Eclipse Form开发指南（1）

[转贴 2005-11-26 14:41:35 | 发表者: redlly]

一、前言

Form 是 Eclipse3.0 后向编程者新增的一种界面，不过之前 Eclipse 一直在用它，只不过当时仅限内部使用，3.0 后才成为公用 API。什么是 Form？请看下面打开插件项目的 plugin.xml 文件后的图示：

在 Eclipse 中 Form 主要用于文件的编辑，象上面的 plugin.xml，其他界面是很少用到 Form 的。所以一般来说 Form 大都建于 View 中，但本文为了便于调式运行，将 Form 的实例写在了 Dialog 中。因为 View 是需要创建一个插件项目，而 Dialog 只需写一个 SWT Application 就行了。

先在这里交待一下本文所用的开发环境：WindowsXP + JDK5.0 + Eclipse3.2M2

二、Hello World 实例

（1）主类的代码如下：

```
import org.eclipse.jface.dialogs.Dialog;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Control;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.ui.forms.widgets.FormToolkit;
import org.eclipse.ui.forms.widgets.ScrolledForm;
/**
 * @author ChenGang 2005-10-26
 */
public class FormDialog1 extends Dialog {
    protected FormDialog1(Shell parentShell) {
        super(parentShell);
    }
    /*
     * (non-Javadoc)
     * @see org.eclipse.jface.dialogs.Dialog#createDialogArea(org.eclipse.swt.widgets.Comp
osite)
```

```
     */
    protected Control createDialogArea(Composite parent) {
        FormToolkit toolkit = new FormToolkit(parent.getDisplay());
        ScrolledForm form = toolkit.createScrolledForm(parent);
        form.setText("Hello World!");
        return form;
    }
}
```

例程说明：

- 首先创建一个 FormToolkit 对象
- 再基于此 FormToolkit 对象创建 form（这里是 ScrolledForm）
- 注意：如果是在插件的 ViewPart 中，则需要在 dispose 方法中将 FormToolkit 对象占用的资源释放，如下所示：

  ```
  public void dispose() {
      toolkit.dispose();
      super.dispose();
  }
  ```

（2）运行上面的 Dialog 类，还需要一个入口程序，如下：

```
import org.eclipse.swt.widgets.Shell;
public class TestFormDialog {
    public static void main(String[] args) {
        Shell shell = new Shell();
        new FormDialog1(shell).open();
    }
}
```

（3）运行 TestFormDialog 后，得到如下界面:

（4）特别说明：

- 以后所有 Dialog 例程，通通都用这个入口程序来运行，你只需要将程序中 FormDialog1 相应的改成其他 Dialog 例程类就行了。
- 在运行此例程之前，你需要先创建一个普通的 SWT 项目。(可参考《Eclipse 从入门到精通》一书)
- Form 需要 plugins/下的 org.eclipse.ui.forms_3.1.0.jar 包的支持，。

三、Hyperlink 控件

现在我们来使用更的 Form 控件，本节将使用 Hyperlink 控件，这在显示一个 WEB 地址时候会要用到的，如下图所示：

其例程如下：

```
import org.eclipse.jface.dialogs.Dialog;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.swt.SWT;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Control;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.ui.forms.events.HyperlinkAdapter;
import org.eclipse.ui.forms.events.HyperlinkEvent;
import org.eclipse.ui.forms.widgets.FormToolkit;
import org.eclipse.ui.forms.widgets.Hyperlink;
import org.eclipse.ui.forms.widgets.ScrolledForm;
/**
 * @author ChenGang 2005-10-26
 */
public class FormDialog2 extends Dialog {
    protected FormDialog2(Shell parentShell) {
        super(parentShell);
    }
    /*
     * (non-Javadoc)
     * @see org.eclipse.jface.dialogs.Dialog#createDialogArea(org.eclipse.swt.widgets.Comp
osite)
     */
    protected Control createDialogArea(Composite parent) {
        FormToolkit toolkit = new FormToolkit(parent.getDisplay());
        ScrolledForm form = toolkit.createScrolledForm(parent);
        form.setText("Hello World!");
        //
        Composite body = form.getBody();
        body.setLayout(new GridLayout());
        Hyperlink link = toolkit.createHyperlink(body, "http://www.chengang.com.cn/", SWT.
WRAP);
        link.addHyperlinkListener(new HyperlinkAdapter() {
```

```
            public void linkActivated(HyperlinkEvent e) {
                MessageDialog.openInformation(null, "", "you had click me.");
            }
        });
        return form;
    }
}
```

