# AST 使用笔记

**vwpolo**

http://blog.csdn.net/vwpolo/archive/2008/04/29/2343970.aspx

最近公司要求我做一个可以生成代码的工具，需求为像 SWT-Designer 那样，多页编辑器，一页显示源代码，另外一页用表格的形式显示类的属性，要求可以修改，这也不是什么和困难的事情，不过以前没做过这样的东西，还得找找资料来研究研究，发现有个叫 AST 的东西，是 Eclipse 提供的工具，以前听说过，不过不怎么用，经过一段时间的学习，发现这东西还真不赖，功能挺强大的，像 Eclipse 的重构功能就是通过这个 AST 来实现的，AST 全名叫抽象语法树，不过功能强是强大，用起来有点麻烦，很容易出错。

今天在用这个 AST 生成一个 Java 文件，我是先通过 SWT-Designer 生成了一个 JFace 的 ApplicationWindow 的文件，上面就简单的放一个按钮，单击后弹出一个对话框，然后显示信息，功能是很简单，经过一天的不断尝试，终于通过了编译，直接运行出来了，可以看到我的对话框了，那下面来介绍一下怎么使用这个东西

```
    ASTParser parser = ASTParser.newParser(AST.JLS3);
    parser.setSource("".toCharArray());
    CompilationUnit unit = (CompilationUnit) parser.createAST(null);
    unit.recordModifications();
    AST ast = unit.getAST();
```

上面的代码是生成一个空的编译单元，也就是我们的 java 文件了，我们通过 ASTParser.newParser(AST.JLS3);来指定遵守的 Java 规范,如果是 JLS2 的话是指定 Java 1.4 的编译规范，那么 JLS3 为遵守 Java 5 以上的规范。

```
    PackageDeclaration packageDeclaration = ast.newPackageDeclaration();
    unit.setPackage(packageDeclaration);
    packageDeclaration.setName(ast.newSimpleName("astdemo"));
```

上面是生成 Java 文件的包， 结果为 package astdemo;

```
    //类名
    TypeDeclaration classType = ast.newTypeDeclaration();
    classType.setInterface(false);
    List classTypeModifier = classType.modifiers();
    classTypeModifier.add(ast.newModifier(ModifierKeyword.PUBLIC_KEYWORD));
    classType.setName(ast.newSimpleName("MyFirstApp"));
    classType.setSuperclassType(ast.newSimpleType(ast.newSimpleName("Applicat
ionWindow")));
    unit.types().add(classType);
```

这里为指定生成文件的类型，可以在这里指定生成类的名称，是生成的是否是接口,还有访问类型，结果是

```
    public class MyFirstApp extends ApplicationWindow {
    }
```

```
     //构造方法
    MethodDeclaration methodConstructor = ast.newMethodDeclaration();
    methodConstructor.setConstructor(true);
    List methodConstructorModifier = methodConstructor.modifiers();
    methodConstructorModifier.add(ast.newModifier(ModifierKeyword.PUBLIC_KEYW
ORD));
    methodConstructor.setName(ast.newSimpleName("MyFirstApp"));
    classType.bodyDeclarations().add(methodConstructor);
```

上面为生成一个方法，并指定这个方法是构造方法，生成结果

```
    public MyFirstApp();
```

哎，算了，这样太费劲了，下面是程序的代码

```
    package com.vwpolo.jet.example;

    import java.util.ArrayList;
    import java.util.List;
    import java.util.StringTokenizer;

    import org.eclipse.jdt.core.dom.AST;
    import org.eclipse.jdt.core.dom.ASTParser;
    import org.eclipse.jdt.core.dom.AnonymousClassDeclaration;
    import org.eclipse.jdt.core.dom.Block;
    import org.eclipse.jdt.core.dom.ClassInstanceCreation;
    import org.eclipse.jdt.core.dom.CompilationUnit;
    import org.eclipse.jdt.core.dom.ImportDeclaration;
    import org.eclipse.jdt.core.dom.InfixExpression;
    import org.eclipse.jdt.core.dom.MethodDeclaration;
    import org.eclipse.jdt.core.dom.MethodInvocation;
    import org.eclipse.jdt.core.dom.PackageDeclaration;
    import org.eclipse.jdt.core.dom.PrimitiveType;
    import org.eclipse.jdt.core.dom.ReturnStatement;
    import org.eclipse.jdt.core.dom.SingleVariableDeclaration;
    import org.eclipse.jdt.core.dom.StringLiteral;
    import org.eclipse.jdt.core.dom.SuperConstructorInvocation;
    import org.eclipse.jdt.core.dom.TypeDeclaration;
    import org.eclipse.jdt.core.dom.VariableDeclarationFragment;
    import org.eclipse.jdt.core.dom.VariableDeclarationStatement;
    import org.eclipse.jdt.core.dom.InfixExpression.Operator;
    import org.eclipse.jdt.core.dom.Modifier.ModifierKeyword;

    public class JavaASTParserExample1 {

    private      static      final      String[]      IMPORTS      =
{ "org.eclipse.jface.action.*",
          "org.eclipse.jface.window.*", "org.eclipse.swt.events.*",
          "org.eclipse.swt.*", "org.eclipse.jface.dialogs.*",
          "org.eclipse.swt.graphics.*",
"org.eclipse.swt.widgets.*" };

