus Notes, IBM Award Winner 2008
- Weitere Informationen:
  http://www.mindoo.de

[mindu:]
Mindoo GmbH

Motivation

- „Notes kann auch das“ - ja, aber wieso so kompliziert?
  - Eclipse- und XPages-Entwickler mussten bisher zu viele Workarounds nutzen, um mit der klassischen UI interagieren zu können
  - Man denke nur an „Nutzung der Notes.ini zum Datenaustausch“ oder „Seiten öffnen, die sich schließen, nur um LotusScript-Code im QueryClose-Event auszuführen“
  - Es fehlte eine Java-API zur Ansteuerung der Notes-Oberfläche.
  - Der Sinn der neuen APIs ist, diese Lücke zu füllen und den Entwicklern damit das Leben zu erleichtern.
- Wir waren an den Diskussionen über die Features der neuen „Notes and Designer extensibility APIs“ beteiligt
  - Diskussionen mit IBM dev über API-Entwürfe bereits auf der Lotusphere 2009 und später in Konferenzschaltungen
  - Feedback zum Design / Berichte über Tests im Zuge des Design Partner Programms
- Wie immer bei ersten Releases gibt es noch Raum für Verbesserungen
  - Wir denken jedoch, dass diese APIs schon ein **riesiger Schritt** vorwärts sind

Achtung

- Wir können hier nicht alle APIs im Detail betrachten!
  - Wir werden uns stattdessen auf die wichtigsten Neuerungen konzentrieren: die neue Java UI API und die Designer Java-API
- Wir wollen Ihnen einen guten Eindruck vermitteln, was sie wirklich mit den Notes 8.5.1/8.5.2 APIs tun können
  - Hoffentlich verlassen Sie diesen Vortrag bereits mit einigen eigenen Ideen
  - Wir haben ca. 10 Demos für Sie erstellt!
- Wir versuchen Sie heute nicht mit Code zu Tode zu langweilen!
- Verfolgen Sie stattdessen die kommende Serie von Artikeln mit Details zu den Demos in unserem Blog:
- http://blog.mindoo.com

Startvorbereitungen

- Die APIs und Demos basieren auf
  - Eclipse 3.4.2
  - Lotus Expeditor® Toolkit 6.2.2
  - IBM Lotus Notes 8.5.2
- Installieren Sie Expeditor in Eclipse und setzen Sie Lotus Notes 8.5.2 als Ziel-Plattform

- Erstellen Sie ein Plugin-Projekt für Ihren eigenen Code
- Für die UI API müssen Sie die folgende Abhängigkeit definieren:
  - com.ibm.notes.java.ui
- Für die DDE API müssen Sie die folgenden Abhängigkeiten definieren:
  - com.ibm.designer.domino.ui.commons
  - com.ibm.designer.domino.ide.resources

Agenda

- Einführung
- Lotus Notes mit Java erweitern
- Domino Designer mit Java erweitern
- Mehrwert für das klassische Lotus Notes und XPages
- Best Practices
- Q&A