例程说明：

- Form 的布局和普通 SWT 控件是一样的，这里用了 GridLayout 布局
- addHyperlinkListener 是 Hyperlink 控件单击响应事件

四、Text、Button、Composite 控件

实例的运行界面如下：

实例的源代码如下：

```
import org.eclipse.jface.dialogs.Dialog;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.swt.SWT;
import org.eclipse.swt.layout.GridData;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.layout.RowLayout;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Control;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.swt.widgets.Text;
import org.eclipse.ui.forms.events.HyperlinkAdapter;
import org.eclipse.ui.forms.events.HyperlinkEvent;
import org.eclipse.ui.forms.widgets.FormToolkit;
import org.eclipse.ui.forms.widgets.Hyperlink;
import org.eclipse.ui.forms.widgets.ScrolledForm;
/**
 * @author ChenGang 2005-10-26
 */
```

```
public class FormDialog3 extends Dialog {
    protected FormDialog3(Shell parentShell) {
        super(parentShell);
    }
    protected Control createDialogArea(Composite parent) {
        FormToolkit toolkit = new FormToolkit(parent.getDisplay());
        ScrolledForm form = toolkit.createScrolledForm(parent);
        form.setText("Hello World!");
        //
        Composite body = form.getBody();
        body.setLayout(new GridLayout());
        Hyperlink link = toolkit.createHyperlink(body, "http://www.chengang.com.cn/", SWT.
WRAP);
        link.addHyperlinkListener(new HyperlinkAdapter() {
            public void linkActivated(HyperlinkEvent e) {
                MessageDialog.openInformation(null, "", "you had click me.");
            }
        });
        //
        Composite infoComp = toolkit.createComposite(body);
        infoComp.setLayoutData(new GridData(GridData.FILL_BOTH));
        infoComp.setLayout(new GridLayout(2, false));
        //create a label
        toolkit.createLabel(infoComp, "姓名:");
        //create a Text
        Text text = toolkit.createText(infoComp, "");
        text.setLayoutData(new GridData(GridData.FILL_HORIZONTAL));
        //create a button
        toolkit.createLabel(infoComp, "爱好:");
        {
            Composite c = toolkit.createComposite(infoComp);
            //把上一句换成new Composite(infoComp, SWT.BORDER);试试^_^
            c.setLayout(new RowLayout());
            toolkit.createButton(c, "读书", SWT.CHECK);
            toolkit.createButton(c, "泡妞", SWT.CHECK);
            toolkit.createButton(c, "足球", SWT.CHECK);
        }
        toolkit.paintBordersFor(infoComp);
        return form;
    }
}
```

例程说明：

- 在 Form 中 Text、Button、Composite 控件的创建方式和寻常不同，都是由 toolkit 来创建
- 布局和普通 SWT 控件一样
- 一定要记得最后一句：toolkit.paintBordersFor(infoComp)；，不信你删除试试

Eclipse Form开发指南（2）

[转贴 2005-11-26 14:42:16 | 发表者: redlly]

Eclipse 中还提供了两个新的布局：TableWrapLayout、ColumnLayout，本文将介绍这两个布局

（1）TableWrapLayout

在 Eclipse3.1.1 中文版的帮助中是这样介绍 TableWrapLayout 的。

> TableWrapLayout 是基于网格的布局，它与 SWT 的通用 GridLayout 非常类似。不同之处在于，它使用工作方式更类似于 HTML 表的布局算法。它尝试保持为其提供的客户区，并垂直地增大以弥补空间不足问题。
>
> 在 GridLayout 与 TableWrapLayout 之间有很多相似点。它们都在网格中组织子代。两者都有用于指示布局如何处理每个控件的布局数据。它们都能接受有关哪个控件应该获得额外空间等的提示。
>
> 但是，它们在布局方式上有着根本的差别。TableWrapLayout 以列开头。它计算每一列的最小、首选和最大宽度并使用此信息来指定额外的空间。它还尝试在各列之间划分空间时保持最佳效果，以便不会使某些控件回绕次数过多。
>
> 可以将 GridLayout 与 TableWrapLayout 混合使用，但是，使用 GridLayout 的分支就是回绕停止分支。如果您不希望进行回绕（如果组合体包含无法以任何方式进行回绕的控件，如文本、按钮或树等），这种效果就相当令人满意。但是，应该存在从表单主体到每个需要进行回绕的文本控件的未中断路径。

