Rich Ajax Platform, Part 1: An introduction

Web 2.0, the Eclipse way

Chris Aniszczyk, Software Engineer, IBM, Software Group
Benjamin Muskalla (bmuskalla@innoopract.com), Software Engineer, Innoopract Informationssysteme GmbH

Summary: Asynchronous JavaScript + XML (Ajax) and the concept of Web 2.0 has spread through the development community as a way add liveliness to Web-based applications. The Rich Ajax Platform (RAP) is a way to build Ajax-enabled Web applications by using the Eclipse development model. This article introduces RAP, tells you how to set up a RAP development environment, shows off some demos, and concludes with some simple-to-understand examples.

Date: 23 Oct 2007
Level: Intermediate
Activity: 2608 views
Comments: 0 (Add comments)

★★★★☆ Average rating

The RAP project aims to enable developers to build Rich Internet Applications by using the Eclipse development model. What does the "Eclipse development model" mean, exactly? Well, RAP allows you developers to build browser-based Ajax applications using full Java™ libraries and Eclipse APIs. It does so by providing a Web-enabled implementation of SWT, JFace, and the Eclipse Workbench. The purpose of this article is to introduce RAP through some simple examples.

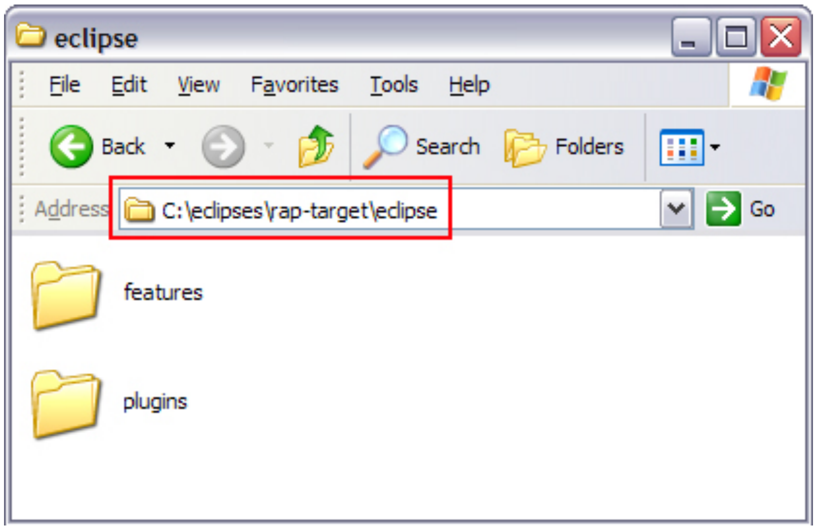
Setup two-step

The setup of RAP is fairly simple. It only requires two steps: downloading RAP and configuring Eclipse to use it.

Step 1: Download RAP

Get RAP from Eclipse.org. We recommend you grab the latest stable milestone. For this article, we used the 1.0 release. Once you download the archive file containing the RAP target platform, unpack it into a directory of your choice (see Figure 1). This directory is important because this will be the directory you will use when setting up your target platform in the next step.