Java UI API – eine grobe Übersicht

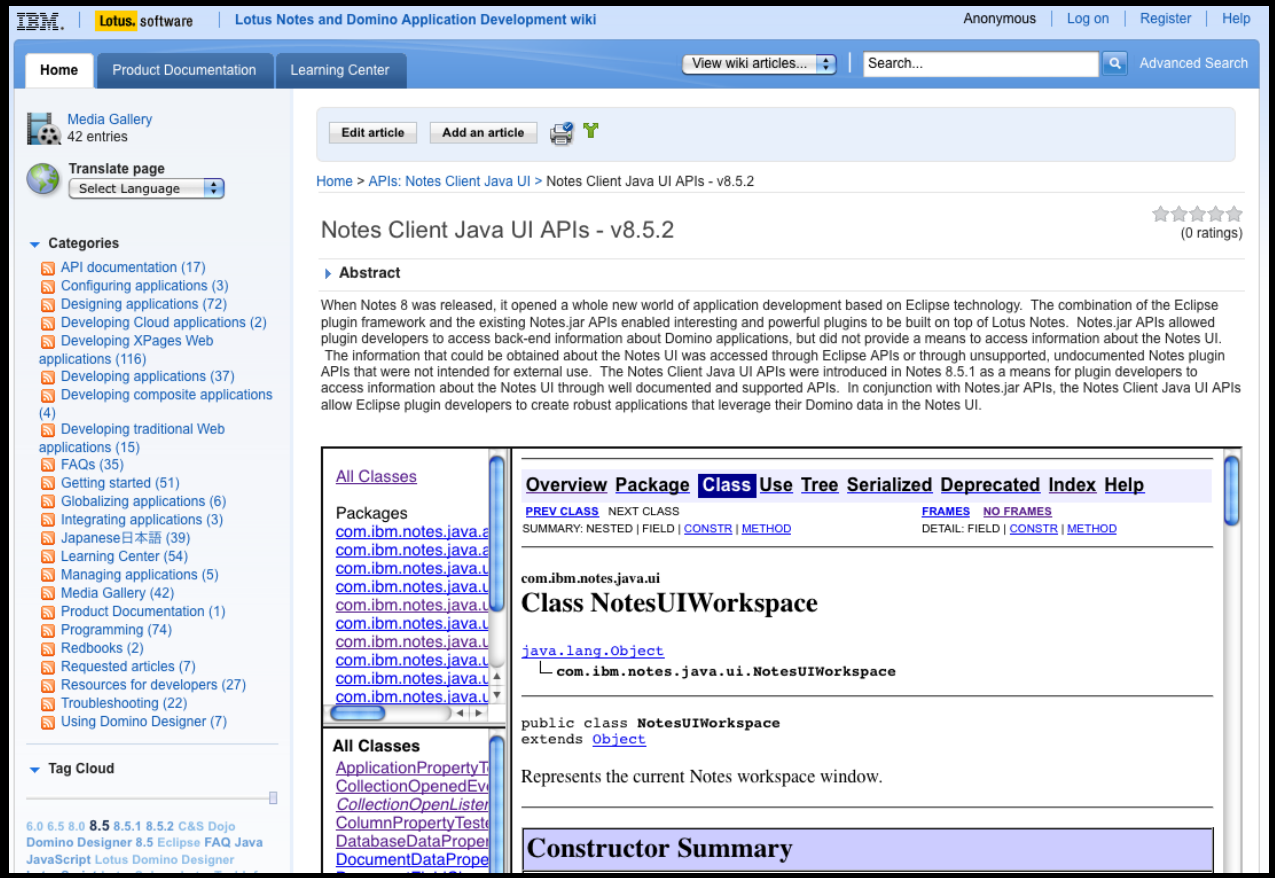
- API ist neu in 8.5.1
  - Wenige Features wurden in 8.5.2 ergänzt
- Verbessert die Integration zwischen Eclipse und dem klassischen Lotus Notes Client
  - Eclipse-Plugins können endlich Informationen über den Zustand von Formularen und Ansichten des klassischen Clients bekommen
  - Neue Wege der Interaktion dieser beiden Welten
- Bestehende Funktionalität lässt sich nun einfacher nutzen
  - Öffnen von Design-Elementen
  - Drucken von Dokumenten und Ansichten aus Eclipse heraus
  - Erstellen neuer Dokumente und Vorbelegen von Werten
  - Temporäre Dokumente, um Daten zu speichern und zu übertragen
  - Ausführen von Notes-Code in einem Hintergrund-Thread inklusive Speicher-Management (NotesSessionJob)
  - Auslesen der gewählten Dokumente und Spalteninformationen in einer Ansicht (8.5.2)

Java UI API – eine grobe Übersicht

- Ergänzt neue Funktionalitäten und verkleinert die Lücke zwischen Eclipse und dem klassischen Lotus Notes Client
  - Lesen/Modifizieren von Dokumenten-Inhalt im Bearbeiten-Modus
  - Document Listener (Bearbeiten-Modus an/aus, Änderungen und Schließen von Dokumenten verfolgen)
  - Eclipse-Selektion für fokussierte Felder und neue noch nicht gespeicherte Dokumente
  - Datenbank zu Workspace hinzufügen
  - Eingabe-Methoden von NotesUIWorkspace, z.B. um Datenbanken auszuwählen
  - Viele Eclipse Property Tester (eine Art „hide when“ für Eclipse-Elemente wie z.B. Actions)
  - LotusScript-Agenten in UI-Thread ausführen (es können Dialoge angezeigt werden in den Agenten) inkl. Datenaustausch und Benachrichtigung per Callback
- Was nützt die API reinen XPages-Entwicklern?
  - UI API eignet sich ideal zur Integration bestehender Notes-Anwendungen in XPages
  - Wir demonstrieren Ihnen später in der Session, wie Sie die UI API von XPages-Anwendungen aufrufen können!

Wo ist die Dokumentation?

- Die aktuellste Version der UI API-Doku befindet sich im AppDev-Wiki
  - http://www-10.lotus.com/ldd/ddwiki.nsf/dx/Notes_Client_Java_UI_APIs-v8.5.2
- DDE API Javadocs sind verfügbar in der Notes Client Hilfe



Klein anfangen – Ihr erstes Beispiel

- Toolbar-Aktion zum Ändern eines Feldes in einem geöffneten Formular
- Nutzen Sie diesen Code in einer Standard Eclipse Toolbar-Aktion:

```
NotesUIWorkspace ws = new NotesUIWorkspace();
NotesUIDocument uidoc = ws.getCurrentDocument();
if (uidoc != null) {
   NotesUIField field = uidoc.getField("Subject");
   if (field != null)
      field.setText("Hello world!");
}
```