    public static void main(String[] args) {
        CompilationUnit unit = getCompilationUtit();
        System.out.println(unit.toString());
    }

    @SuppressWarnings("unchecked")
    private static CompilationUnit getCompilationUtit() {
        ASTParser parser = ASTParser.newParser(AST.JLS3);
        parser.setSource("".toCharArray());

        CompilationUnit unit = (CompilationUnit) parser.createAST(null);
        unit.recordModifications();

        AST ast = unit.getAST();

        PackageDeclaration packageDeclaration =
ast.newPackageDeclaration();
        unit.setPackage(packageDeclaration);
        packageDeclaration.setName(ast.newSimpleName("astdemo"));

        for (int i = 0; i < IMPORTS.length; ++i) {
            ImportDeclaration importDeclaration =
ast.newImportDeclaration();
```

```
    importDeclaration.setName(ast.newName(getSimpleNames(IMPORTS[i]))
);
            if (IMPORTS[i].indexOf("*") > 0)
                importDeclaration.setOnDemand(true);
            else
                importDeclaration.setOnDemand(false);

            unit.imports().add(importDeclaration);
        }

        // 类名
        TypeDeclaration classType = ast.newTypeDeclaration();
        classType.setInterface(false);
        List classTypeModifier = classType.modifiers();

    classTypeModifier.add(ast.newModifier(ModifierKeyword.PUBLIC_KEYW
ORD));
        classType.setName(ast.newSimpleName("MyFirstApp"));
        classType.setSuperclassType(ast.newSimpleType(ast
                .newSimpleName("ApplicationWindow")));
        unit.types().add(classType);

        // 构造方法
        MethodDeclaration methodConstructor =
ast.newMethodDeclaration();
        methodConstructor.setConstructor(true);
        List methodConstructorModifier = methodConstructor.modifiers();
        methodConstructorModifier.add(ast
                .newModifier(ModifierKeyword.PUBLIC_KEYWORD));
        methodConstructor.setName(ast.newSimpleName("MyFirstApp"));
        classType.bodyDeclarations().add(methodConstructor);

        Block constructorBlock = ast.newBlock();
        methodConstructor.setBody(constructorBlock);

        // super(null);
        SuperConstructorInvocation superConstructorInvocation = ast
                .newSuperConstructorInvocation();
        constructorBlock.statements().add(superConstructorInvocation);

    superConstructorInvocation.arguments().add(ast.newNullLiteral());

        // createActions();
        MethodInvocation methodInvocation = ast.newMethodInvocation();
        methodInvocation.setName(ast.newSimpleName("createActions"));
        constructorBlock.statements().add(
                ast.newExpressionStatement(methodInvocation));

        // addToolBar(SWT.FLAT | SWT.WRAP);
        MethodInvocation methodInvocation2 = ast.newMethodInvocation();
        methodInvocation2.setName(ast.newSimpleName("addToolBar"));
        InfixExpression infixExpression = ast.newInfixExpression();
        infixExpression.setOperator(Operator.OR);

    infixExpression.setLeftOperand(ast.newName(getSimpleNames("SWT.FL
AT")));
        infixExpression
    .setRightOperand(ast.newName(getSimpleNames("SWT.WRAP")));
        methodInvocation2.arguments().add(infixExpression);
        constructorBlock.statements().add(
                ast.newExpressionStatement(methodInvocation2));
```

```
        // addMenuBar();
        MethodInvocation methodInvocation3 = ast.newMethodInvocation();
        methodInvocation3.setName(ast.newSimpleName("addMenuBar"));
        constructorBlock.statements().add(
                ast.newExpressionStatement(methodInvocation3));

        // addStatusLine();
        MethodInvocation methodInvocation4 = ast.newMethodInvocation();
        methodInvocation4.setName(ast.newSimpleName("addStatusLine"));
