

# RCP Development Experience Sharing

Zhuo Zhang<sup>(1)</sup>, Qiang Song<sup>(1)</sup>, GaoZhong Liang<sup>(1)</sup>  
*IBM China Software Development Lab, 8/F, Building A, Power Creative,  
No.1, East Road, Shang Di, Beijing 100085, P.R. China  
{zhuoz, songq, lianggz}@cn.ibm.com*

## Introduction

We are from IBM China software development laboratory and now working on a DB2 client side product based on Eclipse RCP. We have more than three years RCP related experiences. As a client side tool for DB2, the product has the following main characteristics:

- client-server architecture
- multiple user thread support
- graphical navigator/editor for models

During the development processes, we have gained rich RCP related experiences; also we found some points need to be improved in Eclipse. The following sections describe our experiences, problems and proposals

## Experiences

We have more than 3 years Eclipse plug-ins/RCP related experiences, covering the following aspects:

- **Class loader:** There are tens of plug-ins in our product, before Eclipse3.2 it was hard to both keep the component independence and dynamically load classes from other components. For example, we have component *A* in one plug-in *PA* and component *B* in another plug-in *PB*. Component *A* is needed for component *B* when compiling. So *PA* is declared as a required plug-in for *PB*. Now in component *A* we need to dynamically load a class in component *B* by class loader, before 3.2 we have to build a customized class loader to realize it. Fortunately in Eclipse 3.2 we can fulfill this requirement easily.
- **Job related:** In our product, user needs to schedule multiple user threads at the same time. Eclipse has provided a set of convenient APIs to support job related requirements: start a job, cancel a job, define rules for a job and view the status of a job. With them we have fulfilled basic requirements related to jobs. At the same time, we extend the progress dialog to show the elapsed time of a job.
- **SWT/JFace:** Based on user requirements, we have built lots of customized controls in our product, such as wizard with step-tree, calendar dialog, etc. Also we propose a validation framework for preference/wizard/dialog page.
- **Workbench related:** Views and perspectives are widely used in our product. Also our product provides a customized resource navigator. An SQL editor with syntax highlighting, content assists and vertical rulers is also available in our product.
- **GEF:** It's very important for us to provide graphical navigator/editor for some models. GEF is a good framework to realize it. And also, in order to provide a high-quality graphical presentation of model for end users, some additional technologies are used to compensate GEF/Draw2D's insufficiency.
- **Web application:** We need to embed web application into views in our product; the internal tomcat server is used to realize this requirement.
- **Intro parts:** Eclipse provided a strong user assistance support including Intro part, help system and cheat-sheet. Intro part is a "quick start" for beginning users; cheat-sheet focuses on guiding user to complete a task step by step; help system provides the full documentations related to a product. In our product we have used all of them to help users in different scenarios.
- **Product release related:** Our product need to have splash screen, about dialog, etc. And it is fully

integrated with install-shield to build a commercial product.

- **JVM related:** A special requirement for our product is using existing JVM other than starting up a new one. In some cases the product needs to be invoked from another java application, the product is required to share the same JVM with the invoking java application. Our solution is using the class `org.eclipse.core.launcher.Main` in `startup.jar` file to start up the product.

## Problems/Proposals

We found some enhancements are still needed for several aspects:

- **Class loader:** Due to some license issue, we can not bundle some third-party jar files with our commercial product. Eclipse should support to access these external files at runtime, for example, by referring to an environment variable. The so-called 'external' keyword doesn't work in our product.
- **Job related:** In our application, besides the existing functions, user needs to perform some more complex operations on jobs:
  - View/maintain jobs by group;
  - Suspend/resume/restart a selected job;
  - Change the priorities of running/waiting jobs;Although in low level some APIs are available to support these requirements, in GUI level it's hard to extend the existing implementation (the progress dialog, progress view and job API). It will be very useful if job related API can become more flexible and extensible.
- **Customization of existing plug-ins:** Let's begin with an example. In an application we want to use the progress view defined in plug-in `org.eclipse.ui.ide`, however we do not want to bring other menu items/tool items/views defined in the plug-in into our product. Now the only way we found is to modify the `plugin.xml` of the plug-in to hide them. If there is a way that we can declare what we are interested in an existing plug-in, the customization ability of Eclipse will be enhanced.
- **SWT/JFace:** Missing some common controls for products, such as calendar, common error dialog, cell editor with both editable text and dialog function, etc. There is a project named "Nebula Project" related to this issue. However it does not have an explicit target for these common controls. Also we hope Eclipse could provide an integrated framework to validate the values in a preference/wizard/dialog page. `FieldEditor` can be used for this purpose, but it is only available for preference page.
- **Graphical navigator/editor:** SWT does a nice job in building responsive user interface, but the graphical capabilities of SWT are kept to a minimum. While toolkits like GEF/Draw2D improve the raw features of SWT, which provide a layout and rendering toolkit for displaying graphics and enable us to create a rich graphical editor from an existing application model, however when we want to get the high-quality graphics, just like Java2D does for us, sometime, the current provided toolkits are somewhat weak. They do not increase its intrinsic graphical capabilities like:
  - Antialiasing
  - Transparency
  - True vector representation of shapes (i.e. arbitrary precision of coordinates)
  - User defined transformations for scaling, rotation, etc
  - Separation of the geometry representation and the rendering process
  - Image operations and large image handling
  - SVG support

We really hope those aspects mentioned above can be enhanced in the future.

- **Autonomic Update:** Eclipse has provided a very good framework or component for us to create and publish bundles of plug-ins (called features) to an update site so that users can download and install them directly into Eclipse using the Eclipse update manager. However, to be applied in the enterprise context, we hope, a more flexible and autonomic one should be provided. The reason is: As the client of a complex enterprise system, the components installed on the client are usually different for different users that play different roles in the system. For example, in a role-based access controls (RBAC) system, the features that are to be added, updated or removed in the client-side are usually decided by customization information, for example, who is using this client, what type of role

the user plays in the system, what privileges are permitted to users in that role. So, to fulfill such requirement, a role-based customization mechanism should be provided by the update manager API.