Team Development Concepts

In This Chapter

- Before We Begin...
  - Servoy Solutions and Eclipse Projects
  - The Resource Project
- Team Development Concepts
  - Code Repository
  - Version-Control
  - Asynchronous
  - Common Terminology

Before We Begin...

Before we discuss team development concepts, there are some basic Servoy Concepts that we should review.

Servoy Solutions and Eclipse Projects

Solutions and projects are two terms for what is essentially the same thing in Servoy Developer. A Servoy solution (this includes modules, as they are solutions as well) will equate to a Eclipse project. As most of the team providers and repositories we will discuss here were designed to work within Eclipse, most of the terminology will relate to projects.

An important fact to remember is that a project in Eclipse is stored in a single folder in the workspace, and all the information (meaning design and code of the actual solution) about the project is stored in text files (in the case of Servoy, JSON files and Javascript files). That means that even if we are using a graphical editor (i.e. the Form Designer), the actual code is stored in a text file (in this case, a JSON file). Team sharing systems simply track changes in text files between remote (the developer) locations and the central (the repository) locations.

The Resource Project

While a corresponding project folder exists to store all the resources contained in a single solution, there are other resources which may be shared by multiple solutions, such as security settings, i18n (internationalization) settings, CSS style sheets, etc. These resources are also stored in an Eclipse project folder, called the *resource project*. Every Servoy Solution uses a single resource project to provide these shared resources; however, a single resource project may be used by many solutions. A single workspace may contain multiple resource projects, although this is uncommon and not recommended.

Team Development Concepts

Modern team development technologies all share some basic concepts. The key point is that activities that are normally simple and inconsequential for a single developer have impact and need management in a team development environment.

Code Repository

A repository is a centralized storage mechanism for all source code and other project resources. The storage system is often based on file system or database storage.

Version-Control

A repository should provide version control, such that revisions of existing code are non-destructive and a history of all revisions is maintained, including the developer that made the revision and the timestamp, as well as optional comments. Some technologies have mechanisms to even control versions on a more granular level, as well as mechanisms to identify specific versions readily and easily (see tags and branches).

Asynchronous

Team development supports distributed, asynchronous development, such that developers may work on their own local, offline copies of resources and at different times than other developers in the team. Developers may then synchronize their environment with the repository, submitting any revisions to the repository, as well as retrieving from the repository, new revisions created by other developers. Team development also supports conflict resolution in cases where developers have revised the same resource since their last synchronization with the repository.

Common Terminology

A list of common terms and their definitions in Team Development methodologies.

- *Share* (or *sharing*) - The act or command that allows a single developer to share their code with other developers. This action creates a "link" between the developer's version of the code and the repository version of the code.

- *Checkout* - The act or command that allows a developer to bring code from the repository to the developer's local system for the first time, and effectively adds the developer to share the code. This action also creates a link between the developer's version of the code and the repository version of the code.
- *Commit* - When changes are made to any of the resources in a shared, local project, they are determined to be *outgoing* changes, which may be *co mmitted* to the repository. The *commit* command pushes these changes to the repository and any metadata about the commit (timestamp, comments, etc.).
- *Update* - Because a project is shared, one developer may have committed their own changes, which are not yet reflected in another developer's local copy. This action brings the incoming changes into the developer's local system, replacing the code with the repository version.

> ⚠ NOTE: Both commits and updates only change the code when there is a change made. Example: If you update from the repository, it will only replace code that has been changed, not all of the code.

- *Synchronize* - The _synchronize_ process simply refers to the task of connecting with the remote repository and evaluating the net difference between local and repository copies. Allows the developer to view any changes (pending commits and updates) before they occur, as well as view any conflicts that may exist.
- *Conflicts* - A *conflict* will occur when a developer tries to commit changes to a resource which has also been modified by another developer since the last update. Conflicts are a common occurrence in team development and are usually no cause for concern. Conflicts do require a developer to resolve the conflict, although most team sharing systems will give the developer tools to make conflict resolution easier. Example
- *Override and Update* - a common command used during conflict resolution that will allow the developer to disregard outgoing changes on a local file and take the incoming changes on the repository file.
- *Mark as Merged* - a common command used during conflict resolution that will allow the developer to disregard incoming changes from the repository file and use the changes in the developer's local file.
- *Database Information (DBI) Files - DBI files* are text files that contain metadata about the database. These files are used in team development to keep every developer's database synchronized with each other. In Servoy Developer, there are functions that will assist with using the DBI files to change the local developer's database.