        constructorBlock.statements().add(
                ast.newExpressionStatement(methodInvocation4));

        MethodDeclaration              methodDeclaration              =
ast.newMethodDeclaration();
        methodDeclaration.setConstructor(false);
        List methodModifiers = methodDeclaration.modifiers();

    methodModifiers.add(ast.newModifier(ModifierKeyword.PROTECTED_KEY
WORD));
        methodDeclaration.setReturnType2(ast.newSimpleType(ast
                .newSimpleName("Control")));

    methodDeclaration.setName(ast.newSimpleName("createContents"));
        classType.bodyDeclarations().add(methodDeclaration);
        Block methodBlock = ast.newBlock();
        methodDeclaration.setBody(methodBlock);

        // createContents(Composite parent) {
        SingleVariableDeclaration variableDeclaration = ast
                .newSingleVariableDeclaration();
        variableDeclaration.setType(ast.newSimpleType(ast
                .newSimpleName("Composite")));
        variableDeclaration.setName(ast.newSimpleName("parent"));
        methodDeclaration.parameters().add(variableDeclaration);

        // Composite container = new Composite(parent, SWT.NONE);
        VariableDeclarationFragment variableFragment = ast
                .newVariableDeclarationFragment();
        variableFragment.setName(ast.newSimpleName("container"));
        VariableDeclarationStatement variableStatement = ast
                .newVariableDeclarationStatement(variableFragment);
        variableStatement.setType(ast.newSimpleType(ast
                .newSimpleName("Composite")));
        ClassInstanceCreation classCreation =
ast.newClassInstanceCreation();
        classCreation
    .setType(ast.newSimpleType(ast.newSimpleName("Composite")));
        variableFragment.setInitializer(classCreation);
        methodBlock.statements().add(variableStatement);
        classCreation.arguments().add(ast.newSimpleName("parent"));

    classCreation.arguments().add(ast.newName(getSimpleNames("SWT.NON
E")));

        // final Button button = new Button(container, SWT.NONE);
        VariableDeclarationFragment variableFragment2 = ast
                .newVariableDeclarationFragment();
        variableFragment2.setName(ast.newSimpleName("button"));
        VariableDeclarationStatement variableStatement2 = ast
                .newVariableDeclarationStatement(variableFragment2);
```

```
        List variableModifier2 = variableStatement2.modifiers();

    variableModifier2.add(ast.newModifier(ModifierKeyword.FINAL_KEYWO
RD));
        variableStatement2.setType(ast.newSimpleType(ast
                .newSimpleName("Button")));
        ClassInstanceCreation             classCreation2             =
ast.newClassInstanceCreation();

    classCreation2.setType(ast.newSimpleType(ast.newSimpleName("Butto
n")));
        variableFragment2.setInitializer(classCreation2);
        methodBlock.statements().add(variableStatement2);

    classCreation2.arguments().add(ast.newSimpleName("container"));

    classCreation2.arguments().add(ast.newName(getSimpleNames("SWT.NO
NE")));

        // button.addSelectionListener(new SelectionAdapter() {});
        MethodInvocation methodInvocation5 = ast.newMethodInvocation();
        methodBlock.statements().add(
                ast.newExpressionStatement(methodInvocation5));
        methodInvocation5.setExpression(ast.newSimpleName("button"));

    methodInvocation5.setName(ast.newSimpleName("addSelectionListener
"));

        ClassInstanceCreation ci = ast.newClassInstanceCreation();

    ci.setType(ast.newSimpleType(ast.newSimpleName("SelectionAdapter"
)));
        methodInvocation5.arguments().add(ci);
        AnonymousClassDeclaration cd =
ast.newAnonymousClassDeclaration();
        ci.setAnonymousClassDeclaration(cd);