Figure 1. RAP target



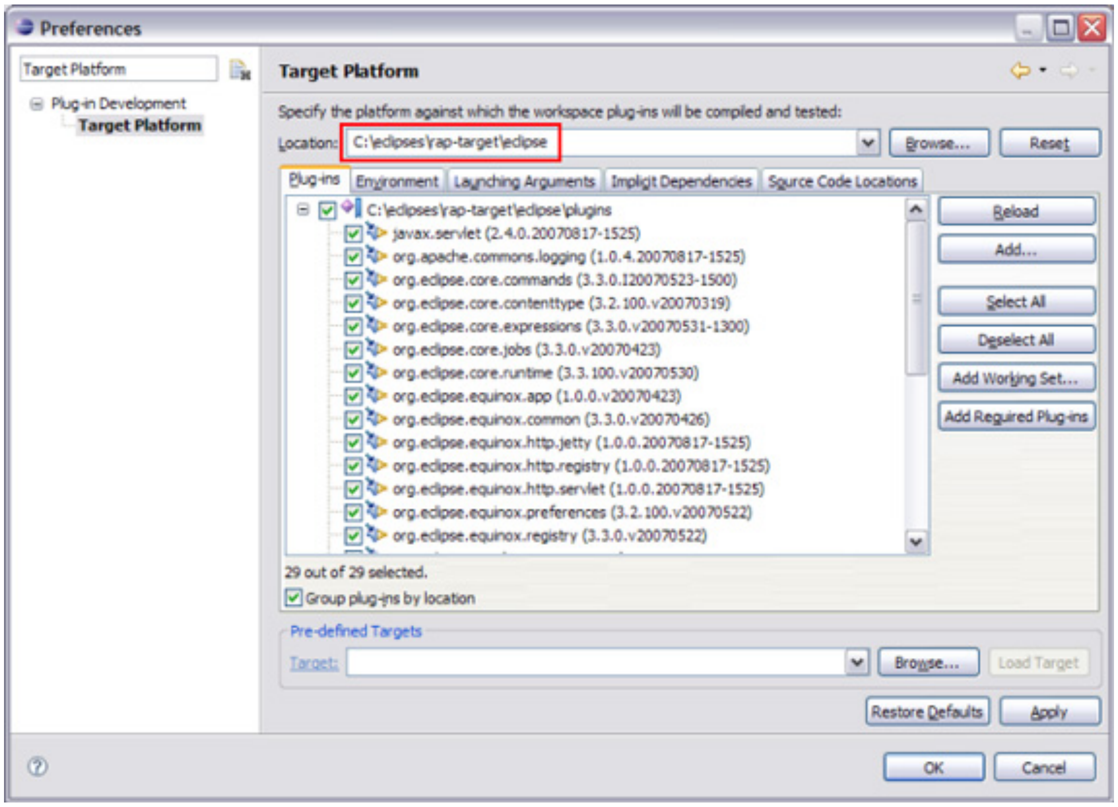
C:\eclipses\rap-target\eclipse

Step 2: Setting the target platform

The Eclipse Plug-in Development Environment (PDE) uses the concept of a target platform. The target platform is simply composed of a set of plug-ins that represents what you're developing against (i.e., targeting). By default, the target platform is set to that of the current running instance of Eclipse. This means that the plug-ins you're currently developing are meant to run on the current running instance. What's neat about the target platform is that it can be changed. For example, you can set the target platform to be against a Eclipse V3.2 installation or even another product's run time. (This neat trick allows you to use the latest Eclipse regardless of what you're developing against.)

In this case, we need to target the RAP platform because that's what we are developing against. To do this, we need to set the target platform preference (**Window > Preferences... > Plug-in Development > Target Platform**) to the directory you unzipped the RAP plug-ins in step 1 (see Figure 2).

Figure 2. Setting RAP as your target platform



Target Platform

Specify the platform against which the workspace plug-ins will be compiled and tested:

Location: C:\eclipses\rap-target\eclipse

- C:\eclipses\rap-target\eclipse\plugins
  - javax.servlet (2.4.0.20070817-1525)
  - org.apache.commons.logging (1.0.4.20070817-1525)
  - org.eclipse.core.commands (3.3.0.I20070523-1500)
  - org.eclipse.core.contenttype (3.2.100.v20070319)
  - org.eclipse.core.expressions (3.3.0.v20070531-1300)
  - org.eclipse.core.jobs (3.3.0.v20070423)
  - org.eclipse.core.runtime (3.3.100.v20070530)
  - org.eclipse.equinox.app (1.0.0.v20070423)
  - org.eclipse.equinox.common (3.3.0.v20070426)
  - org.eclipse.equinox.http.jetty (1.0.0.20070817-1525)
  - org.eclipse.equinox.http.registry (1.0.0.20070817-1525)
  - org.eclipse.equinox.http.servlet (1.0.0.20070817-1525)
  - org.eclipse.equinox.preferences (3.2.100.v20070522)
  - org.eclipse.equinox.registry (3.3.0.v20070522)

29 out of 29 selected.

Group plug-ins by location

Now that you have your target platform set properly, you can develop RAP applications. But before we start looking at code, let's look at a simple demo to see what RAP can do and familiarize ourselves with launching RAP-based applications.

