IBM

Revitalize your applications with Eclipse Forms

Get a Web-like look without using the embedded browser

Chris Aniszczyk, Software Engineer, IBM

Summary: Eclipse Forms offers a way to enhance the user experience of your Eclipse applications. It offers a "Web-like" look for your applications without using the embedded browser. The goal of this article is to give a brief introduction to Eclipse Forms as a user interface (UI) toolkit and to provide an easy-to-follow example to get you started.

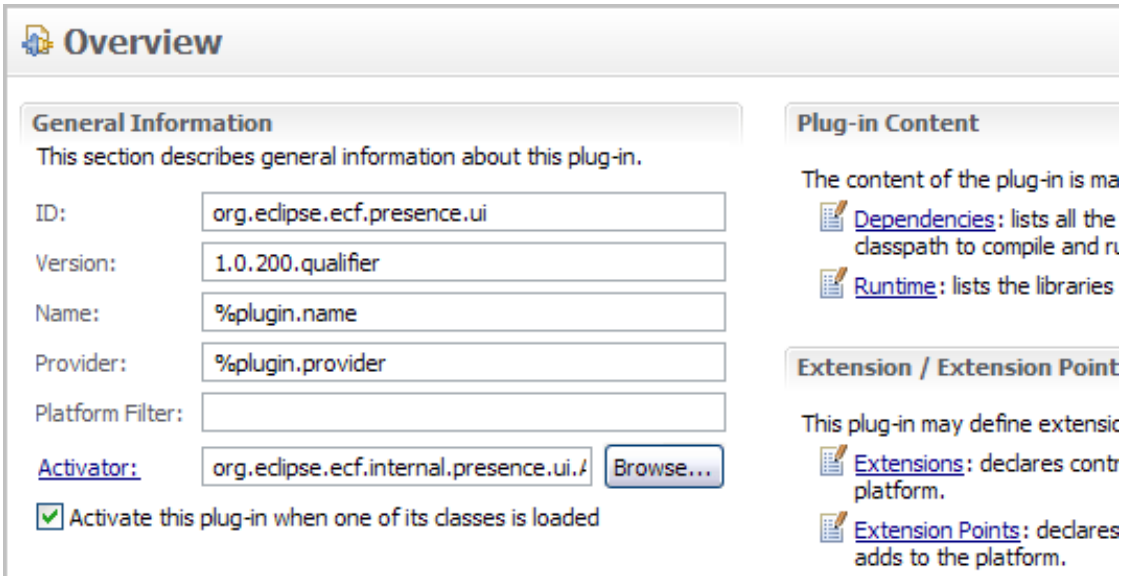
Date: 25 Sep 2007
Level: Intermediate
Activity: 1835 views
Comments: 0 (Add comments)

Average rating

Background

What is Eclipse Forms? Before I answer that question, let me show you a use of Eclipse Forms in the wild. Have you ever built a plug-in in Eclipse? If so, you probably recognize this clean UI.

Figure 1. PDE's manifest editor



This editor is part of Eclipse's Plug-in Development Environment (PDE). It was also the first adopter of Eclipse Forms and is continually one of the exemplary implementations of what Eclipse Forms can do. So now that we have established a visual of what Eclipse Forms looks like, how about we define it?

Eclipse is known for its ability to allow applications built on it to look and feel like native applications. This is possibly due to the Standard Widget Toolkit (SWT), which Eclipse is built on. At its core, SWT offers a portable, lightweight, and *native* widget set that runs on various platforms. Eclipse Forms was originally designed to add a new flavor of user interaction, commonly seen in HTML authoring tools. The Eclipse Forms mission statement is to provide support for creating portable Web-style UIs across all Eclipse projects.