        // public void widgetSelected(SelectionEvent e) {
        MethodDeclaration md = ast.newMethodDeclaration();
        md.setConstructor(false);
        List mdModifiers = md.modifiers();

    mdModifiers.add(ast.newModifier(ModifierKeyword.PUBLIC_KEYWORD));
        md.setName(ast.newSimpleName("widgetSelected"));
        cd.bodyDeclarations().add(md);

        SingleVariableDeclaration variableDeclaration2 = ast
                .newSingleVariableDeclaration();
        variableDeclaration2.setType(ast.newSimpleType(ast
                .newSimpleName("SelectionEvent")));
        variableDeclaration2.setName(ast.newSimpleName("e"));
        md.parameters().add(variableDeclaration2);
        Block methodBlock2 = ast.newBlock();
        md.setBody(methodBlock2);

        MethodInvocation methodInvocation6 = ast.newMethodInvocation();

    methodInvocation6.setExpression(ast.newSimpleName("MessageDialog"
));

    methodInvocation6.setName(ast.newSimpleName("openInformation"));
```

```
        MethodInvocation methodInvocation12 = ast.newMethodInvocation();
        methodInvocation12.setName(ast.newSimpleName("getShell"));
        methodInvocation6.arguments().add(methodInvocation12);

        StringLiteral stringLiteral1 = ast.newStringLiteral();
        stringLiteral1.setEscapedValue("\"信息\"");
        StringLiteral stringLiteral2 = ast.newStringLiteral();
        stringLiteral2.setEscapedValue("\"你好\"");
        methodInvocation6.arguments().add(stringLiteral1);
        methodInvocation6.arguments().add(stringLiteral2);
        methodBlock2.statements().add(
                ast.newExpressionStatement(methodInvocation6));

        MethodInvocation methodInvocation7 = ast.newMethodInvocation();
        methodInvocation7.setExpression(ast.newSimpleName("button"));
        methodInvocation7.setName(ast.newSimpleName("setText"));
        StringLiteral stringLiteral3 = ast.newStringLiteral();
        stringLiteral3.setEscapedValue("\"按 钮\"");
        methodInvocation7.arguments().add(stringLiteral3);
        methodBlock.statements().add(
                ast.newExpressionStatement(methodInvocation7));

        // button.setText("按 钮");
        MethodInvocation methodInvocation8 = ast.newMethodInvocation();
        methodInvocation8.setExpression(ast.newSimpleName("button"));
        methodInvocation8.setName(ast.newSimpleName("setBounds"));
        StringLiteral stringLiteral4 = ast.newStringLiteral();
        stringLiteral4.setEscapedValue("\"按 钮\"");
        // button.setBounds(69, 28, 44, 23);
        methodInvocation8.arguments().add(ast.newNumberLiteral("69"));
        methodInvocation8.arguments().add(ast.newNumberLiteral("28"));
        methodInvocation8.arguments().add(ast.newNumberLiteral("44"));
        methodInvocation8.arguments().add(ast.newNumberLiteral("23"));
        methodBlock.statements().add(
                ast.newExpressionStatement(methodInvocation8));

        // return container;
        ReturnStatement returnStatement = ast.newReturnStatement();
        returnStatement.setExpression(ast.newSimpleName("container"));
        methodBlock.statements().add(returnStatement);

        // private void createActions()
        MethodDeclaration             methodDeclaration2             =
ast.newMethodDeclaration();
        methodDeclaration2.setConstructor(false);
        List methodModifiers2 = methodDeclaration2.modifiers();

    methodModifiers2.add(ast.newModifier(ModifierKeyword.PRIVATE_KEYW
ORD));
        methodDeclaration2.setReturnType2(ast
                .newPrimitiveType(PrimitiveType.VOID));

    methodDeclaration2.setName(ast.newSimpleName("createActions"));
        Block methodBlock3 = ast.newBlock();
        methodDeclaration2.setBody(methodBlock3);