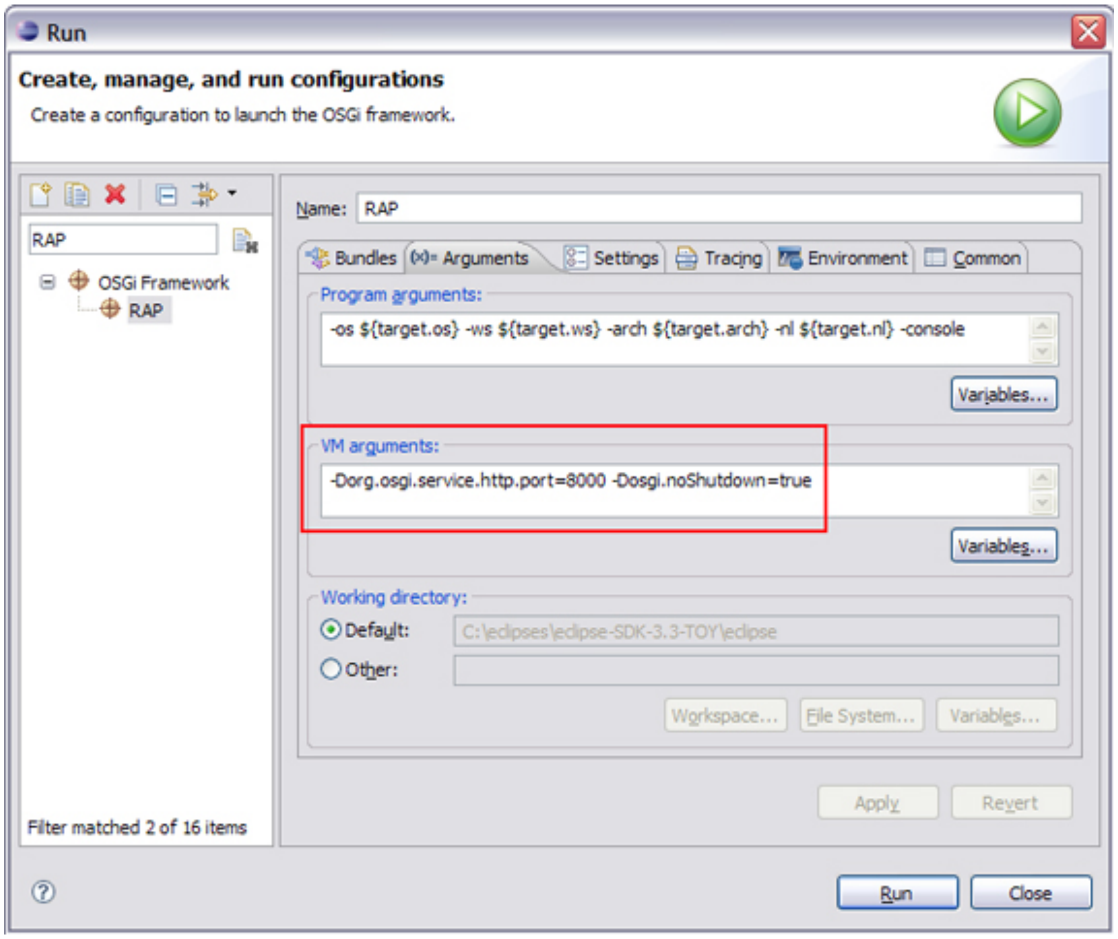
RAP vs. Google Web Toolkit

The Google Web Toolkit (GWT) and RAP are similar in that they both allow you to use Java to code Rich Internet Applications. The big differences are that GWT is running on the client vs. RAP, which mainly running on the server. Since RAP is running on the server, it allows you to access the full Java API and make use of the famous Eclipse plug-in model via OSGi. Another way to think about this is in Eclipse terms: GWT is like a stand-alone SWT application (i.e., just a widget toolkit), where RAP enables an RCP-style approach for Web applications.

Demo

To begin our experience with RAP, we turn to a demo. To interact with RAP, we need to create a launch configuration to launch RAP. To do this, we need to open the Run dialog (**Run > Open Run Dialog...**) and create a new OSGi framework launch configuration. Once this is done, you need to make sure these VM arguments are set: `-Dorg.osgi.service.http.port=8000 -Dosgi.noShutdown=true` (see Figure 3). These arguments enable RAP to launch itself on port 8000 and keep Eclipse from shutting down immediately after it launches.

