Domino - Eclipse

Jörn Schwarze

Auf dem Weg zu neuen Ufern

oder

wie kann ich eine moderne IDE für Entwicklungen in Lotus Domino nutzbar machen?

Der Fahrplan

- 1. Der Schmerz
- 2. Die Idee!
- 3. Die Möglichkeiten
- 4. Der Ansatz
- 5. Der Ausblick

1. Der Schmerz – Im Designer

- Stiefmütterliche Unterstützung der Java-Entwicklung
  - Context-Hilfe – fehlt
  - Auto-Vervollständigung – fehlt
  - Javadoc-Unterstützung – fehlt
  - Rudimentärer Editor
- Schlechte Möglichkeiten zum Debuggen
- Offene IDE kann nicht oder nur bedingt genutzt werden
- Kein CVS, kein „richtiges Projekt“

1. Der Schmerz – Parallele Nutzung

- Mühsamer Weg des „Deployments“
- Viele Fenster um eine Entwicklung zu testen
- Fehlende Umgebung für echtes Debugging (AgentContext, CurrentDatabase, ...)

2. Die Idee

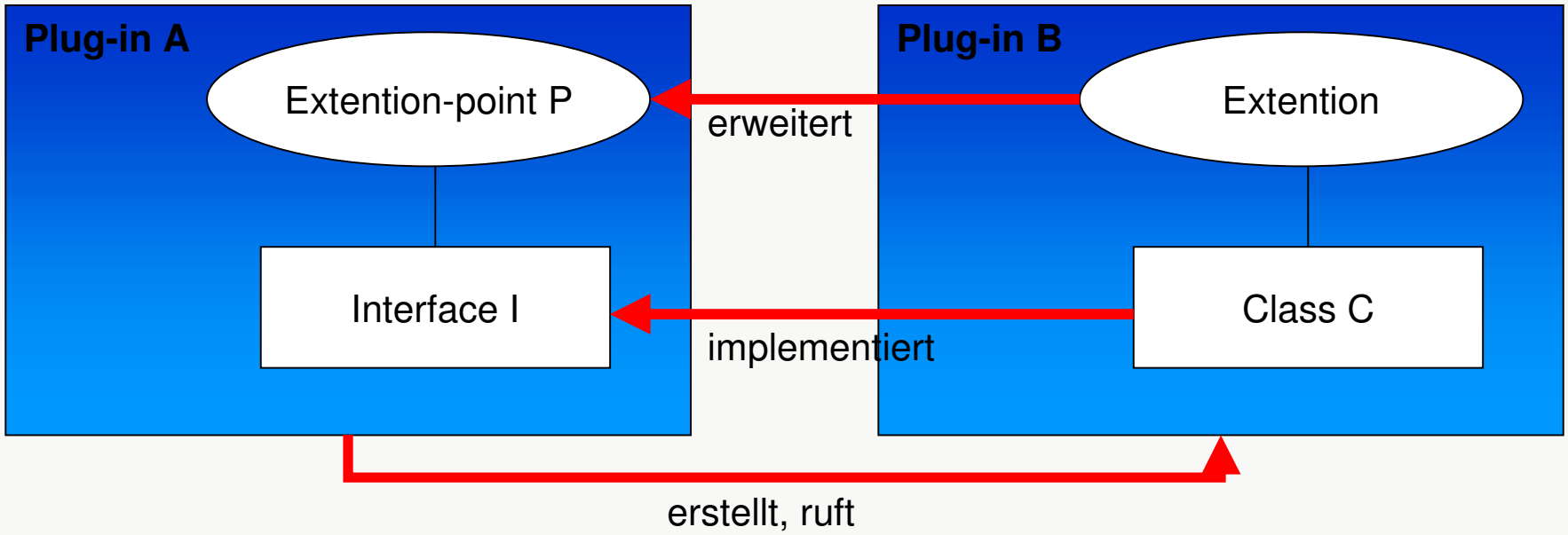
- Viele Entwicklungen bereits in Eclipse
- „Da müßte man doch mal was machen...“
- Plattform zur Domino-Entwicklung innerhalb von Eclipse schaffen
- Erweiterbarkeit der Plugin-Technologie nutzen (offene Schnittstellen schaffen)

3. Die Möglichkeiten - Domino

- DXL als Austauschformat für (Design)dokumente
- DXLImporter / DXLExporter als Java API
- DominoConsole als Möglichkeit zur Serversteuerung/Überwachung
- Notes-URLs zur Ausführung

3. Die Möglichkeiten - Eclipse

Plug-in-Konzept



- Plug-in A
  - Deklariert Extension-Point A
  - Deklariert Interface I
- Plug-in B
  - Implementiert I mit C
  - Stellt C an P zur Verfügung