Klein anfangen – Ihr erstes Beispiel

- Für Felder in Backend-Dokumenten:

```
NotesUIWorkspace ws = new NotesUIWorkspace();
NotesUIDocument uidoc = ws.getCurrentDocument();
if (uidoc != null) {
   NotesBEDocument beDoc = uidoc.getBEDocument();
   String oldValue = beDoc.getItemString("Flag");
   // do something here
   beDoc.setItemValue("Flag", "1");
   //optional to see your changes in the UI:
   uidoc.reload();
}
```

- Kein voller Zugriff auf das Dokument durch technische Restriktionen

Demo

- Erweiterung des Open-Menüs des Notes Clients
- Mit Eclipse-Aktionen auf ein UI-Dokument zugreifen und dieses modifizieren

Tiefer einsteigen - NotesUIWorkspace erkunden

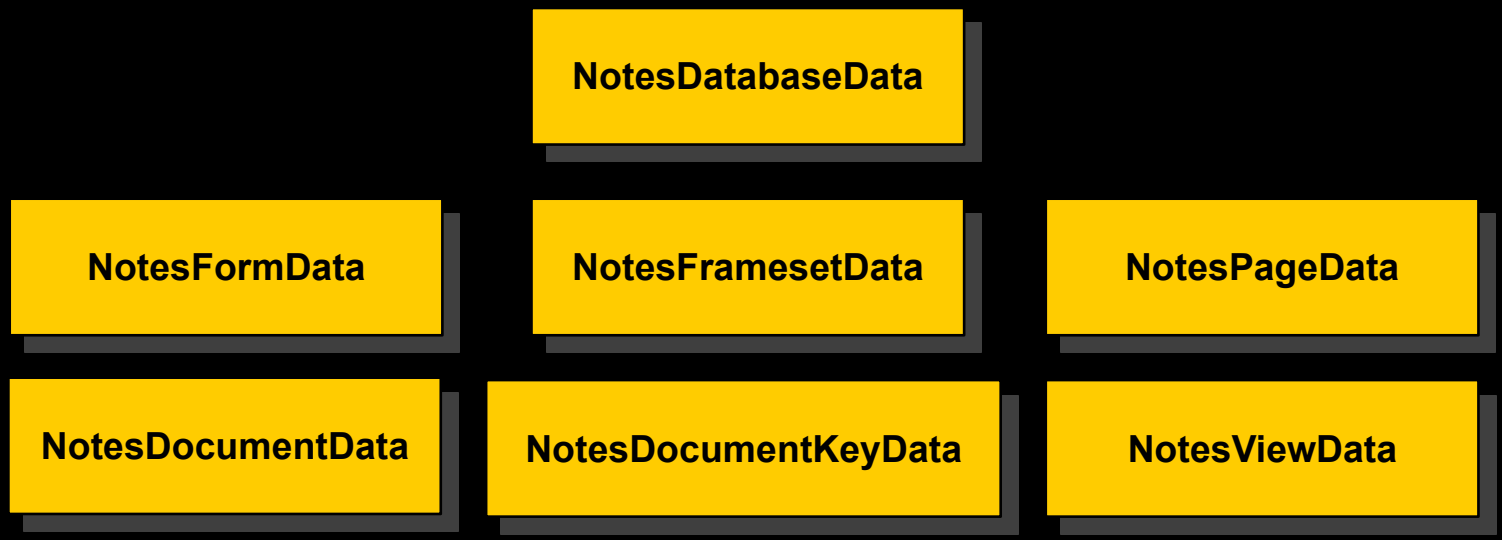
- Ein neues Dokument mit vorbesetzten Feldern erstellen

```
NotesUIWorkspace ws = new NotesUIWorkspace();
NotesDatabaseData dbData =
   new NotesDatabaseData("Server/Org", "main/jdoe.nsf");
NotesFormData formData =
   new NotesFormData(dbData, "Memo");
formData.addComposeItem("SendTo", "Peter Smith/Org");

ws.composeDocument(formData);
```


Tiefer einsteigen - NotesUIWorkspace erkunden

- Daten-Klassen werden in der API genutzt, um Daten zwischen Notes-Sessions zu halten
- Daten können sicher zwischen API-Aufrufen übergeben und lokal gesichert werden
- In einigen Fällen fehlen einige Daten-Werte (z.B. der Datenbankpfad)
  - Ist eine technische Beschränkung der API
  - Es können dann „open“-Methoden genutzt werden, um diese Daten zu laden
  - z.B. NotesDatabaseData.open(Session)