Figure 3. RAP launch configuration


Run

Create, manage, and run configurations

Create a configuration to launch the OSGi framework.

Name: RAP

Program arguments:

```
-os ${target.os} -ws ${target.ws} -arch ${target.arch} -nl ${target.nl} -console
```

VM arguments:

```
-Dorg.osgi.service.http.port=8000 -Dosgi.noShutdown=true
```

Working directory:

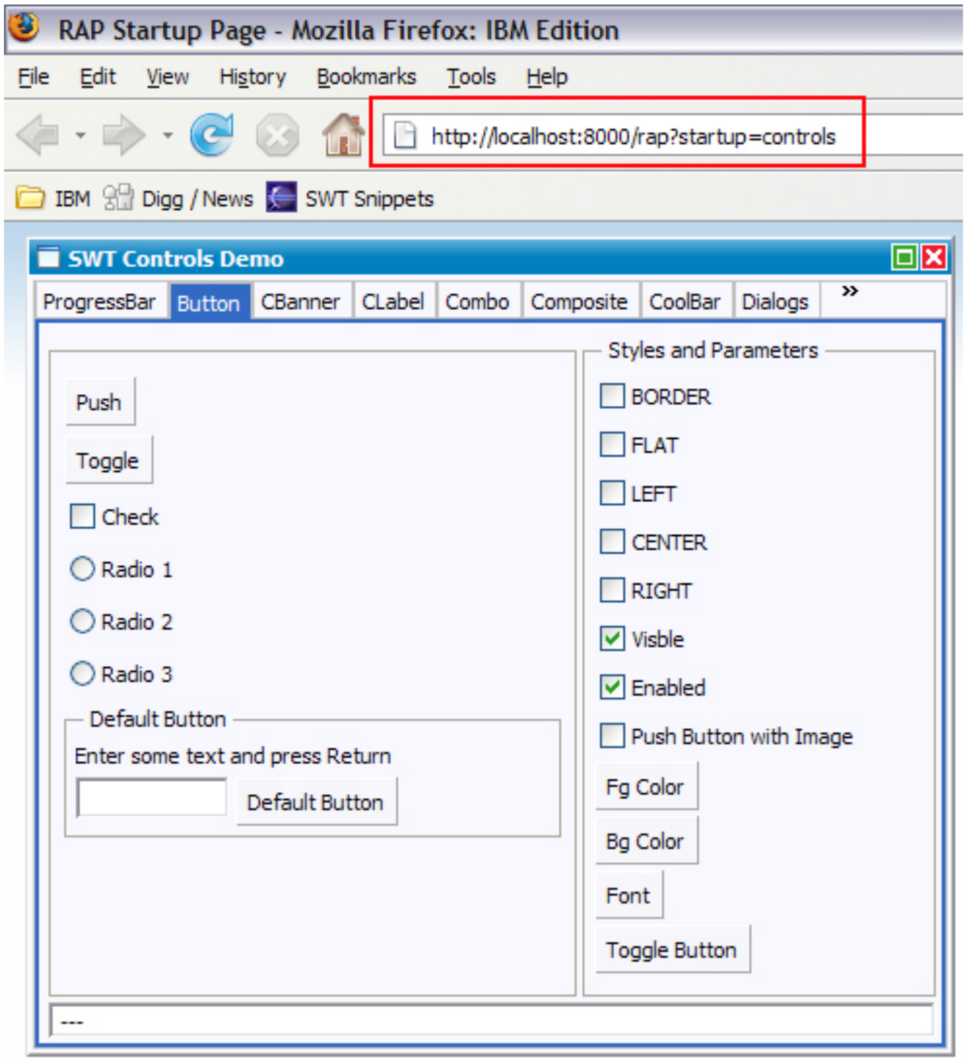
Default: C:\eclipses\eclipse-SDK-3.3-TOY\eclipse

Other:

Filter matched 2 of 16 items

Finally, we can launch a browser and point to `http://localhost:8000/rap?startup=controls` to see the controls demo (see Figure 4). This demo is based on the famous `ControlExample` from the set of SWT Examples.