        classType.bodyDeclarations().add(methodDeclaration2);

        // protected MenuManager createMenuManager()
        MethodDeclaration             methodDeclaration3             =
ast.newMethodDeclaration();
```

```
        methodDeclaration3.setConstructor(false);
        List methodModifiers3 = methodDeclaration3.modifiers();
        methodModifiers3

    .add(ast.newModifier(ModifierKeyword.PROTECTED_KEYWORD));
        methodDeclaration3.setReturnType2(ast.newSimpleType(ast
                .newName("MenuManager")));

    methodDeclaration3.setName(ast.newSimpleName("createMenuManager")
);
        Block methodBlock4 = ast.newBlock();
        methodDeclaration3.setBody(methodBlock4);
        classType.bodyDeclarations().add(methodDeclaration3);

        // MenuManager menuManager = new MenuManager("menu");
        VariableDeclarationFragment variableFragment3 = ast
                .newVariableDeclarationFragment();
        variableFragment3.setName(ast.newSimpleName("menuManager"));
        VariableDeclarationStatement variableStatement3 = ast
                .newVariableDeclarationStatement(variableFragment3);
        variableStatement3.setType(ast.newSimpleType(ast
                .newSimpleName("MenuManager")));
        ClassInstanceCreation             classCreation3             =
ast.newClassInstanceCreation();
        classCreation3.setType(ast.newSimpleType(ast
                .newSimpleName("MenuManager")));
        StringLiteral stringLiteral5 = ast.newStringLiteral();
        stringLiteral5.setEscapedValue("\"menu\"");
        classCreation3.arguments().add(stringLiteral5);
        variableFragment3.setInitializer(classCreation3);
        methodBlock4.statements().add(variableStatement3);

        // return menuManager;
        ReturnStatement returnStatement2 = ast.newReturnStatement();

    returnStatement2.setExpression(ast.newSimpleName("menuManager"));
        methodBlock4.statements().add(returnStatement2);

        // protected ToolBarManager createToolBarManager(int style) {
        MethodDeclaration             methodDeclaration4             =
ast.newMethodDeclaration();
        methodDeclaration4.setConstructor(false);
        List methodModifiers4 = methodDeclaration4.modifiers();
        methodModifiers4

    .add(ast.newModifier(ModifierKeyword.PROTECTED_KEYWORD));
        methodDeclaration4.setReturnType2(ast.newSimpleType(ast
                .newName("ToolBarManager")));

    methodDeclaration4.setName(ast.newSimpleName("createToolBarManage
r"));
        SingleVariableDeclaration variableDeclaration5 = ast
                .newSingleVariableDeclaration();

    variableDeclaration5.setType(ast.newPrimitiveType(PrimitiveType.I
NT));
        variableDeclaration5.setName(ast.newSimpleName("style"));
        methodDeclaration4.parameters().add(variableDeclaration5);
        Block methodBlock5 = ast.newBlock();
        methodDeclaration4.setBody(methodBlock5);
        classType.bodyDeclarations().add(methodDeclaration4);
```

```
        // MenuManager menuManager = new MenuManager("menu");
        VariableDeclarationFragment variableFragment4 = ast
                .newVariableDeclarationFragment();

    variableFragment4.setName(ast.newSimpleName("toolBarManager"));
        VariableDeclarationStatement variableStatement4 = ast
                .newVariableDeclarationStatement(variableFragment4);
        variableStatement4.setType(ast.newSimpleType(ast
                .newSimpleName("ToolBarManager")));
        ClassInstanceCreation             classCreation4             =
ast.newClassInstanceCreation();
        classCreation4.setType(ast.newSimpleType(ast
                .newSimpleName("ToolBarManager")));
        StringLiteral stringLiteral6 = ast.newStringLiteral();
        stringLiteral6.setEscapedValue("\"menu\"");