Tiefer einsteigen - NotesUIWorkspace erkunden

- Neue Dokumente auf Basis bestehender erstellen (1/2)

```
NotesUIWorkspace ws = new NotesUIWorkspace();
session.setConvertMIME(false); // Do not convert MIME to RT

Document tmpDoc = ws.getTemporaryDocument(session);
setBasicMailFields(tmpDoc);    // e.g. set subject, receipient

//Create the body as a MIME entity
MIMEEntity body = tmpDoc.createMIMEEntity("Body");
Stream stream = session.createStream();
stream.writeText("<ul><li>hello</li><li>world</li></ul>");
body.setContentFromText(stream, "text/html;charset=UTF-8",
   MIMEEntity.ENC_IDENTITY_7BIT);

//Save the document
tmpDoc.save(true, true);
```


Tiefer einsteigen - NotesUIWorkspace erkunden

- Neue Dokumente auf Basis bestehender erstellen (1/2)

```
//now compose a new document in the mail database
//based on the temporary document

Database mailDb = openMailDatabase();
NotesDatabaseData mailDbData = new NotesDatabaseData(mailDb);
ws.composeDocument(mailDbData, tmpDoc);

//don't forget to restore conversion
session.setConvertMime(true);
```


Tiefer einsteigen - NotesUIView erkunden

- Neu in 8.5.2: Einheitlicher Zugriff auf die Ansichtsselektion
- Funktioniert in Java Views und klassischen Ansichten

```
NotesUIWorkspace ws = new NotesUIWorkspace();
NotesUIElement uiElement = ws.getCurrentElement();

if (uiElement instanceof NotesUIView) {
  NotesUIView uiview = (NotesUIView) uiElement;
  NotesUIViewEntryCollection entryCol = uiview.getActionableEntries();

  //things we can get without opening the entry collection
  NotesUIViewEntry firstEntry = entryCol.getFirstEntry();
  int nrOfEntries = entryCol.size();

  //to access more than the first entry, we need to open the collection
}
```


Tiefer einsteigen - NotesUIView erkunden

- Neu in 8.5.2: Einheitlicher Zugriff auf die Ansichtsselektion

```
//to access all entries, we need to open the collection
entryCol.open( new CollectionOpenListener() {
 public void collectionOpened( CollectionOpenedEvent evt ) {

  If ( evt.getError() == null ) {
   NotesUIViewEntryCollection loadedCol = evt.getCollection();
   Iterator<NotesUIViewEntry> iterator = loadedCol.iterator();

   //iterate over entries here:
   //NotesUIDocumentEntry, NotesUITotalEntry, NotesUICategoryEntry
  }
 }
}, false);
```


Demo

- Flexibler Report-Generator
  - Ansichtsselektion verarbeiten
  - Report erzeugen durch Ausführen von Formelcode oder JavaScript auf der Selektion
  - Neues Dokument erzeugen mit Reportergebnis in Richtext-Feld

Tiefer einsteigen - Agent in der UI aufrufen

- Bisher konnten LotusScript-Agenten nur im Backend ausgeführt werden
- Keine Möglichkeit, UI-Änderungen von einem Agenten aus durchzuführen
  - Neuheit in der API in 8.5.1
  - Neue Funktion, um einen Agent in der Notes UI auszuführen
  - z.B. um auf das aktuelle Dokument/die aktuelle Ansicht zuzugreifen und Dialoge anzuzeigen
  - Man kann sogar Agenten anderer Datenbanken ausführen!
- Nützlich für Funktionen, die bisher nicht Teil der API sind
  - Code in einen LotusScript-Agenten schreiben
  - `NotesUIWorkspace.runAgent` aufrufen
  - Callback-Listener nutzen, um bei Abschluss des Agenten benachrichtigt zu werden
  - Daten-Austausch zwischen Eclipse und LotusScript

Verbesserte Eclipse-Selektion

- In 8.5: Eclipse-Selektion begrenzt auf bereits gespeicherte Dokumente aus Ansichten
  - Es wurden hauptsächlich Notes-URLs übergeben
  - Kein Zugriff auf „In-Memory“-Dokumente
- In 8.5.1: Informationen auch über noch nicht gespeicherte Dokumente und sogar fokussierte Felder eines Formulars
  - Es kann verfolgt werden, was der Nutzer editiert!
  - Einführung von Klassen, die mehr Informationen liefern als nur die Notes-URL (z.B. NotesUIElement, NotesUIDocument und NotesUIField)
  - Nutzt Standard Eclipse-Konzept (Adapter), um zusätzliche Daten zu liefern!
- In 8.5.2: Erweiterte Liste von “Property Testern” und mehr Adapter-Unterstützung
  - Property Tester werden verwendet wie “Hide When's” in Definitionen von Top-Level- und Kontext-Menüeinträgen
  - Das OpenNTF-Projekt “Java UI API Exerciser” enthält eine Liste der verfügbaren Property Tester

Verbesserte Eclipse-Selection

- Verwendung von IAdaptable auf Selektionen

```
// iterate over a selection and print the form-name
for(Iterator<?>
  i=((IStructuredSelection)selection).iterator();
  i.hasNext(); ) {
   Object item = i.next();
   if (item instanceof IAdaptable) {
       NotesUIDocument uidoc = (NotesUIDocument)
         ((IAdaptable)item).getAdapter(NotesUIDocument.class);

       if (uidoc != null)
          System.out.println( uidoc.getForm() );
   }
}
```