The history of Eclipse Forms

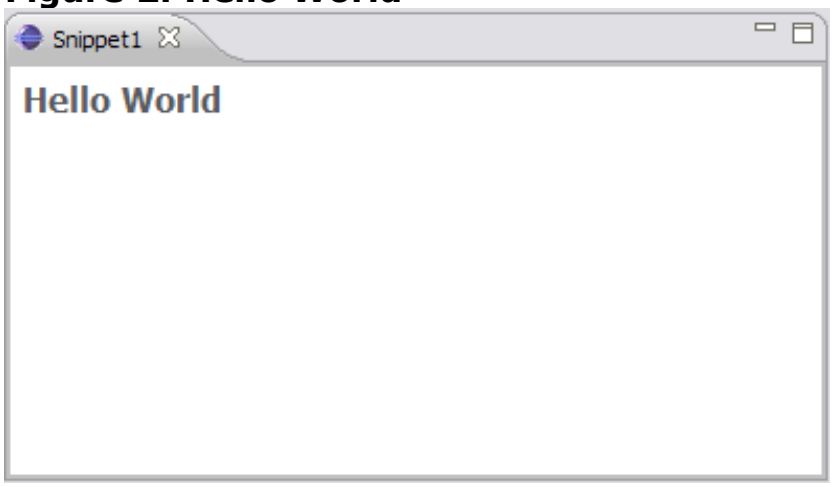
For a more detailed history of Eclipse Forms, I recommend reading "Eclipse Forms: Rich UI for the Rich Client," an introduction by Dejan Glozic (the father of Eclipse Forms), in an article over at the Eclipse Corner.

It's time to begin with your typical Hello World example, followed by a description of the various Eclipse Forms widgets, and we'll conclude with a more concrete real-world example.

Hello, Forms

What better way to start learning about something new than with a simple Hello World-type example. An important thing to understand about Eclipse Forms is that you can literally use its widgets anywhere you can use SWT widgets. In our simple example, we'll use Eclipse Forms within an Eclipse view.

Figure 2. Hello World



Listing 1. Hello World — Eclipse Forms edition (Snippet1.java)

```
public class FormView extends ViewPart {
        private FormToolkit toolkit;
        private ScrolledForm form;

        public FormView() {}

        public void createPartControl(Composite parent) {
                toolkit = new FormToolkit(parent.getDisplay());
                form = toolkit.createScrolledForm(parent);
                form.setText("Hello World");
        }

        public void setFocus() {
                form.setFocus();
        }

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

In Listing 1, we have a typical Eclipse view (that extends `ViewPart`). However, in the `createPartControl(...)` method, we create a new instance of `FormToolkit`. `FormToolkit` is responsible for creating SWT controls and adapting them to work in the Eclipse Forms world. In working with Eclipse Forms, you'll find yourself using this utility class frequently. On the next line, we created a new form (based on `ScrolledForm`) that acts as a container for forms-related content. That's all the magic that's really involved using Eclipse Forms. The code used in the listing above will serve as a template for all your future Eclipse Forms adventures.

Tour de Forms

The next stop on our tour of Eclipse Forms is going over the actual widgets it provides you to work with. We'll discuss each of these widgets and provide a small code snippet demonstrating its use. Finally, we will also discuss how to adapt existing widgets to blend into Eclipse Forms-based applications.

org.eclipse.ui.forms.examples

On top of the sample code we go over in this article, the Eclipse User Assistance team maintains an excellent example Forms plug-in that I found very useful when starting to learn forms. It can be found over in Eclipse's CVS repository: org.eclipse.ui.forms.examples. It contains a plethora of sample code.

Form and ScrolledForm

The first two basic widgets we'll discuss are types of forms: `Form` and `ScrolledForm`. These forms widgets can be considered the foundation of Eclipse Forms and act as containers for content (see Figure 3). They have a title, optional image, an optional title drop-down menu, and some other useful features (see Listing 2).

Figure 3. Form example

Listing 2. Form example (Snippet2.java)

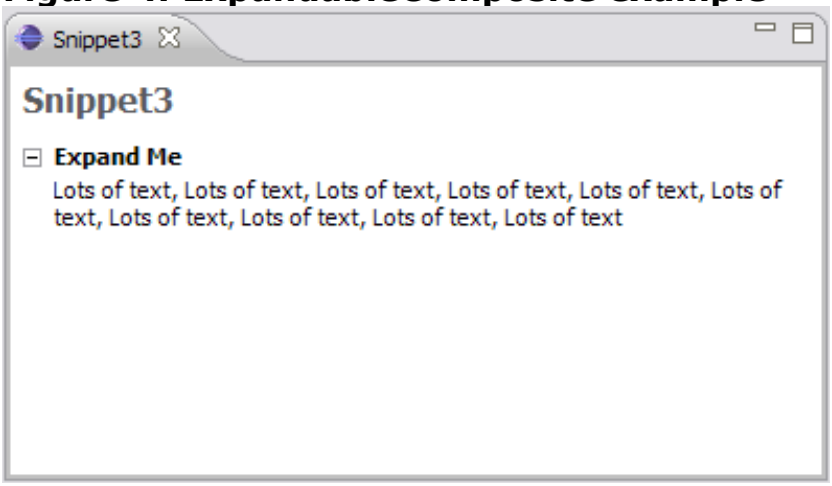
```
...
public void createPartControl(Composite parent) {
        toolkit = new FormToolkit(parent.getDisplay());
        form = toolkit.createForm(parent);
        form.setText("Snippet2");
        TableWrapLayout layout = new TableWrapLayout();
        form.getBody().setLayout(layout);
```

