Open Source Integrated Development Environment

Basic Overview

- What is Eclipse?
- Who is behind the Eclipse Project?
- How do I get Eclipse?
- What should I know about getting started?
- Addressing dependencies...

Resources

- http://eclipse.org
- http://eclipse.org/downloads/index.php
- http://sourceforge.net/projects/eclipsesql
- http://jboss.org
- http://www.eclipse.org/webtools/initial-contribution/IBM/Getting%20Started.html

How To

- Download and install
- Create a Java Project
- Install a plug-in using “Find & Install”
- CVS Projects

Sharing code with the Eclipse Platform

PDF e-mail it!

How Eclipse works with source code version control

Level: Intermediate

Pawel Leszek (pawel.leszek@ipgate.pl)
Independent consultant
13 Mar 2003

This article offers an overview of how the Eclipse Platform supports source code version control in software projects. We'll begin with a brief discussion about the ideas for team code development, and then see how Eclipse works with CVS code repositories. We'll also look at some of the source code management software tools that are supported through Eclipse plug-in extensions.

Sharing source code within a team project

Most of today's applications are developed by a team of people. Even small projects that involve only a few developers require tight control over changes in source code. This is a task for source code management software. Two core features must be supported by source code version control software:

- A way to coordinate and integrate changes into the source code
- A history of the work submitted by the team

As team members produce new work, they share this work by committing those changes to the repository. Similarly, when they wish to get the latest available work, they update their local workspaces to the changes on the repository. This means the project repository is constantly changing as team members submit new work. In the other words, the repository should represent the current state of the project. At any point, team members can update their workspaces from the repository and know they are up to date.

Maintaining history is also important so that you can compare the current work against previous versions, and if needed, revert to the older work. Coordinating the team's work so that there is only one definition of the current project state, and containing the integrated work of the team are also essential to managing version control. This coordination is probably the hardest goal to achieve.

An optimal model is one where any member of the team can make changes to any resource he has access to. Because two team members can commit changes to the same resource, conflicts can occur and must be dealt with. This model assumes that conflicts are rather unique. Unfortunately, no source code exists in isolation; it typically contains implicit or explicit dependencies on other resources. Source code has references to artifacts described in other source code resources. And this is the point where source code management software ends its work, because it isn't a substitute for project management. Project managers must do their work: coordinating others' work, and looking after schedules, project phases, and release dates. Further, source code management is not a substitute for developer communication.