Figure 4. RAP control demo


RAP Startup Page - Mozilla Firefox: IBM Edition
File Edit View History Bookmarks Tools Help
http://localhost:8000/rap?startup=controls
IBM Digg / News SWT Snippets
SWT Controls Demo
ProgressBar Button CBanner CLabel Combo Composite CoolBar Dialogs
Push
Toggle
Check
Radio 1
Radio 2
Radio 3
Default Button
Enter some text and press Return
Default Button
Styles and Parameters
BORDER
FLAT
LEFT
CENTER
RIGHT
Visble
Enabled
Push Button with Image
Fg Color
Bg Color
Font
Toggle Button

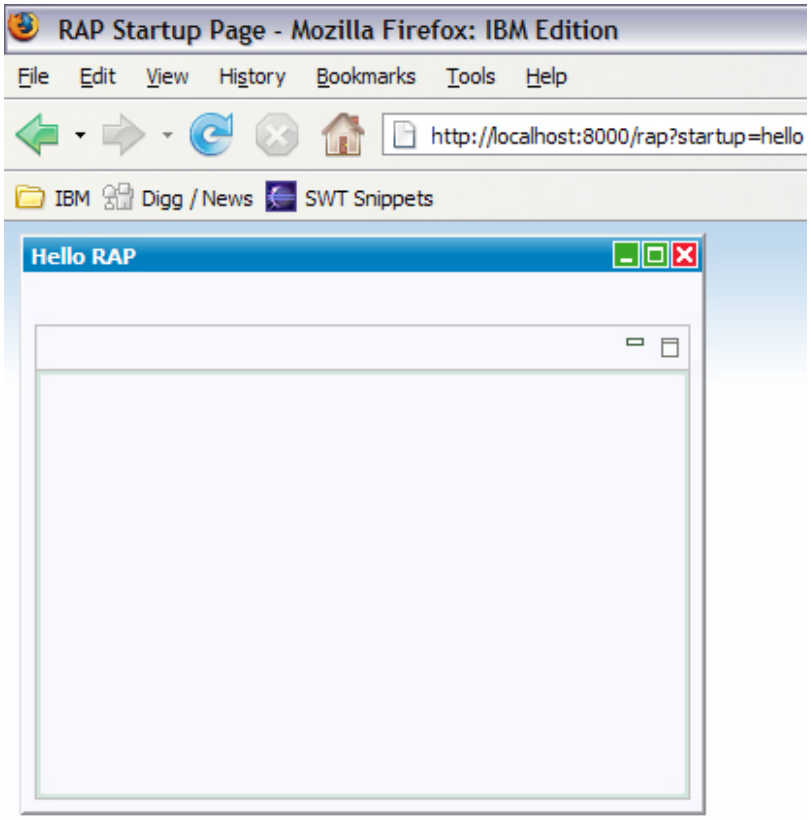
Example applications

We'll go over two examples that are based on some of the Rich Client Platform (RCP) templates provided by PDE.

Hello World example

The traditional programming example is always the Hello World example. We'll start with what it takes to get a simple RAP application running.

Figure 5. Hello World, RAP Edition


RAP Startup Page - Mozilla Firefox: IBM Edition
File Edit View History Bookmarks Tools Help
http://localhost:8000/rap?startup=hello
IBM
Digg / News
SWT Snippets
Hello RAP

The main differences between the RAP version and the RCP version are in the plug-in dependencies and the application entry point. If we look at the plug-in manifest (see Listing 1), we see that we have different dependencies.

Listing 1. RAP Hello World dependencies (MANIFEST.MF)

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Helloworld Plug-in
Bundle-SymbolicName: rap.helloworld; singleton:=true
Bundle-Version: 1.0.0
Bundle-Activator: rap.helloworld.Activator
Require-Bundle: org.eclipse.rap.ui
Eclipse-LazyStart: true
```

Notice the dependency on `org.eclipse.rap.ui`? That's the RAP plug-in that is analogous to the standard `org.eclipse.ui` plug-in from the RCP base. If you're familiar with OSGi and the `Import-Package` header, you don't have to depend on specific plug-ins at all; you can just depend on the packages you need. This is important because you can structure your code in such a way that it works in both RAP and RCP. For example, if we look at code for the workbench advisors in RAP (see Listing 2) and RCP (see Listing 3), we notice similar code and imported packages.

Listing 2. RCP workbench advisor