```
        toolkit.createLabel(form.getBody(), "Snippet2");
}
...
```

ExpandableComposite

This widget is a simple composite that allows its child to be shown or hidden (i.e., collapsed or expanded). There are style options for the expansion toggle (twistie or tree) that can be used depending on your needs (see Figure 4 and Listing 3).

Figure 4. ExpandableComposite example



Listing 3. ExpandableComposite example (Snippet3.java)

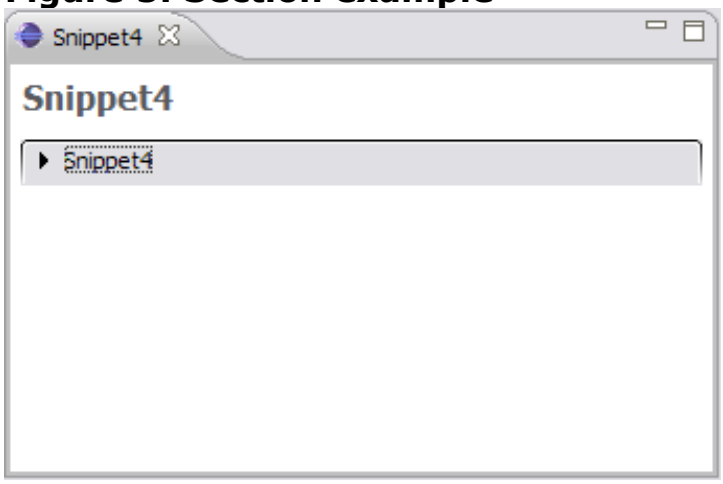
```
...
public void createPartControl(Composite parent) {
        toolkit = new FormToolkit(parent.getDisplay());
        form = toolkit.createScrolledForm(parent);
        form.setText("Snippet3");
        TableWrapLayout layout = new TableWrapLayout();
        form.getBody().setLayout(layout);

        ExpandableComposite composite =
                toolkit.createExpandableComposite(
                        form.getBody(),
                        ExpandableComposite.TREE_NODE|
                        ExpandableComposite.CLIENT_INDENT);
        composite.setText("Expand Me");
        String text = "Lots of text, Lots of text," +
        Lots of text, Lots of text, Lots of text," +
        Lots of text, Lots of text, Lots of text," +
        Lots of text, Lots of text";
        Label label = toolkit.createLabel(composite, text, SWT.WRAP);
        composite.setClient(label);
        TableWrapData td = new TableWrapData();
```

```
        td.colspan = 1;
        composite.setLayoutData(td);
        composite.addExpansionListener(new ExpansionAdapter() {
                public void expansionStateChanged(ExpansionEvent e) {
                        form.reflow(true);
                }
        });
}
...
```

Sections

A *section* is a special type of expandable composite that adds optional description below the title. Sections are used in Eclipse Forms to group information and are probably the most commonly used widget in Eclipse Forms. If you require images or hyperlinks in the description area, you can use a control in the description area.

Figure 5. Section example



Listing 4. Section example (Snippet4.java)

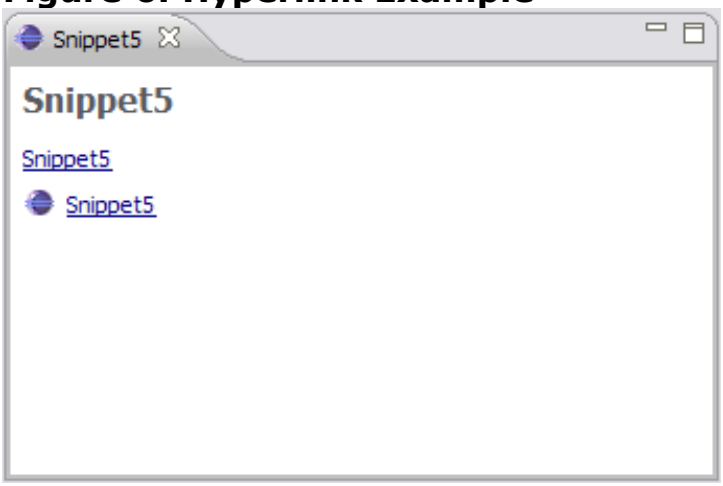
```
...
public void createPartControl(Composite parent) {
        toolkit = new FormToolkit(parent.getDisplay());
        form = toolkit.createScrolledForm(parent);
        form.setText("Snippet4");
        TableWrapLayout layout = new TableWrapLayout();
        form.getBody().setLayout(layout);