也就是说：

- TableWrapLayout 是和 GridLayout 类似的网格型布局
- 与 GridLayout 不同之处在于 TableWrapLayout 更有 HTML 布局方式的特点
- TableWrapLayout 和 GridLayout 可以混用

- 问题：如果将上例中超链接的文本设置的足够长

```
link.setText("This is an example of a form that is much longer and will need to wrap.");
```

即使设置了 SWT.WRAP，文本内容不会自动 WRAP，这是因为体内容的布局是 GridLayout

- Eclipse Form 提供替代的布局 TableWrapLayout：类似于 GridLayout，但是具有象 HTML 表格一样自动 WRAP 功能
- 下面是解决超链接文本自动 WRAP 的例子：

```
public void createPartControl(Composite parent) {
        toolkit = new FormToolkit(parent.getDisplay());
        form = toolkit.createScrolledForm(parent);
        form.setText("Hello, Eclipse Forms");
```

```
                Composite body = form.getBody();
                TableWrapLayout layout = new TableWrapLayout();
                body.setLayout(layout);
                Hyperlink link = toolkit.createHyperlink(body, "Click here.",
SWT.WRAP);
                link.addHyperlinkListener(new HyperlinkAdapter() {
                        public void linkActivated(HyperlinkEvent e) {
                                System.out.println("Link activated!");
                        }
                });

                layout.numColumns = 2;
                link.setText("This is an example of a form that is much longer
and will need to wrap.");
                TableWrapData td = new TableWrapData();
                td.colspan = 2;
                link.setLayoutData(td);
                Label label = toolkit.createLabel(body, "Text field label:");
                Text text = toolkit.createText(body, "");
                td = new TableWrapData(TableWrapData.FILL_GRAB);
                text.setLayoutData(td);
                text.setData(FormToolkit.KEY_DRAW_BORDER,
FormToolkit.TEXT_BORDER);
                Button button = toolkit.createButton(body,
                                "An example of a checkbox in a form",
SWT.CHECK);
                td = new TableWrapData();
                td.colspan = 2;
                button.setLayoutData(td);
                toolkit.paintBordersFor(body);
```

```
        }
```

- 下面是程序变化的地方：
  - TableWrapLayout 替代 GridLayout
  - 使用 TableWrapData 来提供布局数据信息
  - 设置的属性使用 colspan、rowspan 等来源于 HTML 表格单元的属性
- 要注意的是：需要自动 WRAP 的控件，需要设置成 SWT.WRAP 风格

一个悬而未决的问题：

这里 Link 未能向有些资料所提到的自动折行，估计是因为我们将 Form 创建在 dialog 中的原因，如果将其创建在 View 中可能就有自动折行的效果。我没试过。这里 View 和 Dialog 在这里并没有本质的区别，有区别的是 form 所在的容器--parent 所用的布局也许不同。

2、ColumnLayout

在 Eclipse3.1.1 中文版的帮助中是这样介绍 ColumnLayout 的。

> UI 表单中的另一个定制布局是 RowLayout 的变体。如果我们将 RowLayout 配置成垂直地放置子代（放置成列）并使列中的所有控件都相同，就会得到若干列（这取决于控件的宽度），但最后一列通常没有完全填满（这取决于控件的数目）。并且，如果放到表单中，所有控件都将在一列中，这是因为 RowLayout 无法进行“垂直”回绕。如果使用 GridLayout，我们就必须事先选择列数并提供选项。
>
> 某些情况下，有一些更复杂的表单，我们希望列数有适应能力。换而言之，我们希望列数根据表单的宽度进行更改 - 尽可能使用更多的列，当宽度减小时，列数也减小。我们还希望或多或少均匀地填充表单区域(所有列的高度差不多相同)。所有这些目标都可以通过 ColumnLayout 实现。
>
> 与TableWrapLayout相比，ColumnLayout 要简单得多。几乎不需要进行任何配置。唯一需要您选择的选项是所需的列数范围（缺省值是 1 到 3）。