Demo

- Universelles kontext-basiertes Online Hilfe-System
  - Sidebar-Hilfe für aktuelle Notes-Inhalte und XPages
  - Agenten aus Eclipse aufrufen und Daten übermitteln

Agenda

- Einführung
- Lotus Notes mit Java erweitern
- Domino Designer mit Java erweitern
- Mehrwert für das klassische Lotus Notes und XPages
- Best Practices
- Q&A

Domino Designer Extensibility API

- Die DDE API ermöglicht Programmierung von Java-Erweiterungen der Domino Designer IDE
- Einführung neuer Funktionalität
  - Design-Element-Information der aktuellen Eclipse-Selektion
  - Basis-Daten von Design-Elementen oder Datenbanken setzen
  - Aktualisieren von Projekten/einzelnen Designelementen nach einer Änderung im Hintergrund (z.B. durch einen DXL-Import)
  - Datenbanken im DDE-Navigator öffnen
- Haupteinsatzzweck:
  - Reagieren auf Nutzer-Selektion – z.B. Anzeige zusätzlicher Daten in eigenen Ansichts-Bereichen (Eclipse-Views)
  - Automatische Verarbeitung von Daten anbieten, z.B. Flags für alle selektierten Bilder setzen oder Code-Generatoren, die Design-Elemente erzeugen
- Zusätzliche Erweiterbarkeit gewinnen durch Nutzung von Eclipse-APIs
  - Ein NSF-Projekt ist nur eine Erweiterung von Eclipse IProject
- API unverändert in 8.5.2 gegenüber 8.5.1

Eclipse-Selektion in DDE API Objekte umwandeln

- Konvertiert ein Eclipse IProject in ein DesignerProject
  - Ein IProject ist ein generisches Entwicklungs-Projekt in der Eclipse IDE

```
DesignerProject nsfProject =
   DesignerResource.getDesignerProject(iproject);
String dbServer = nsfProject.getServerName();
String dbPath = nsfProject.getDatabaseName();

//
//modify db design here, then notify DDE about changes
//

nsfProject.refresh();
```


Eclipse-Selektion in DDE API Objekte umwandeln

- Konvertiert Eclipse IResource in ein DesignerDesignElement
  - Eine IResource ist ein generisches Unterelement eines Eclipse IProjects

```
DesignerDesignElement de =
   DesignerResource.getDesignElement(iresource);
String oldName = de.getName();

//
//modify design element here, then notify DDE about changes
//

de.refresh();
```


Demo

- Eigene Eigenschaften für Notes Design-Elemente
- Automatische Design-Manipulation
- LotusScript.doc-Support im Designer nachrüsten

Agenda

- Einführung
- Lotus Notes mit Java erweitern
- Domino Designer mit Java erweitern
- Mehrwert für das klassische Lotus Notes und XPages
- Best Practices
- Q&A

Grenzen der API

- Klingt toll! Und wo ist der Haken?
  - Wir wollen nicht nur zeigen, was möglich ist, sondern auch was nicht möglich ist
- Nur ein kleiner Teil der LotusScript API
- Event Listener blockieren nicht bei Form/View-Events
  - Kein Ersatz für LotusScript Events, z.B. QuerySave
- Richtung Eclipse => klassisches Notes
  - Hilft nur der Eclipse-Welt, keine Verbesserungen für die Welt des klassischen Clients
- Irgendwelche Vorteile für XPages-Entwickler?

Was können wir tun, um den klassischen Client, XPages-Anwendungen und Eclipse enger zusammen zu bringen?

→ Wie können wir Eclipse-Funktionen von LotusScript aus nutzen?

→ Ist die UI API relevant für XPages-Entwickler?