        Section section = new Section(form.getBody(), Section.TITLE_BAR | Section.TWISTIE);
        section.setText("Snippet4");
        Label label = toolkit.createLabel(section, "Snippet4", SWT.WRAP);
        section.setClient(label);
        TableWrapData td = new TableWrapData();
        td.colspan = 1;
```

```
        td.grabHorizontal = true;
        section.setLayoutData(td);
}
...
```

Hyperlink and ImageHyperlink

The Hyperlink widget is used to allow actions based on whether a user clicks — or even hovers — over text. If you have ever worked with HTML, this is very similar to the concept of the `<a>` tag. In Eclipse Forms, hyperlinks can be associated with a `HyperlinkGroup` that manages things like normal and active colors.

Figure 6. Hyperlink Example



Listing 5. Hyperlink example (Snippet5.java)

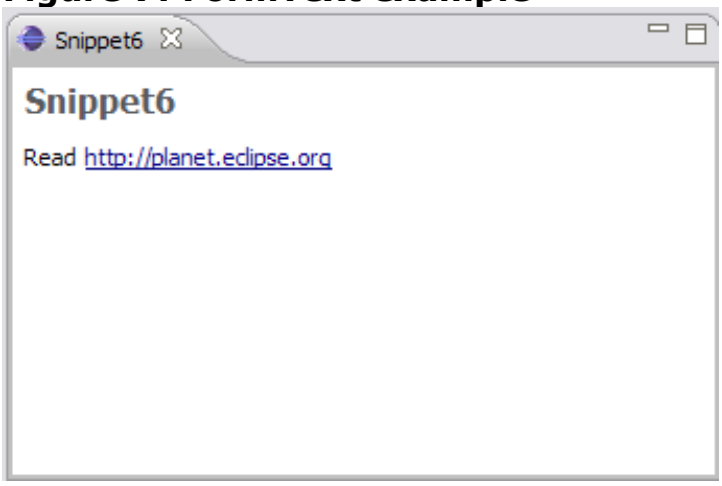
```
...
public void createPartControl(Composite parent) {
        toolkit = new FormToolkit(parent.getDisplay());
        form = toolkit.createScrolledForm(parent);
        form.setText("Snippet5");
        TableWrapLayout layout = new TableWrapLayout();
        form.getBody().setLayout(layout);

        HyperlinkGroup group = new HyperlinkGroup(parent.getDisplay());

        Hyperlink link = new Hyperlink(form.getBody(), SWT.NONE);
        link.setBackground(form.getBackground());
        link.setText("Snippet5");
        link.addHyperlinkListener(new HyperlinkAdapter() {
                public void linkActivated(HyperlinkEvent e) {
                        System.out.println("Snippet5");
                }
        });
```

```
        ImageHyperlink imageLink = new ImageHyperlink(form.getBody(), SWT.NONE);
        imageLink.setBackground(form.getBackground());
        imageLink.setText("Snippet5");
        imageLink.setImage(Activator.getImageDescriptor("icons/sample.gif").createImage());
        imageLink.addHyperlinkListener(new HyperlinkAdapter() {
                public void linkActivated(HyperlinkEvent e) {
                        System.out.println("Snippet5");
                }
        });

