Overview of Servoy Developer Concepts

This chapter lists definitions of terms and concepts used to describe items in the Eclipse IDE, and specifically, how they relate to Servoy. The chapter is organized from macro to micro; that is, it first describes concepts relating to the general development environment and goes down to its specific parts.

In This Chapter

- Workspace
- Projects
- Workbench
- Servoy Design Perspective
- Setting Preferences in Servoy

Workspace

A **workspace** is where all the developer's work is stored in Eclipse. It is a collection of Eclipse project files and metadata. The developer can have multiple workspace folders, but can only work with one at a time.

Upon installation, a default Servoy workspace is created as `../servoy_workspace`. This Servoy workspace folder is used to store solution metadata, style, and security information; it will also house separate folders for each Servoy solution. Users have the option to create new workspaces and switch between workspaces during development as needed using the **File** > **Switch Workspace** menu item.

Projects

Eclipse organizes different code sets into **projects**. Servoy has the following types of projects:

- **Resource projects** – Resource projects store information about databases, styles, users, and i18n data that are used by a Servoy solution.
- **Solution projects** – A solution is an application built within Servoy and can contain one or more forms, methods, and relations. Solutions are stored as separate projects in Servoy.

A Servoy solution is typically assigned a resource project upon creation. For more information, see Working With Resource Projects.

In case circular projects support is needed, see Setting Preferences in Servoy below.

Workbench

The Eclipse workbench is the graphical user interface window for Eclipse. It can contain many different perspectives, as described below, but only one perspective can be active at a time.

Servoy Design Perspective

The **Servoy Design** perspective is where the average Servoy developer will spend most of their development time.

Definitions:

A **perspective** in Eclipse is a set of views and editors, that is, everything one sees on a screen. The individual views and editors that appear can be customized and saved as a 'perspective.'

Views are windows within a perspective. Developers use views to view, edit, and manage these resources and data.

Editors are views in Servoy that provide users with specialized tools to edit resources, such as scripts, tables, and others.

The **Servoy Design** perspective contains views for working with Servoy solutions and all their parts. Servoy-specific views include the Solution Explorer, Script and Form Editors, and the Properties views. These views are described further in the Servoy Design Perspective chapter.

Other perspectives used when programming in Servoy include **Debug** and **Team Synchronizing**. These are discussed in their respective chapters as well.

Perspectives can be customized to the taste of the developer via the **Window** menu item. Views can be moved or removed, and the new perspective saved with a different name to enable the developer to switch between perspectives. Instructions for customizing perspectives are included found in the Eclipse user guide, accessible via the menu item **Help** > **Help Contents.**

Setting Preferences in Servoy

Servoy preferences can be accessed via the **Window** > **Preferences** menu (respectively, **Servoy Developer** > **Preferences** on a Mac). Three important preference items are highlighted below:

- **General** preferences control the basic appearance of Servoy including colors and fonts, content types and their file associations, general text editor behaviors, shortcut keys, security, and workspaces.
- **Javascript** preferences control basic editor options such as syntax coloring and warning icons.
- **Servoy** preferences control Servoy-specific options including **Form Editor** formatting tools, **Solution Explorer** behavior, and **Internationalization** options.
- **Dynamic Languages** preferences control Global DLTK settings, such as **Script Editor** code assist options or UI options. Also **Build Path** problems like the severity level of the Circular Dependencies problem marker can be configured on this page. Please note that if you already have a cycle, you have to remove and then add again to see the marker having the new severity level.

If installed, **SQL Explorer** options are also set using the **Preferences** menu.