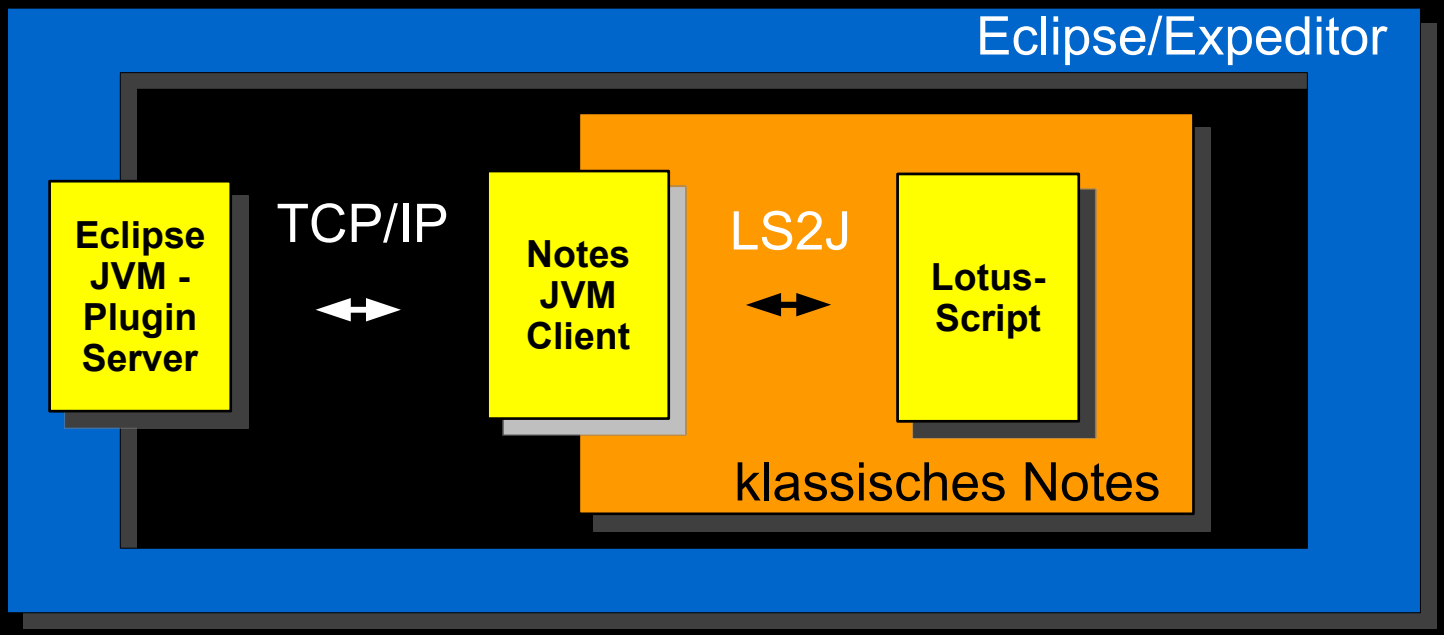
Verbindung zweier Welten

- Beim EC10* haben wir verschiedene Ansätze vorgestellt, um den klassischen Notes Client mit Eclipse zu verbinden und kamen zu einer Lösung:
- Zwei JVMs: Eclipse JVM und klassische Notes JVM
  - Keine direkte Verbindung zwischen Eclipse-Plugins und Notes-Agenten
- Benutze lokale Netzwerk-Kommunikation zwischen beiden
  - Öffne einen Server-Port in Eclipse, verbinde darauf von der klassischen JVM
- Benutze eigenes Protokoll oder Industrie-Standards wie z.B. Java RMI
  - Und rufe remote Eclipse-Plugin-Code vom klassischen Notes auf
- Optional: Nutze LS2J für den Zugriff auf die Remote-API in LotusScript

* Präsentation erhältlich in unserem Blog

Verbindung zweier Welten

- Das Resultat war beeindruckend:
  - Kombination von LotusScript und Eclipse-Plugins ermöglicht neue Entwurfsmuster
  - z.B. Hintergrund-Threads für lange laufenden LotusScript-Code oder Erzeugung von Eclipse-Reitern und Layouts durch LotusScript on-the-fly