        group.add(link);
        group.add(imageLink);
}
...
```

FormText and ScrolledFormText

This widget is a read-only text control capable of rendering wrapped text by parsing specialized XML tags (similar to HTML). Furthermore, it allows for the parsing of links (i.e., http://www.eclipse.org) automatically into hyperlinks if you desire.

Figure 7. FormText example



Listing 6. FormText example (Snippet6.java)

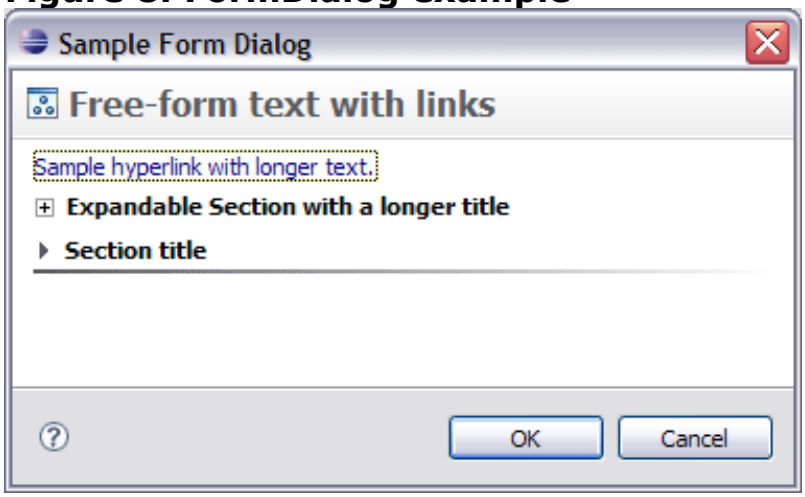
```
...
public void createPartControl(Composite parent) {
        toolkit = new FormToolkit(parent.getDisplay());
        form = toolkit.createScrolledForm(parent);
        form.setText("Snippet6");
        TableWrapLayout layout = new TableWrapLayout();
        form.getBody().setLayout(layout);

        FormText text = toolkit.createFormText(form.getBody(), true);
        text.setText("Read http://planet.eclipse.org", false, true);
```

```
        TableWrapData td = new TableWrapData(TableWrapData.FILL);
        td.colspan = 1;
        text.setLayoutData(td);
}
...
```

FormDialog

This widget is a general-purpose dialog that hosts a form. If you want to use Eclipse Forms within dialogs, this class will help you (see Figure 8 and Listing 7).

Figure 8. FormDialog example



Listing 7. FormDialog example

```
...
        MyFormDialog dialog = new MyFormDialog(shell);
        dialog.create();
        dialog.getShell().setSize(800, 600);
        dialog.getShell
        dialog.open();
...
```

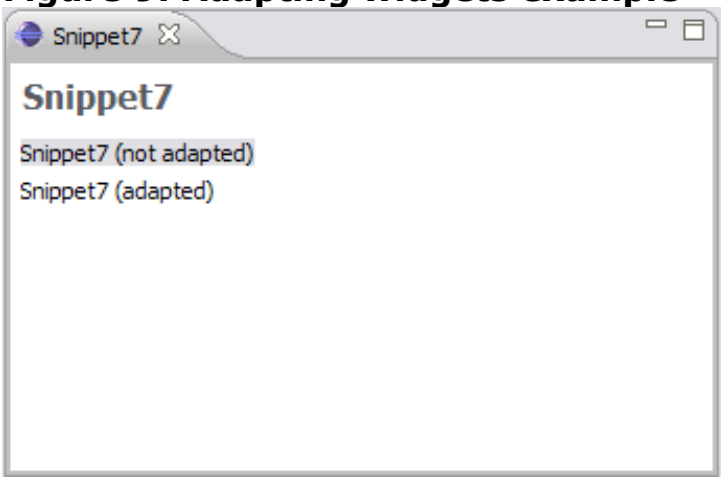
Adapting existing widgets

One of the original design goals of Eclipse Forms was to allow the reuse of existing SWT controls. There's nothing special about the form body in an Eclipse Forms. It's composites all the way down.

To use your own SWT widget (if it isn't easily created using `FormToolkit`), you simply need to *adapt* it. Adapting widgets is done using the `FormToolkit` class (see Figure 9

and Listing 8).

Figure 9. Adapting widgets example



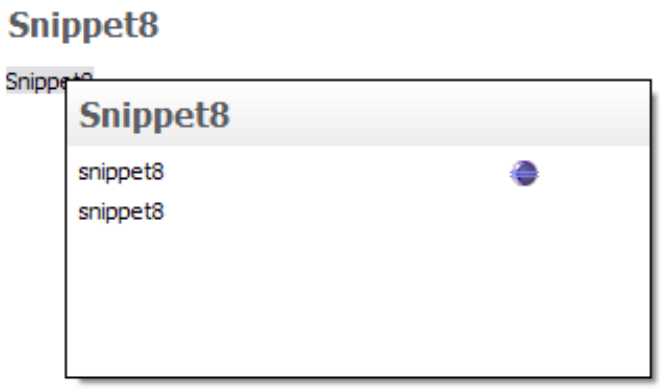
Listing 8. Adapting widgets example (Snippet7.java)