```
import org.eclipse.swt.graphics.Point;
import org.eclipse.ui.application.ActionBarAdvisor;
import org.eclipse.ui.application.IActionBarConfigurer;
import org.eclipse.ui.application.IWorkbenchWindowConfigurer;
import org.eclipse.ui.application.WorkbenchWindowAdvisor;

public class ApplicationWorkbenchWindowAdvisor extends WorkbenchWindowAdvisor {

    public ApplicationWorkbenchWindowAdvisor(IWorkbenchWindowConfigurer configurer) {
        super(configurer);
    }

    public ActionBarAdvisor createActionBarAdvisor(IActionBarConfigurer configurer) {
        return new ApplicationActionBarAdvisor(configurer);
    }

    public void preWindowOpen() {
        IWorkbenchWindowConfigurer configurer = getWindowConfigurer();
        configurer.setInitialSize(new Point(400, 300));
        configurer.setShowCoolBar(false);
        configurer.setShowStatusLine(false);
        configurer.setTitle("Hello RCP");
    }
}
```

Listing 3. RAP workbench advisor

```
import org.eclipse.swt.graphics.Point;
import org.eclipse.ui.application.ActionBarAdvisor;
import org.eclipse.ui.application.IActionBarConfigurer;
import org.eclipse.ui.application.IWorkbenchWindowConfigurer;
import org.eclipse.ui.application.WorkbenchWindowAdvisor;

public class ApplicationWorkbenchWindowAdvisor extends WorkbenchWindowAdvisor {

    public ApplicationWorkbenchWindowAdvisor(IWorkbenchWindowConfigurer configurer) {
        super(configurer);
    }

    public ActionBarAdvisor createActionBarAdvisor(IActionBarConfigurer configurer) {
        return new ApplicationActionBarAdvisor(configurer);
    }

    public void preWindowOpen() {
        IWorkbenchWindowConfigurer configurer = getWindowConfigurer();
        configurer.setInitialSize(new Point(400, 300));
        configurer.setShowCoolBar(false);
        configurer.setShowStatusLine(false);
        configurer.setTitle("Hello RAP");
    }
}
```

The other major difference between RAP and RCP applications is the entry point (similar to the `main(String[] args)` method in the Java language). In RCP, we have the `org.eclipse.core.runtime.applications` extension point where you define the `IApplication` (note, previous to Eclipse V3.3, this was known as a `IPlatformRunnable`). RAP's equivalent of RCP's application is the `org.eclipse.rap.ui.entrypoint` extension point, which defines an `IEntryPoint`. If we look at the typical entry point code in an RCP context (see Listing 4) vs. the RAP context (see Listing 5), you'll notice that there are similarities. In both cases, we are creating a display and workbench to run our workbench advisor.

Listing 4. RCP application entry point (org.eclipse.core.runtime.applications)

```
public class Application implements IApplication {

        public Object start(IApplicationContext context) throws Exception {
                Display display = PlatformUI.createDisplay();
                try {
                        int returnCode = PlatformUI.createAndRunWorkbench(display,
                        new ApplicationWorkbenchAdvisor());
                        if (returnCode == PlatformUI.RETURN_RESTART)
                                return IApplication.EXIT_RESTART;
                        else
                                return IApplication.EXIT_OK;
                } finally {
                        display.dispose();
                }

        }

        public void stop() {
                final IWorkbench workbench = PlatformUI.getWorkbench();
                if (workbench == null)
                        return;
                final Display display = workbench.getDisplay();
                display.syncExec(new Runnable() {
                        public void run() {
                                if (!display.isDisposed())
                                        workbench.close();
                        }
                });
}
```

Listing 5. RAP application entry point (org.eclipse.rap.ui.entrypoint)

```
public class Application implements IEntryPoint {


        public Display createUI() {

                Display display = PlatformUI.createDisplay();

                PlatformUI.createAndRunWorkbench( display, /

new ApplicationWorkbenchAdvisor() );
```

```
                return display;