给出一个实例：

```
import  org.eclipse.jface.dialogs.Dialog;
```

```
import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Control;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.ui.forms.widgets.ColumnLayout;
import org.eclipse.ui.forms.widgets.FormToolkit;
import org.eclipse.ui.forms.widgets.ScrolledForm;
/**
 * @author ChenGang 2005-11-03
 */
public class FormDialog1 extends Dialog {
    protected FormDialog1(Shell parentShell) {
        super(parentShell);
    }
    /*
     * (non-Javadoc)
     * @see org.eclipse.jface.dialogs.Dialog#createDialogArea(org.eclipse.swt.widgets.Co
mposite)
     */
    protected Control createDialogArea(Composite parent) {
        FormToolkit toolkit = new FormToolkit(getShell().getDisplay());
        ScrolledForm form = toolkit.createScrolledForm(parent);
```

```
        form.setText("子在川上曰");
        //将 form 设定为 2 列的 ColumnLayout 布局
        Composite body = form.getBody();
        {
            ColumnLayout layout = new ColumnLayout();
            layout.maxNumColumns = 4;
            layout.minNumColumns = 1;
            body.setLayout(layout);
        }
        toolkit.createButton(body, "1", SWT.PUSH);
        toolkit.createButton(body, "2", SWT.PUSH);
        toolkit.createButton(body, "3", SWT.PUSH);
        toolkit.createButton(body, "4", SWT.PUSH);
        toolkit.createButton(body, "5", SWT.PUSH);
        toolkit.createButton(body, "6", SWT.PUSH);
        toolkit.createButton(body, "7", SWT.PUSH);
        toolkit.createButton(body, "8", SWT.PUSH);
        //
        toolkit.paintBordersFor(body);
        return parent;
    }
    /*(non-Javadoc)
```

```
     * @see org.eclipse.jface.window.Window#getShellStyle()
     */
    protected int getShellStyle() {
        return super.getShellStyle() | SWT.RESIZE;
    }
}
```

实例的效果图如下：

从效果图来看，有些资料上提到的“ColumnLayout 的布局方式是从上到下，从左到右”似乎还不够准确，补上这一句：当宽度不够时，则左边控件往折行排列。

Eclipse Form程序设计指南（3）

[转贴 2005-11-26 14:46:29 | 发表者: redlly]

4、复杂控件

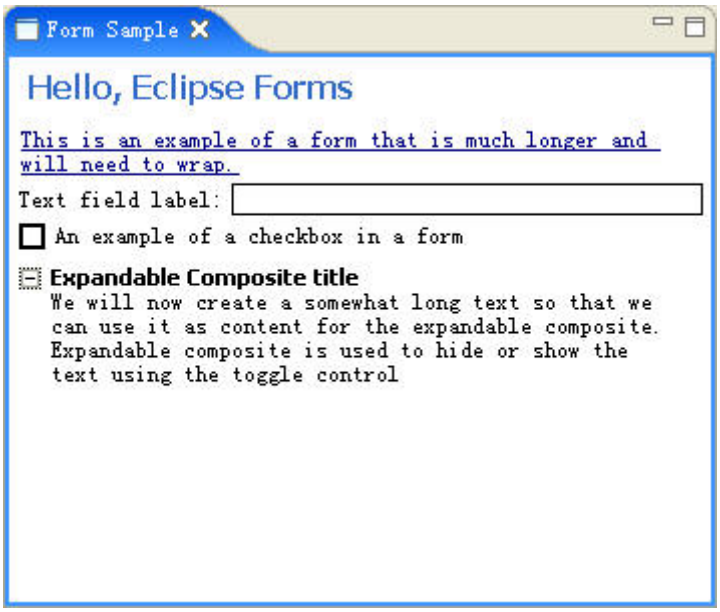
（1） ExpandableComposite

- Web 页面中一个通用的主题是具有收缩一部分页面内容的能力
- Eclipse Form 也提供了这样一个控件：ExpandableComposite
- 下面的代码片断是使用 ExpandableComposite 的一个例子：

```
ExpandableComposite ec =
toolkit.createExpandableComposite(body,
        ExpandableComposite.TREE_NODE
            | ExpandableComposite.CLIENT_INDENT);
ec.setText("Expandable Composite title");
String ctext = "We will now create a somewhat long text so that
"
        + "we can use it as content for the expandable composite.
"
        + "Expandable composite is used to hide or show the text
using the "
        + "toggle control";
Label client = toolkit.createLabel(ec, ctext, SWT.WRAP);
ec.setClient(client);
td = new TableWrapData();
td.colspan = 2;
ec.setLayoutData(td);
ec.addExpansionListener(new ExpansionAdapter() {
        public void expansionStateChanged(ExpansionEvent e) {
            form.reflow(true);
        }
});
```

