Sistemas baseados em plugins com Eclipse

ARCHIMEDES

The Open CAD

O que é o Archimedes?

- Um projeto *open source* do IME
- Um CAD para arquitetos
- Um software baseado em comandos

História do projeto

- Projeto da matéria de XP em 2006
- Projeto de TCC em 2006
- Projeto de XP em 2007

Estado no fim de 2006

- Não muito longe de versão decente
- Classes enormes para interface
- Código repetido para coisas semelhantes mas diferentes

Motivações para mudanças

- Facilitar desenvolvimento distribuído
- Forçar “componentização”
- Promover a reutilização de código

O que é o Eclipse?

- Uma das melhores IDEs para Java
- Um software open source excelente
- Uma interface nativa em Java

O que é RCP?

- *Eclipse's Rich Client Platform*
- Um projeto altamente O.O.
- *Framework* para Aplicações *Desktop*

A estrutura do RCP

- Um pequeno núcleo
- Bibliotecas gráficas do SWT
- Definições de XMLs

Funcionamento normal

- Boot do programa (.product)
- Carregamento dos plugins
- Montagem da interface gráfica

Carregando um plugin

- Busca por plugin.xml
- Verificação de dependências
- Instanciação das classes necessárias

Modo de ação

- Associações indiretas via xml
- Utilização por outro componente
- Chamadas do framework ao plugin

Plugin HelloWorld

- Uma ação simples
- Criou plugin.xml e dependências
- Apenas 1 classe “útil”

Exemplos no Archimedes

- Help e Update Actions
- Open layer Action
- SelectAll Action

Perspective e *View*

- Permite organizar a interface
- Adicionar elementos de visualização
- Tudo flexível como no Eclipse

Extensões mais avançadas

- Editores para tipos de arquivos
- Atualizações internas
- Sistema de ajuda completo

Help e Update no Archimedes

- Infraestrutura pronta
- Maior trabalho no conteúdo
- Processo feito pelo framework

Não é o suficiente!

- Definição de um ponto de extensão
- Criação de um framework próprio
- Flexibilidade garantida

Comandos sem extensão

```
public CommandParser () {

    commands = new HashMap<String, CommandFactory>();
    addCommand(new ArcFactory());
    addCommand("a", commands.get("arc"));
    addCommand(new AreaPerimeterFactory());
    addCommand(new CircleFactory());
    addCommand("c", commands.get("circle"));
    addCommand(new CopyPasteFactory());
    addCommand("co", commands.get("copy"));
    addCommand(new CopyToClipboardFactory());
    addCommand(new DimensionFactory());
    addCommand(new DistanceFactory());
    addCommand("dist", commands.get("distance"));
    addCommand("d", commands.get("distance"));
    addCommand(new EditTextFactory());
    addCommand(new EraseFactory());
    addCommand(new ExplodeFactory());
    addCommand("x", commands.get("explode"));
    addCommand("e", commands.get("erase"));
    addCommand(new FilletFactory());
    addCommand("f", commands.get("fillet"));
    addCommand(new InfiniteLineFactory());
    addCommand("xl", commands.get("infiniteline"));
    addCommand("xline", commands.get("infiniteline"));
    addCommand(new LayOffFactory());
    addCommand(new LayOnFactory());
    addCommand(new LeaderFactory());
    addCommand(new LineFactory());
    addCommand("l", commands.get("line"));
    addCommand(new MirrorFactory());
    addCommand("mi", commands.get("mirror"));
    addCommand(new MoveFactory());
    addCommand("m", commands.get("move"));
    addCommand(new OffsetFactory());
    addCommand("o", commands.get("offset"));
    addCommand(new OrtoFactory());
    addCommand(new PanFactory());
    addCommand("p", commands.get("pan"));
    addCommand(new PasteFactory());
    addCommand(new PolyLineFactory());
    addCommand("pl", commands.get("polyline"));
    addCommand(new RectangleFactory());
    addCommand("rect", commands.get("rectangle"));
    addCommand("rec", commands.get("rectangle"));
    addCommand("r", commands.get("rectangle"));
    addCommand(new RedoFactory());
    addCommand("re", commands.get("redo"));
    addCommand(new RotateFactory());
    addCommand("ro", commands.get("rotate"));
    addCommand(new ScaleFactory());
    addCommand(new TextFactory());
    addCommand(new TrimExtendFactory(false));
    addCommand("tr", commands.get("trim"));
    addCommand(new TrimExtendFactory(true));
    addCommand("ex", commands.get("extend"));
    addCommand(new UndoFactory());
    addCommand("u", commands.get("undo"));
    addCommand(new ZoomFactory());
    addCommand("z", commands.get("zoom"));
    addCommand(new StretchFactory());
}
```

Comandos com extensão

```
private void loadFactories() {

    IExtensionPoint extensionPoint = Platform.getExtensionRegistry()
            .getExtensionPoint("br.org.archimedes.factory");
    if (extensionPoint != null) {
        IExtension[] extensions = extensionPoint.getExtensions();
        for (IExtension extension : extensions) {
            IConfigurationElement[] configElements = extension
                    .getConfigurationElements();
            for (IConfigurationElement element : configElements) {
                CommandFactory factory = parseFactory(element,
                        "factory", "class");
                if (factory != null) {
                    addFactory(element, factory);
                }
            }
        }
    }
}
```

factory.exsd

```
<?xml version='1.0' encoding='UTF-8'?>
<schema targetNamespace="br.org.archimedes">
...
   <element name="factory">
      <annotation>
         <documentation>Bla</documentation>
      </annotation>
      <complexType>
         <attribute name="id" type="string" use="required"/>
         <attribute name="class" type="string" use="required">
            <annotation>
               <documentation>Bla</documentation>
               <appInfo><meta.attribute kind="java"
  basedOn="br.org.archimedes.interfaces.CommandFactory:"/></appIn
  fo>
            </annotation>
         </attribute>
         <attribute name="handlesDoubleClick" type="boolean"/>
         <attribute name="handledElementId" type="string"/>
      </complexType>
   </element>
...
</schema>
```

Código do plugin

```
public class Erase extends Selector implements CommandFactory {
    private UndoableCommand command;
    protected String getCancelMessage () {
        return “Erase canceled”;
    }
    public String getName () {
        return "erase";
    }
    public List<Command> getCommands () {
        List<Command> cmds = new ArrayList<Command>();
        if (command != null) {
            cmds.add(command);
        }
        return cmds;
    }
    protected String finish (Set<Element> selection) {
        try {
            command = new PutOrRemoveElementCommand(selection, true);
        } catch (NullArgumentException e) {...}
        return “Selection erased”;
    }
}
```

Quando preciso estender?

- O núcleo ou outro plugin usam
- Manter registro dos plugins
- Outros instanciam

Quando não precisa ser

- Classes de plugins “amigos”
- Objetos que implementam interface
- Plugins interligados

Flexibilidade da solução

- Aproveitar código existente
- Definir um framework
- Misturar plugins

Possíveis melhoras

- Carregamento dinâmico de plugins
- Sistema de recuperação de erros
- Herança entre arquivos exsd

Perguntas? Comentários?

Hugo Corbucci
corbucci@ime.usp.br

http://www.archimedes.org.br