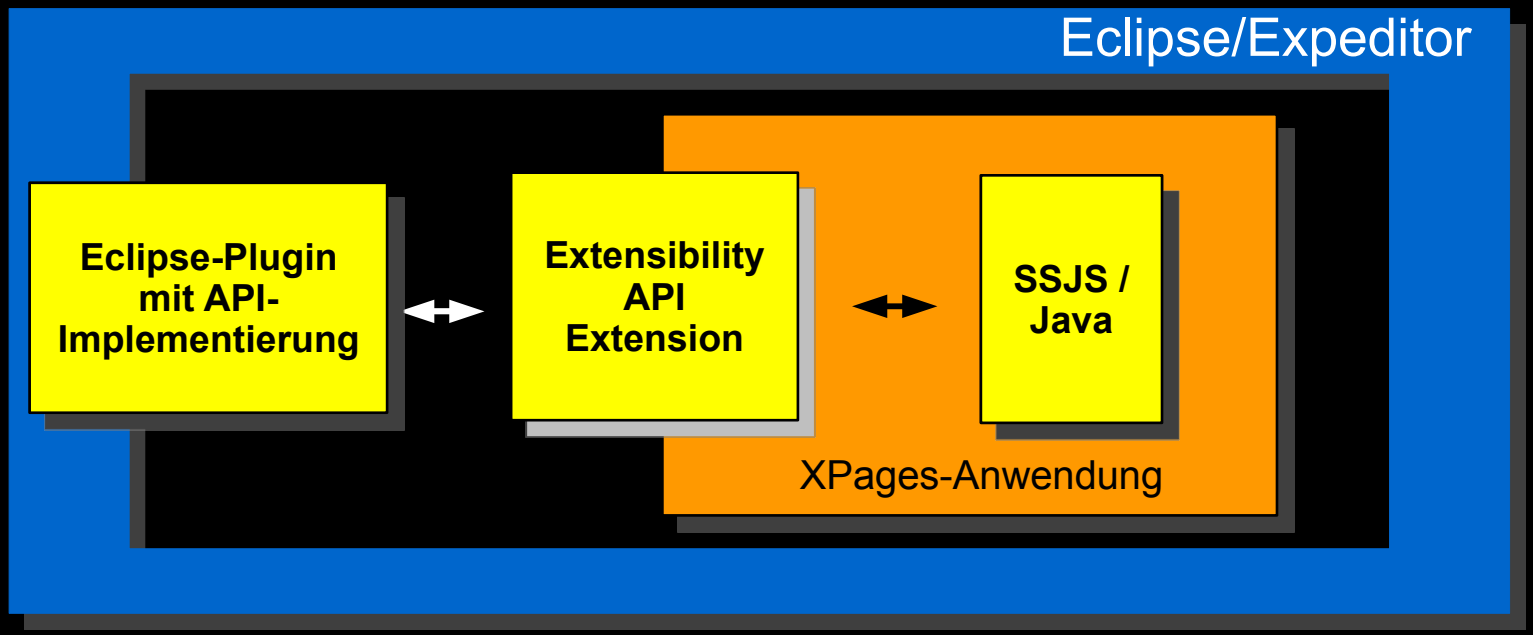


Demo

- LotusScript registriert Kontextmenü-Aktionen
- Multi-Threading in LotusScript-Anwendung

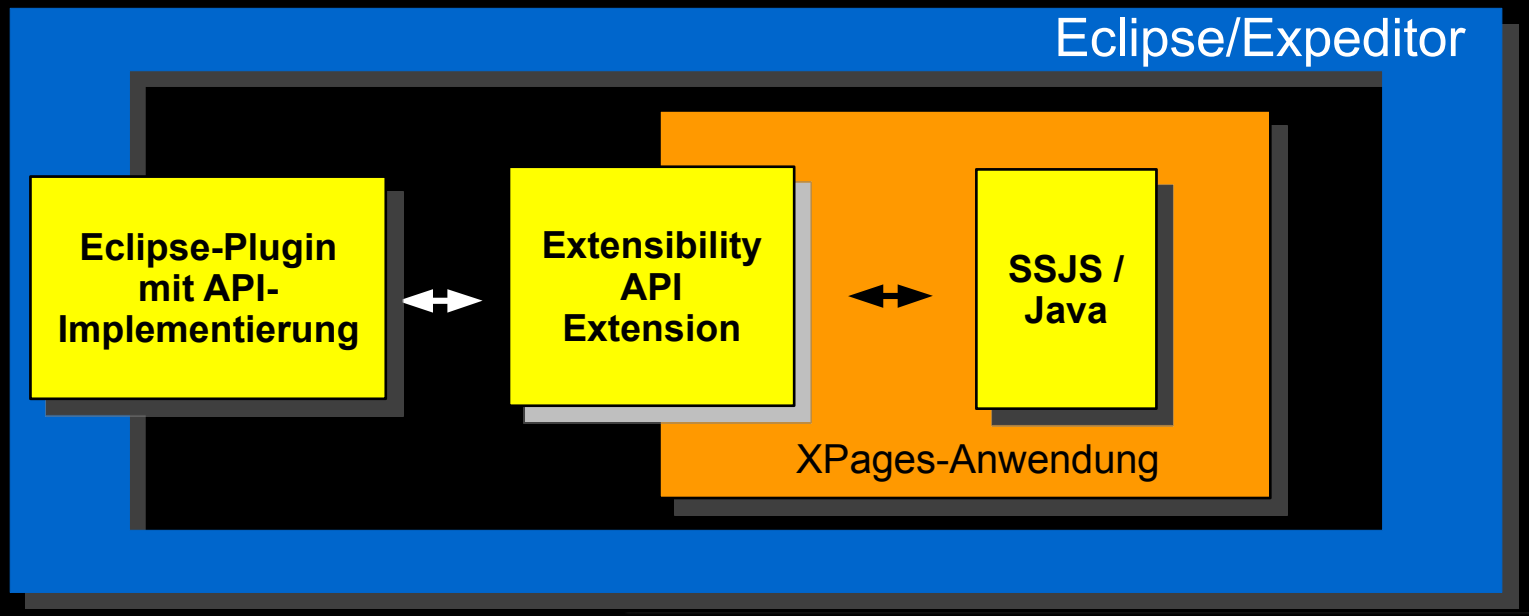
Verbindung zweier Welten

- Lasst uns dasselbe mit XPages im lokalen Client versuchen!
  - Von Expeditor APIs profitieren, wenn XPages-Anwendung lokal läuft
  - XPages-Anwendung wird zu mehr als einer lokalen Web-Anwendung
  - UI API verwenden, um klassische Notes Client-Anwendungen von XPages-Code aus zu steuern
  - Einfache Integration dank der XPages Extensibility API von Lotus Notes 8.5.2
- Hier ist ein Überblick:



Verbindung zweier Welten

- Einige Hinweise zur Implementierung:
  - Nutzt neuen OSGi-Support für XPages in 8.5.2:
  - XPages-Anwendungen mit Java-Controls erweitern, Code befindet sich in Eclipse-Plugins
  - XPages-Editor fügt Plugin-Abhängigkeit zur NSF hinzu, wenn solch ein Java-Control das erste Mal in einer XPage verwendet wird
  - Eclipse-Plugins können außerhalb des XPages SecurityManagers laufen (keine SecurityExceptions!) und Eclipse APIs aufrufen



Demo

- Neue Email aus einer XPages-Anwendung erzeugen
- Lang laufende SSJS-Aufrufe als Eclipse-Jobs visualisieren
- Dynamische Code-Ausführung von LotusScript mit Notes-UI-Zugriff
- Neue Eclipse-Perspektiven und ViewParts von SSJS aus erzeugen

Agenda

- Einführung
- Lotus Notes mit Java erweitern
- Domino Designer mit Java erweitern
- Mehrwert für das klassische Lotus Notes und XPages
- Best Practices
- Q&A

Generelle Empfehlungen

- Das Arbeiten mit Threads erlernen
  - Keine lang andauernden Operationen im UI-Thread ablaufen lassen!
  - Dies blockiert den gesamten Client!
  - Berechnungen in Hintergrund-Jobs auslagern – dann einen UIJob zum Update der UI nutzen

```
NotesSessionJob job = new NotesSessionJob("BG Operation") {
  protected IStatus runInNotesThread(
      Session session, IProgressMonitor monitor)
        throws NotesException {
      //compute something here
      return Status.OK_STATUS;
  }
};
job.schedule();
```


Generelle Empfehlungen

- Keine Notes-Objekte cachen
  - Kann zu schweren Problemen mit dem Memory-Management führen
- Die Notes Java API hat nur eine begrenzte Anzahl von Handles für Daten-Objekte
  - Und der eigene Code ist nicht allein im Client!
  - Wenn möglich stets .recycle() aufrufen
- NotesSessionJob für den Notes-Zugriff nutzen
  - Wird im Hintergrund ausgeführt
  - Holt immer eine neue Session; ist sogar dann sicher, wenn die Notes-ID geändert wurde
  - Recycled automatisch alle Notes-Objekte, die innerhalb der Session erzeugt wurden
    → Kopieren der Notes-Daten in eigene Objekte zur Zwischenspeicherung
  - UI API Daten-Klassen sind hingegen sicher (z.B. NotesDocumentData)

Generelle Empfehlungen

- Falls Sie Ihre eigene Brücke zwischen XPages und Eclipse entwickeln, führt das Ausführen von beschränkten API-Operationen eventuell zu SecurityExceptions
  - XPages-Runtime ist geschützt durch SecurityManager, direkte Ausführung von beschränktem Code ist nicht zulässig
- Workaround: Kapseln des beschränkten Plugin-Codes in AccessController-Aufrufe
  - Deaktiviert die SecurityManager-Prüfung für einen Code-Block

```
T result=
 AccessController.doPrivileged(new PrivilegedAction<T>() {
  public T run() {
    // this code runs out of security manager scope
    // be careful not to open security holes!
   T newT=buildT();
   return newT;
  }
 });
```


Zusammenfassung

- Eclipse-Entwickler erhalten zusätzliche Möglichkeiten, um mit der Lotus Notes UI interagieren zu können
  - Existierender LotusScript-Code kann wiederverwendet werden (Aufruf als UI-Agent)
  - Kann für sanften Übergang von Notes-Code zu XPages und Java-Plugins genutzt werden
- Klassische Notes-Entwicklung kann auch von den APIs profitieren
  - Durch eine Brücke zwischen LotusScript und Eclipse-Plugins können API-Funktionen auch von klassischen Formularen und Ansichten genutzt werden
  - Interessante neue Entwurfsmuster wie Multi-Threading in LotusScript-Anwendungen und Verwendung von Eclipse-UI-Komponenten (Perspective, View)
- In 8.5.2 können XPages-Entwickler ebenfalls Eclipse-Plugin-Code in ihren Anwendungen verwenden
  - UI und Eclipse APIs nutzen zur Verbesserung der Benutzerakzeptanz bei lokaler Ausführung und Integration klassischer Designelemente
- DDE kann jetzt ebenso erweitert werden
  - Selektion von Design-Elementen auswerten, um Design zu modifizieren und z.B. Code/Design-Generatoren zum DDE hinzuzufügen

Vielen Dank!

Zeit für Fragen