- 这个控件有很多风格，TREE_NODE 使得该控件具有树型节点的展开、收缩功能；而 TWISTIE 使得控件具有三角箭头风格
- EXPANDED 使得初始展开显示
- CLIENT_INDENT 使得 Client 内容缩进对齐
- ExpandableComposite 呈现为激活控件和标题，而可以展开、收缩的内容称为 Client

- Client 必须是可展开的 composite（上例是 Label 控件）
- 最后需要添加 Expansion 监听器在状态变化时，reflow Form（即根据控件的新的大小重新定位和更新滚动条）
- 下面是上例的运行结果：



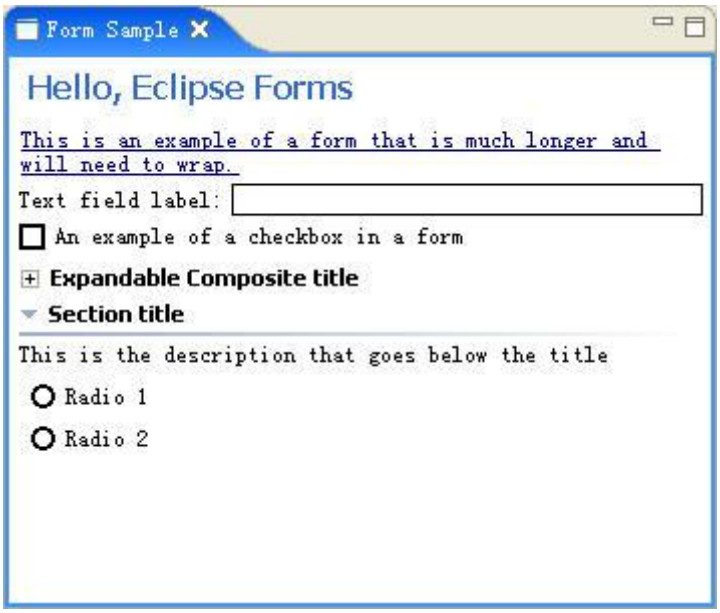
（2）Section

- Eclipse Form 中最常用的定制控件就是 Section（在 PDE 中到处可见）
- Section 扩展 ExpandableComposite，但具有下面的新特性：
  - 在标题下面有一个分隔控件
  - 在分隔控件下面可以有一个描述文本
- 下面的代码片断是使用 Section 的一个例子，代码和 ExpandableComposite 没有太大差别，这里是用了 TWISTIE 风格：

```
                Section section = toolkit.createSection(body,
Section.DESCRIPTION
                                        | Section.TWISTIE | Section.EXPANDED);
                td = new TableWrapData(TableWrapData.FILL);
                td.colspan = 2;
                section.setLayoutData(td);
                section.addExpansionListener(new ExpansionAdapter() {
                        public void expansionStateChanged(ExpansionEvent e) {
                                form.reflow(true);
                        }
                });
```

```
                section.setText("Section title");
                toolkit.createCompositeSeparator(section);
                section
                                .setDescription("This is the description that
goes below the title");
                Composite sectionClient = toolkit.createComposite(section);
                sectionClient.setLayout(new GridLayout());
                button = toolkit.createButton(sectionClient, "Radio 1",
SWT.RADIO);
                button = toolkit.createButton(sectionClient, "Radio 2",
SWT.RADIO);
                section.setClient(sectionClient);
```

- 下面是上例的运行结果：



Eclipse Form程序设计指南（4）

[转贴 2005-11-26 14:47:05 | 发表者: redlly]

5、FromText 控件

（1）概述

- 虽然使用 Label、超链接（或图像链接）以及 TableWrapLayout 布局就能创建丰富的 Form 内容，但是要接近 Web 外观还是很有限的
- Eclipse From 提供了 FromText 控件来创建 Rich 文本，作为上述的补充，有三种形式：
  - 按纯文本呈现
  - 将文本中 URL 转换为超链接呈现
  - 按 XML 标记解析呈现