```
...
public void createPartControl(Composite parent) {
        toolkit = new FormToolkit(parent.getDisplay());
        form = toolkit.createScrolledForm(parent);
        form.setText("Snippet7");
        TableWrapLayout layout = new TableWrapLayout();
        form.getBody().setLayout(layout);

        Label label1 = new Label(form.getBody(), SWT.NONE);
        label1.setText("Snippet7 (not adapted)");

        // note, we could've just used toolkit.createLabel(...)
        Label label2 = new Label(form.getBody(), SWT.None);
        label2.setText("Snippet7 (adapted)");
        toolkit.adapt(label2, true, true);
}
...
```

Forms example

For a more real-world example, I'll use one of the Eclipse projects I work on. In the Eclipse Communications Framework (ECF) project, we have a UI to display online contacts. In that UI, we want to support fancy tool tips. What better way to do this then to use Eclipse Forms? Listing 9 demonstrates how to create a custom tool tip using Eclipse Forms.

Figure 10. Forms ToolTip example



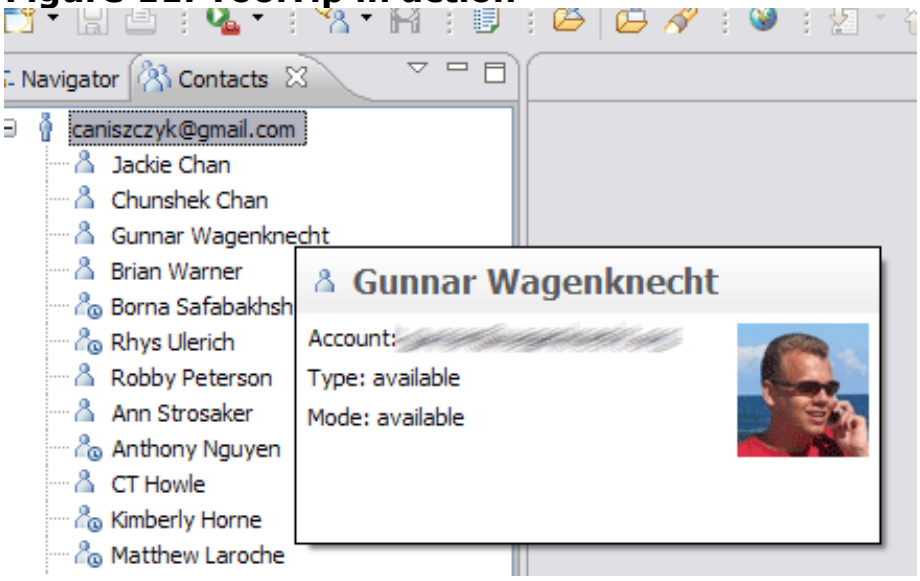
Listing 9. Forms ToolTip example (Snippet8.java)

```
...
public void createPartControl(Composite parent) {
        toolkit = new FormToolkit(parent.getDisplay());
        form = toolkit.createScrolledForm(parent);
        form.setText("Snippet8");
        TableWrapLayout layout = new TableWrapLayout();
        form.getBody().setLayout(layout);

        Label label1 = new Label(form.getBody(), SWT.NONE);
        label1.setText("Snippet8");

        // create a tooltip
        ToolTip tooltip = new MyTooltip(label1);
        tooltip.setPopupDelay(200);
}
...
private class MyTooltip extends ToolTip {

        public MyTooltip(Control control) {
                super(control);
        }

        protected Composite createToolTipContentArea(Event event,
                        Composite parent) {
                FormToolkit toolkit = new FormToolkit(parent.getDisplay());
                FormColors colors = toolkit.getColors();
                Color top = colors.getColor(IFormColors.H_GRADIENT_END);
                Color bot = colors.getColor(IFormColors.H_GRADIENT_START);

                // create the base form
                Form form = toolkit.createForm(parent);
                form.setText("Snippet8");
                form.setTextBackground(new Color[] { top, bot }, new int[] { 100 }, true);
                GridLayout layout = new GridLayout();
                layout.numColumns = 3;
                form.getBody().setLayout(layout);

                // create the text for user information
                FormText text = toolkit.createFormText(form.getBody(), true);
```