- Plug-in A
  - Instanziiert C
  - Ruft Methoden von I

3. Die Möglichkeiten - Eclipse

Plugin.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin
  id="de.twentyone.example"
  name="Example-Plugin"
  class="de.twentyone.example.ExamlpePlugin">
  <requires>
    <import
      plugin="org.eclipse.core.resources"/>
    <import
      plugin="org.eclipse.ui"/>
  </requires>
  <runtime>
    <library name="example.jar"/>
  </runtime>
  <extention
    point="org.eclipse.ui.preferencepages">
    <page
      id="de.twentyone.example.preferences"
      icon="icons/img.gif"
      title="Example"
      class="de.twentyone.example.preferences.ExamplePage"/>
  </extention>
  <extention-point
    name="MyExtentionPoint"
    id="de.twentyone.example.extentionpoint"/>
</plugin>
```

Plug-in Kennung

Andere, benötigte Plug-ins

Plug-in Code

Erweiterungen, welche dieses Plugin bedient

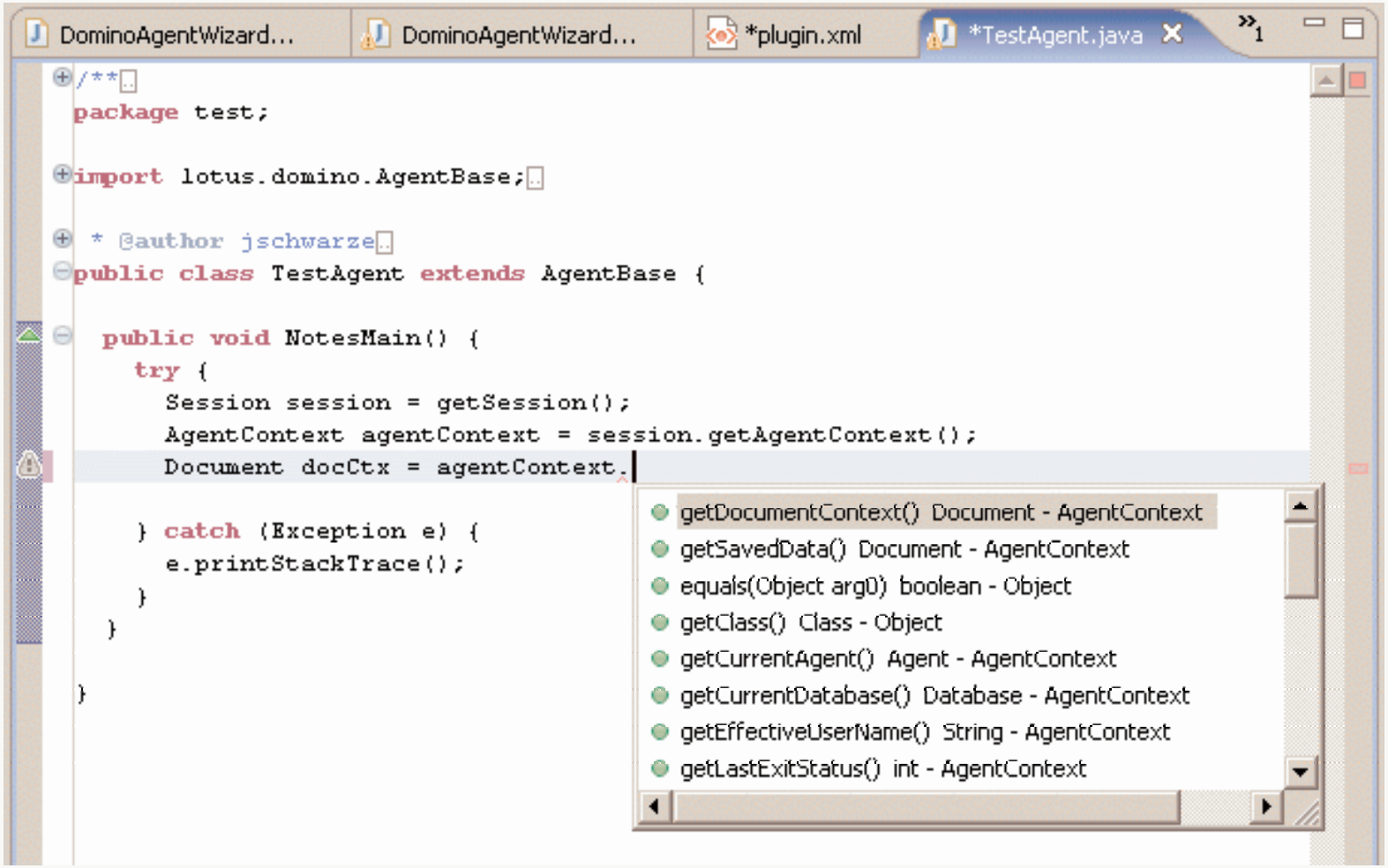
Erweiterungspunkt für andere Plug-ins

3. Die Möglichkeiten - Eclipse

- Demo – Plugin Erstellung in Eclipse
  - New...\Project...\Plugin-Project
  - Projektname
  - Wizard wählen
  - Starten

3. Die Möglichkeiten - Eclipse

Der Java-Editor



3. Die Möglichkeiten - EMF

- **Eclipse Modeling Framework**
- Java Framework für Objekt-Modelle
- Code-Generator
- Schema-Basiert
- Rational-Rose, XSD, UML2
- Kostenlos, auf eclipse.org

4. Der Ansatz – Was bisher geschah.

- Domino-Konsole
- Datenbank-Browser
- Java-Agent-Wizard
- Import-Export-Routine für DXL
- DXL-Model und -Editor

5. Der Ausblick

- Fertigstellung des Java-Agent-Wizards
- Debug-Umgebung
- Erweiterung der Domino-Konsole
  - Gespeicherte Befehlsfolgen (Scripte?)
  - Kommunikation mit DC über GUI
  - Start aus Eclipse
- Lotus-Script-Editor ???
- Plugin als Open-Source-Projekt
- Beteiligung an der Entwicklung / Vorschläge sind herzlich willkommen

Das Ende

- Vielen Dank!

twenty ONE