（2）纯文本

- 下面是按纯文本呈现的例子（等同于 Label）

```
FormText rtext = toolkit.createFormText(body, true);

td = new TableWrapData(TableWrapData.FILL);

td.colspan = 2;

rtext.setLayoutData(td);

String data = "Here is some plain text for the text to render.";

rtext.setText(data, false, false);
```

- FormToolkit 的 createFormText()方法创建 FromText 控件
- FromText 的 setText()方法指定要呈现的文本，后面两个参数指定是否要按 XML 标记进行解析和是否要将文本中的 URL 转换为超链接

（3）URL 转换为超链接

- 下面是将 URL 转换为超链接的例子，和上面例子的唯一区别是 FromText 的 setText()方法的第三个参数设置为 True

```
FormText rtext = toolkit.createFormText(body, true);

td = new TableWrapData(TableWrapData.FILL);

td.colspan = 2;

rtext.setLayoutData(td);

Display display = FormSamplesPlugin.getDefault().getWorkbench().getDisplay();

FormColors formColors = new FormColors(display);

Color activeLinkColor = formColors.createColor("activeLink", 175,225,200);

HyperlinkSettings linkSettings = new HyperlinkSettings(display);

linkSettings.setActiveForeground(activeLinkColor);

linkSettings.setHyperlinkUnderlineMode(HyperlinkSettings.UNDERLINE_HOVER);

rtext.setHyperlinkSettings(linkSettings);
```

```
String data = "Here is some plain text for the text to render; this text is at
http://www.eclipse.org/ web site.";

rtext.setText(data, false, true);

rtext.addHyperlinkListener(new HyperlinkAdapter() {

        public void linkActivated(HyperlinkEvent e) {

                System.out.println("Link active: "+e.getHref());

        }

});
```

- 既然能够将 URL 转换成超链接，FromText 同样提供 addHyperlinkListener()方法来监听超链接事件
- FromText 还提供 setHyperlinkSettings()方法来设置超链接的属性（注意，要在 setText()方法之前设置才有效）
- 超链接的属性由 HyperlinkSettings 对象管理，包括颜色和下划线模式的设置
- 例中使用了 Eclipse From 提供的 FormColors 对象（颜色管理器）来管理颜色，createColor()方法创建一种新的颜色，并提供了 key 值，以便以后可以使用 getColor(key)访问
- 这里是使用 FormColors 的一个简单例子，通常一个插件应该只有一个颜色管理器（FormColors），可以使用 Singlton 模式来访问 FormColors 对象