```
                GridData td = new GridData();
                td.horizontalSpan = 2;
                td.heightHint = 100;
                td.widthHint = 200;
                text.setLayoutData(td);

                text.setText(
                        "<form><p7gt;snippet8</p><p>snippet8</p></form>",
                        true,
                        false);

                // create the picture representing the user
                td = new GridData();
                td.horizontalSpan = 1;
                td.heightHint = 100;
                td.widthHint = 64;
                FormText formImage =
                        toolkit.createFormText(form.getBody(), false);
                formImage.setText(
                        "<form><p><img href=\"image\"/></p></form>",
                        true, false);
                formImage.setLayoutData(td);

                Image image =
                        Activator.getImageDescriptor("icons/sample.gif").createImage();
                formImage.setImage("image", image);

                return parent;
        }
}
```

Using that example sketched out above, I was able to integrate a Eclipse Forms-based tool tip into ECF fairly easily.

Figure 11. ToolTip in action



Conclusion

The purpose of this article was to provide a brief introduction to Eclipse Forms and some examples. I've accomplished this via easy-to-follow code snippets and a downloadable samples project. It's my hope that you will now be able to pick up Eclipse Forms and put it in your toolbox of things to use when developing an Eclipse application.

A special thanks to Dejan Glozic and the User Assistance team for reviewing the article.

Download

Description	Name	Size	Download method
Sample Perl scripts for this article	os-eclipse-forms.zip	18KB	HTTP

Information about download methods

Resources

Learn

- Glance at the original inspiration for Dejan Glozic's original Eclipse Forms article: Eclipse Forms: Rich UI for the Rich Client.
- Check out the Europa release train.
- Documentation, articles, and downloads of Eclipse are available from Eclipse.org.
- Interested in what's happening inside the Eclipse community? Check out PlanetEclipse.
- Check out the available Eclipse plug-ins at Eclipse Plug-in Central.
- Visit EclipseLive for webinars featuring various Eclipse technologies.
- Want to meet Eclipse committers and learn more about Eclipse projects? Attend EclipseCon, Eclipse's premiere conference.
- Check out the "Recommended Eclipse reading list."

- Browse all the Eclipse content on developerWorks.
- New to Eclipse? Read the developerWorks article "Get started with Eclipse Platform" to learn its origin and architecture, and how to extend Eclipse with plug-ins.
- Expand your Eclipse skills by checking out IBM developerWorks' Eclipse project resources.
- To listen to interesting interviews and discussions for software developers, check out check out developerWorks podcasts.
- For an introduction to the Eclipse platform, see "Getting started with the Eclipse Platform."
- Stay current with developerWorks' Technical events and webcasts.
- Watch and learn about IBM and open source technologies and product functions with the no-cost developerWorks On demand demos.
- Check out upcoming conferences, trade shows, webcasts, and other Events around the world that are of interest to IBM open source developers.
- Visit the developerWorks Open source zone for extensive how-to information, tools, and project updates to help you develop with open source technologies and use them with IBM's products.

Get products and technologies

- Check out the latest Eclipse technology downloads at IBM alphaWorks.
- Download Eclipse Platform and other projects from the Eclipse Foundation.
- Download IBM product evaluation versions, and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- Innovate your next open source development project with IBM trial software, available for download or on DVD.

Discuss

- The Eclipse Platform newsgroups should be your first stop to discuss questions regarding Eclipse. (Clicking on this link will launch your default Usenet news

reader and open the group eclipse.platform.)

- The Eclipse newsgroups has many resources for people interested in using and extending Eclipse.
- Chat with other Eclipse developers and committers on IRC.
- The Eclipse Platform newsgroups should be your first stop to discuss questions regarding Eclipse. (Selecting this will launch your default Usenet news reader application and open eclipse.platform.)
- The Eclipse newsgroups has many resources for people interested in using and extending Eclipse.
- Participate in developerWorks blogs and get involved in the developerWorks community.

About the author

Chris Aniszczyk is an Eclipse committer at IBM Lotus who works on OSGi-related development. His primary focus these days is improving Eclipse's Plug-in Development Environment (PDE) and spreading the Eclipse love inside of IBM's Lotus organization. He is an open source enthusiast at heart, specializing in open source evangelism. He evangelizes about Eclipse in his blog, and he's honored to represent the Eclipse committers on the Eclipse Foundation's board of directors. He's always available to discuss open source and Eclipse over a frosty beverage.