        classCreation3.arguments().add(stringLiteral6);
        variableFragment4.setInitializer(classCreation4);
        methodBlock5.statements().add(variableStatement4);

        // toolBarManager
        ReturnStatement returnStatement3 = ast.newReturnStatement();

    returnStatement3.setExpression(ast.newSimpleName("toolBarManager"
));
        methodBlock5.statements().add(returnStatement3);

        MethodDeclaration mainMethod = ast.newMethodDeclaration();
        mainMethod.setConstructor(false);
        List mainMethodModifiers = mainMethod.modifiers();
        mainMethodModifiers
                .add(ast.newModifier(ModifierKeyword.PUBLIC_KEYWORD));
        mainMethodModifiers
                .add(ast.newModifier(ModifierKeyword.STATIC_KEYWORD));

    mainMethod.setReturnType2(ast.newPrimitiveType(PrimitiveType.VOID
));
        mainMethod.setName(ast.newSimpleName("main"));
        Block mainBlock = ast.newBlock();
        mainMethod.setBody(mainBlock);

        SingleVariableDeclaration mainSingleVariable = ast
                .newSingleVariableDeclaration();

    mainSingleVariable.setType(ast.newArrayType(ast.newSimpleType(ast
                .newSimpleName("String"))));
        mainSingleVariable.setName(ast.newSimpleName("args"));
        mainMethod.parameters().add(mainSingleVariable);
        classType.bodyDeclarations().add(mainMethod);

        VariableDeclarationFragment variableFragment5 = ast
                .newVariableDeclarationFragment();
        variableFragment5.setName(ast.newSimpleName("window"));
        VariableDeclarationStatement variableStatement5 = ast
                .newVariableDeclarationStatement(variableFragment5);
        variableStatement5.setType(ast.newSimpleType(ast
                .newSimpleName("MyFirstApp")));
        ClassInstanceCreation             classCreation5             =
ast.newClassInstanceCreation();
        classCreation5.setType(ast.newSimpleType(ast
                .newSimpleName("MyFirstApp")));
```

```
        variableFragment5.setInitializer(classCreation5);
        mainBlock.statements().add(variableStatement5);

        // window.setBlockOnOpen(true);
        MethodInvocation methodInvocation9 = ast.newMethodInvocation();
        methodInvocation9.setExpression(ast.newSimpleName("window"));

    methodInvocation9.setName(ast.newSimpleName("setBlockOnOpen"));

    methodInvocation9.arguments().add(ast.newBooleanLiteral(true));
        mainBlock.statements().add(
                ast.newExpressionStatement(methodInvocation9));

        // window.setBlockOnOpen(true);
        MethodInvocation methodInvocation10 = ast.newMethodInvocation();
        methodInvocation10.setExpression(ast.newSimpleName("window"));
        methodInvocation10.setName(ast.newSimpleName("open"));
        mainBlock.statements().add(
                ast.newExpressionStatement(methodInvocation10));

        // Display.getCurrent().dispose();
        //        MethodInvocation        methodInvocation11        =
ast.newMethodInvocation();
        //
methodInvocation11.setExpression(ast.newName(getSimpleNames("Display.
getCurrent()")));
        // methodInvocation11.setName(ast.newSimpleName("dispose"));
        //
mainBlock.statements().add(ast.newExpressionStatement(methodInvocatio
n11));

        return unit;
    }

    @SuppressWarnings("unchecked")
    static private String[] getSimpleNames(String qualifiedName) {
        StringTokenizer st = new StringTokenizer(qualifiedName, ".");
        ArrayList list = new ArrayList();
        while (st.hasMoreTokens()) {
            String name = st.nextToken().trim();
            if (!name.equals("*"))
                list.add(name);
        }
        return (String[]) list.toArray(new String[list.size()]);
    }

    }
```