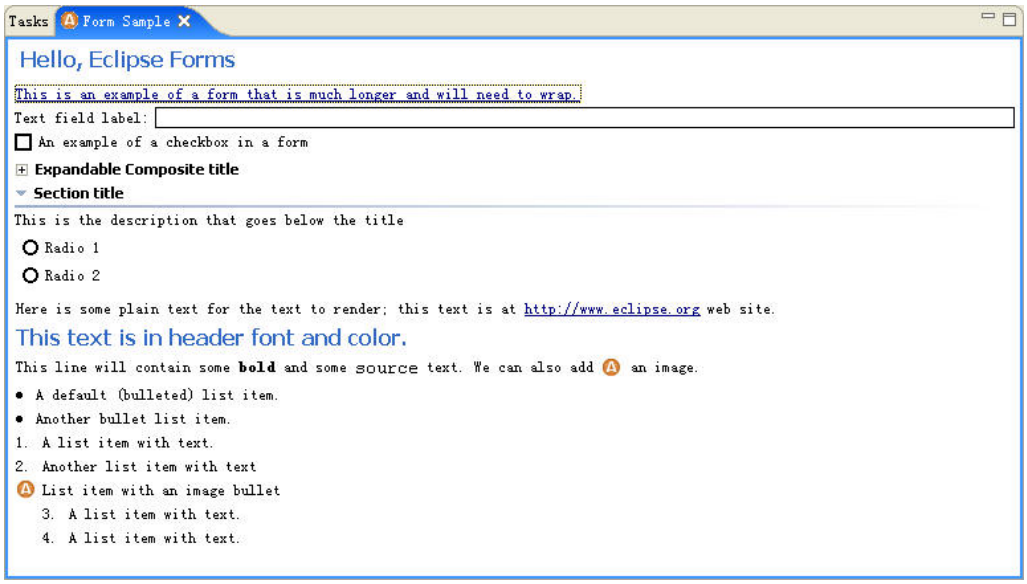
（4）解析 XML 标记

- 解析 XML 标记是 FormText 最强大的用法，但是 FormText 不完全支持所有的标记，下面是 FormText 支持的标记，而且用法有些不同：
  - 根标记必须是<form>
  - <form>可以包含<p>和<li>标记
  - <p>和<li>可以包含普通的文本、
、<b>、、图像（<img>）和超链接（<a>）
  - 标记不允许嵌套
  - <p>有 vspace 属性，表示是否要加垂直空白（缺省是 true）
  - 有下列属性：
    - vspace：同<p>
    - style：bullet（缺省）、text 和 image 值之一
    - value：当 style=bullet 时，无效；当 style=text 时，指定作为 bullet 的文本；当 style=image 时，指定作为 bullet 的图像（key 值）
    - indent：bullet 内容缩进的大小（像素为单位）
    - bindent：bullet 自身缩进的大小（像素为单位）
  - <img>显示图像，其属性 href 指定的是一个 key 值，该值和 FormText 的 setImage()方法中 key 参数指定的值是对应的
  - <a>显示超链接，其属性 href 指定 URL 并通过 FormText 添加监听器来监听超链接的点击事件，<a>还有 nowarp 属性，指定是否允许超链接自动换行
  - <b>：使包含的文本变粗体
  -
：强制换行
  - ：使包含的文本具有特定的颜色（color 属性）和字体（font 属性）这些属性的值也是一个 key 值，和 FormText 的 setColor()、setFont()方法中 key 参数指定的值是对应的
- 下面是一个解析 XML 标记的例子：

```
                StringBuffer buf = new StringBuffer();

                buf.append("<form>");

                buf.append("<p>");

                buf.append("Here is some plain text for the text to render; ");

                buf.append("this text is at <a href=\"http://www.eclipse.org\"
nowrap=\"true\">http://www.eclipse.org</a> web site.");

                buf.append("</p>");

                buf.append("<p>");

                buf.append("<span color=\"header\" font=\"header\">This text is in header font
and color.</span>");

                buf.append("</p>");

                buf.append("<p>This line will contain some <b>bold</b> and some <span
font=\"text\">source</span> text. ");

                buf.append("We can also add <img href=\"image\"/> an image. ");

                buf.append("</p>");

                buf.append("<li>A default (bulleted) list item.</li>");

                buf.append("<li>Another bullet list item.</li>");

                buf.append("<li style=\"text\" value=\"1.\">A list item with text.</li>");

                buf.append("<li style=\"text\" value=\"2.\">Another list item with
text</li>");

                buf.append("<li style=\"image\" value=\"image\">List item with an image
bullet</li>");

                buf.append("<li style=\"text\" bindent=\"20\" indent=\"40\" value=\"3.\">A
list item with text.</li>");

                buf.append("<li style=\"text\" bindent=\"20\" indent=\"40\" value=\"4.\">A
list item with text.</li>");

                buf.append("</form>");

                FormText rtext = toolkit.createFormText(body, true);

                td = new TableWrapData(TableWrapData.FILL);

                td.colspan = 2;

                rtext.setLayoutData(td);

                rtext.setImage("image",
```

```
FormSamplesPlugin.imageDescriptorFromPlugin("FormSamples","images/aspect.gif").createImage(
));

                rtext.setColor("header", toolkit.getColors().getColor(FormColors.TITLE));

                rtext.setFont("header", JFaceResources.getHeaderFont());

                rtext.setFont("text", JFaceResources.getTextFont());

                rtext.setText(buf.toString(), true, false);

                rtext.addHyperlinkListener(new HyperlinkAdapter() {

                        public void linkActivated(HyperlinkEvent e) {

                                System.out.println("Link active: "+e.getHref());

                        }

                });
```

- 下面是上面例子呈现的内容



- 就像上面提到的 FormText 对 XML 标记的支持是有限的，例如：
  - 标记不能嵌套
  - 只支持粗体（<b>），而不支持斜体
  - 文本内容不能被选取，等等
- 所以对于 FormText 的一些限制，需要考虑其它的替代方法：
  - 如果需要具有更为复杂的格式化能力，可以使用 Browser 控件
  - 如果需要具有编辑和字体、颜色风格的能力，可以使用 StyleText 控件
  - 如果需要文本能够自动换行，可以使用具有 SWT.WARP 风格的 SWT Label 控件