GUI for SOFA development

Maksym Nesen
DISTRIBUTED SYSTEMS RESEARCH GROUP
http://nenya.ms.mff.cuni.cz

CHARLES UNIVERSITY PRAGUE
Faculty of Mathematics and Physics
Plan

• Eclipse IDE
• General plug-in description
• Under the cover…
• Graphical editor description
• Graphical editor features
• Graphical editor developer plan
• Summary
• Discussion
The Eclipse platform is structured around the concept of **plug-ins**

Each subsystem in the platform is itself structured as a set of plug-ins that implement some key function.

The Eclipse SDK includes the basic platform plus two major tools that are useful for plug-in development.
Eclipse IDE

Java Development Tooling (JDT)

Plug-in Developer Environment (PDE)

Eclipse Platform

- Workbench
- JFace
- SWT

Workspace

Help

Team

New Tool

New Tool

New Tool

Platform Runtime

Eclipse SDK

Maksym Nesen
Eclipse IDE - Plugging into the workbench

- The workbench is the cockpit for navigating all of the function provided by plug-ins.
Eclipse IDE - Plugging into the workbench

• **Editors** allow the user to edit something in the workbench. Editors are "document-centric," much like a file system editor.

• **Views** provide information about some object that the user is working with in the workbench.
In the workspace, a plug-in is represented by a single project that has a plugin.xml (manifest) file at its root and that encapsulates all the code and resources of the plug-in.
Eclipse IDE – plugin.xml

```xml
<extension>
  point="org.eclipse.ui.intro.configExtension">
    <configExtension
      content="intro/sample.xml"
      configId="org.eclipse.ui.intro.universalConfig">
    </configExtension>
  </extension>
</extension>
<extension>
  point="org.eclipse.ui.views">
    <category
      name="Repository Category"
      id="RepositorySOFAPPlugin">
    </category>
    <view
      name="Repository View"
      icon="icons/small.gif"
      category="RepositorySOFAPPlugin"
      class="repositorysofaplugn.views.RepositoryView"
      id="repositorysofaplugn.views.RepositoryView">
    </view>
</extension>
```
**Graphical Editor plug-in concept**

- Graphical Editor plug-in (GEp) should be 100% Eclipse compatible
- GEp should support fully functional editing in Graphic Mode
- GEp should convert graphics to code and back.
- GEp should provide additional features for developer
GEp features

- GUI editor
- Java editor
- XML view (read only)
- Views
- Properties
GEp – graphical editor
GEp – tree view
GEp – component properties view

Name: Helloworld.HelloArchitecture1
Implementation: org.objectweb.dsrsg.sofa.examples.helloworld.HelloWorld1
Version: 2.0

Save  Revert
GEP – Listener model
GEn – Listner model
GEp – Listner model
Maksym Nesen
Maksym Nesen

GEp – developing plan

- Create Plug-in stub for Eclipse.
- Plug-in to existing repository server.
- Create tree of existing components
- Create Eclipse view for repository tree
- Create properties sequence for each tree component
- Create appropriate listeners
- Create draw area instead of standard text editor
- Update listener accordingly to draw area.
- All drawn objects should be “live”
- Draw area should give a possibility to create new objects easily.
- Java code should be generated automatically or by event
- XML read only view
- All editor properties should be easily configurable
- Java code editor provides all features.
- Commitment to repository server can be done easily.
- All project – related settings should be saved in local file
Summary

• Eclipse IDE
• Plugging into workbench
• Plugin.xml overview.
• GEp description
• GEp features
• GEp development plan
• Summary
Questions

- You are welcome.