        }
}
```

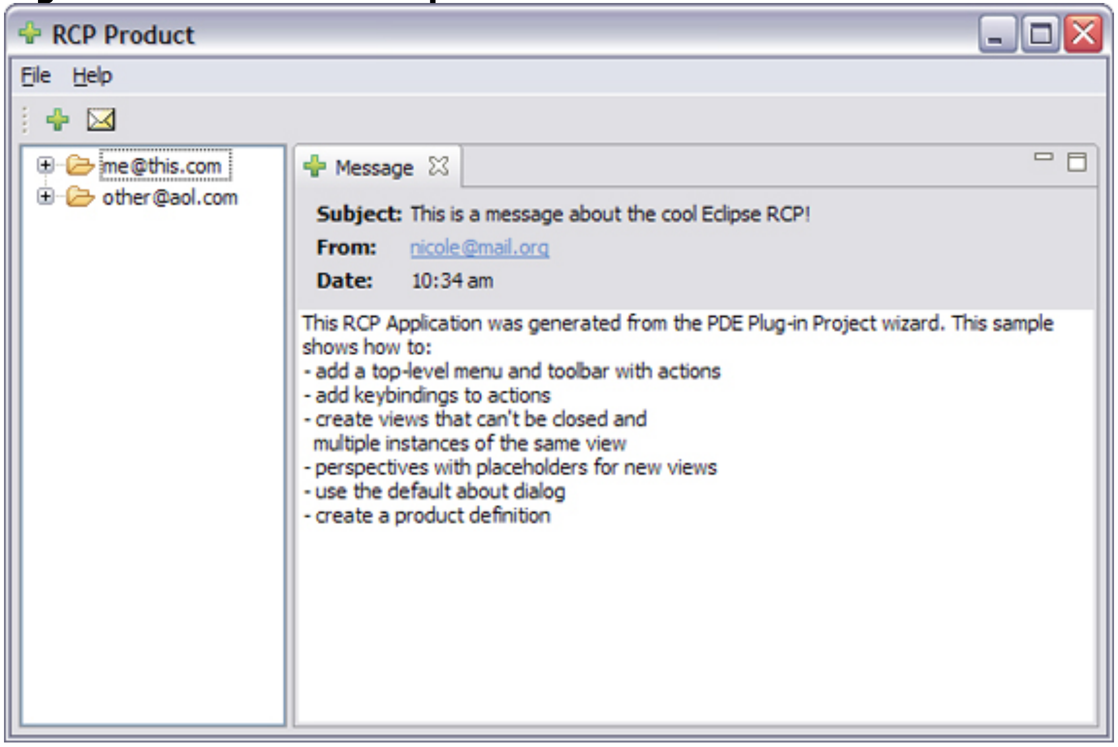
RCP Mail example

The famous RCP Mail example (see Figure 6) is able to run within a RAP environment (see Figure 7) with minimal changes. This example should hint that workbench concepts like views, perspectives, editors (among other things) are available for use within RAP (see Table 1 for a mapping of concepts). The source code for the mail example and the previous example are available for you to tinker with.

Table 1. RAP vs. RCP

RCP	**RAP**
OSGi (server-side)	OSGi
Standard Widget Toolkit (SWT)	RAP Widget Toolkit (RWT)
JFace	JFace
Workbench	Web Workbench

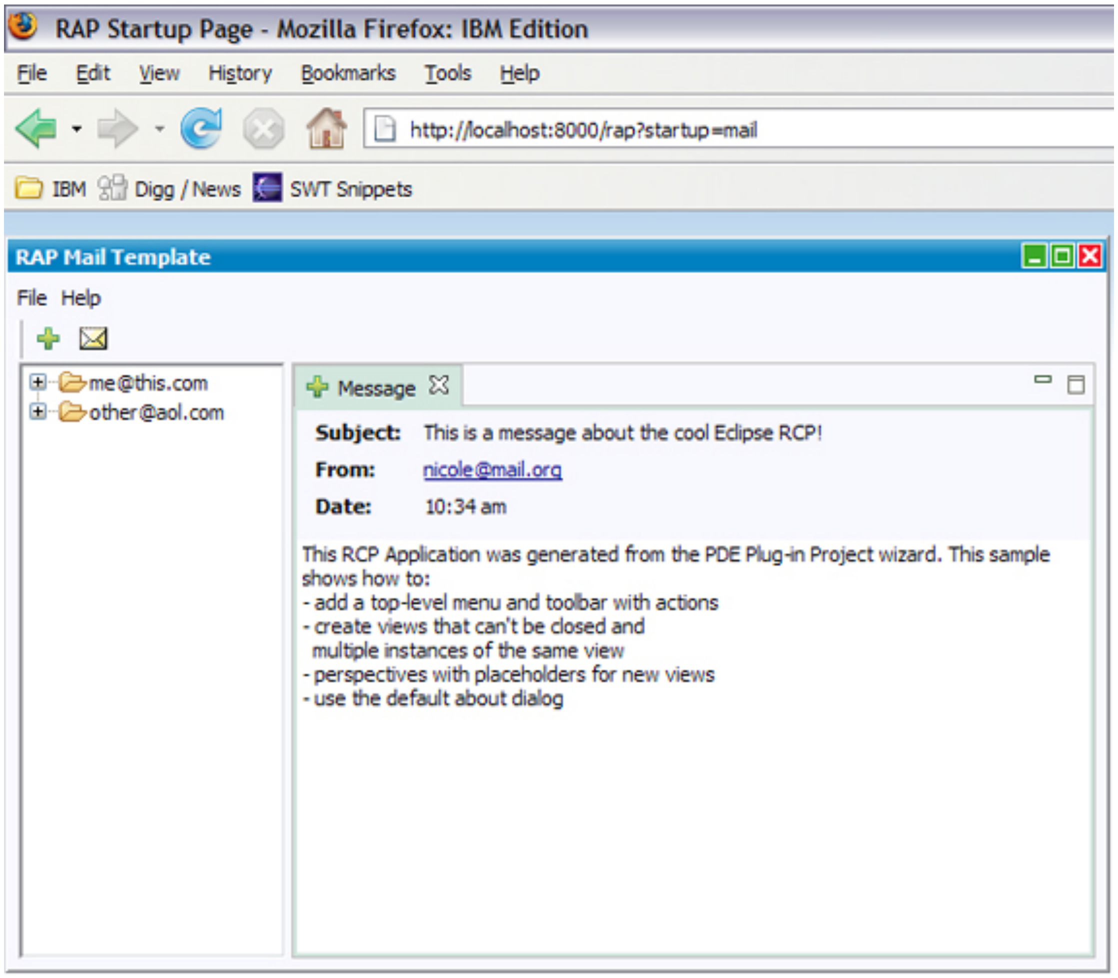
Figure 6. RCP Mail Example



Subject: This is a message about the cool Eclipse RCP!
From: nicole@mail.org
Date: 10:34 am

This RCP Application was generated from the PDE Plug-in Project wizard. This sample shows how to:
- add a top-level menu and toolbar with actions
- add keybindings to actions
- create views that can't be closed and
  multiple instances of the same view
- perspectives with placeholders for new views
- use the default about dialog
- create a product definition

Figure 7. RAP Mail Example


RAP Startup Page - Mozilla Firefox: IBM Edition
File Edit View History Bookmarks Tools Help
http://localhost:8000/rap?startup=mail
IBM Digg / News SWT Snippets
RAP Mail Template
File Help
me@this.com
other@aol.com
Message

Subject: This is a message about the cool Eclipse RCP!
From: nicole@mail.org
Date: 10:34 am

This RCP Application was generated from the PDE Plug-in Project wizard. This sample shows how to:
- add a top-level menu and toolbar with actions
- create views that can't be closed and
  multiple instances of the same view
- perspectives with placeholders for new views
- use the default about dialog

Conclusion

This article introduced RAP via a couple simple examples and a demo. RAP allows Eclipse you to reuse your existing skills and create Rich Internet Applications. RAP will also enable you to structure your code in such a way that it can be reused across the desktop (RCP) and the browser (RAP). In essence, RAP now brings Eclipse to the browser and Web 2.0 party.

Download

Description	**Name**	**Size**	**Download method**
Sample code	os-eclipse-richajax.zip	52KB	HTTP

Information about download methods

Resources

Learn

- Learn more about the Rich Ajax Platform (RAP) by checking it out at Eclipse.org.
- Check out the RAP wiki to learn about new features and advanced use cases.
- 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 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. (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 authors

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.

Benjamin Muskalla works as a software developer and consultant at Innoopract Informationssysteme in Karlsruhe. He is a committer on the Rich Ajax Platform (RAP) project, mostly responsible for the workbench implementation. He is also an active contributor to the Eclipse Platform.