NaturalONE

Introduction

Version 8.2.7

March 2013

Document ID: ONE-INTRODUCTION-827-20130320

Table of Contents

Preface

This introduction is organized under the following headings:

Eclipse: A Brief Introduction	What is Eclipse? This information is intended for Natural developers who are not yet familiar with Eclipse.
What is NaturalONE?	A brief overview of the different components of NaturalONE.
Different Modes for Developing Natural Applications	Information on the different modes for developing Natural applications using NaturalONE (local mode and Natural server mode).
Using a Version Control System	Information on the supported version control systems.

1 Eclipse: A Brief Introduction

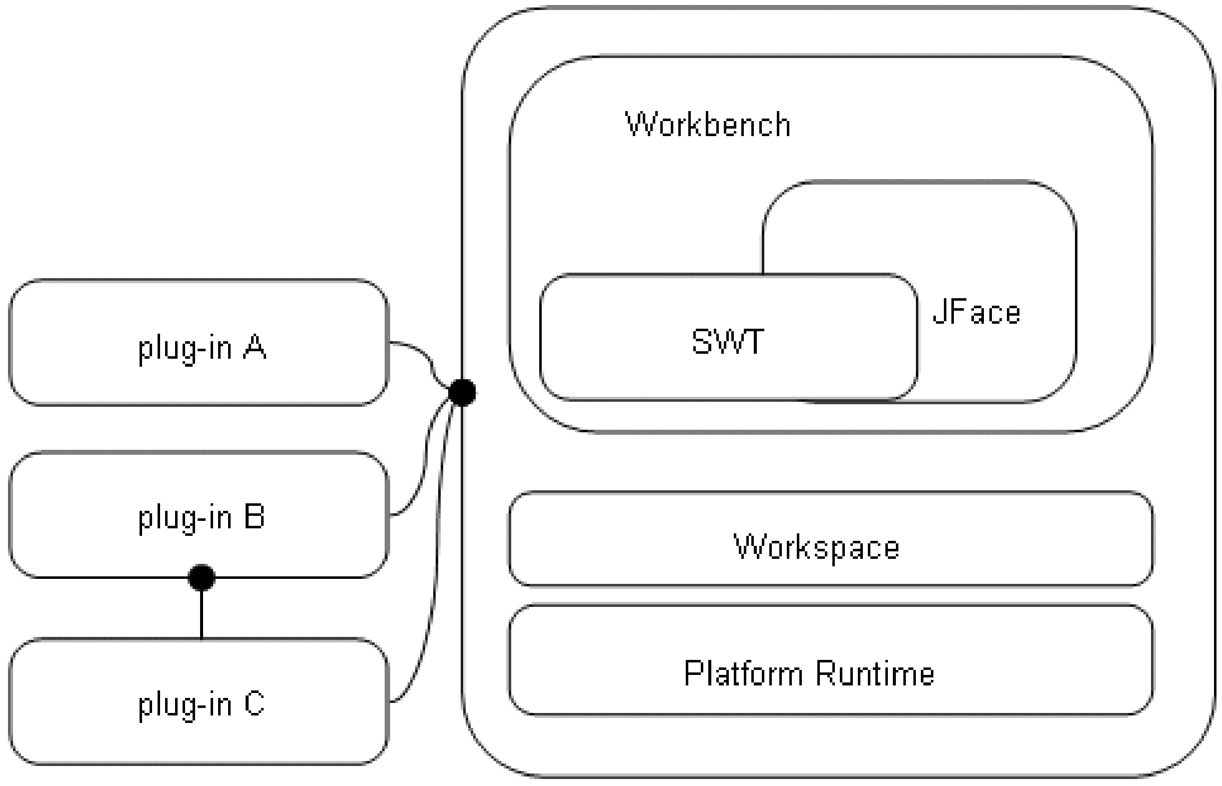
The Eclipse Platform is a very flexible open source development platform for tool integration. It provides a framework for building an integrated development environment from plug-in software components. More and more products nowadays are Eclipse-based, so that it is becoming increasingly important for developers and users to know what Eclipse is and what it offers.

The information in this document gives you a short overview of the basic Eclipse architecture, its components and standard user interfaces, as well as an introduction to what an Eclipse-based product may look like. This will help you understand Software AG products that are based on Eclipse.

Concept

The benefit of Eclipse is that it offers a single integrated platform for all development tasks. Plug-ins provide a feature of the Eclipse development environment and lead to the final Eclipse-based product. Plug-ins can be de-installed without impacting the Eclipse installation as such. Eclipse supports collaboration of development teams and is freely available from *www.eclipse.org*.

As shown in the following graphic, the basic Eclipse installation consists of three parts: the Workbench (which is further subdivided into the Standard Widget Toolkit and the JFace), the Workspace and the Platform Runtime.



- **Workbench**

  The *Workbench* is the user interface of Eclipse. The Standard Widget Toolkit (SWT) holds a set of widgets and graphics for building graphical user interfaces, such as buttons, menus, tree lists etc. With JFace, these elements are grouped into bigger, task-oriented units.

- **Workspace**

  The *Workspace* is the connection to the file system. It is used to create and manage project resources (such as files and folders).

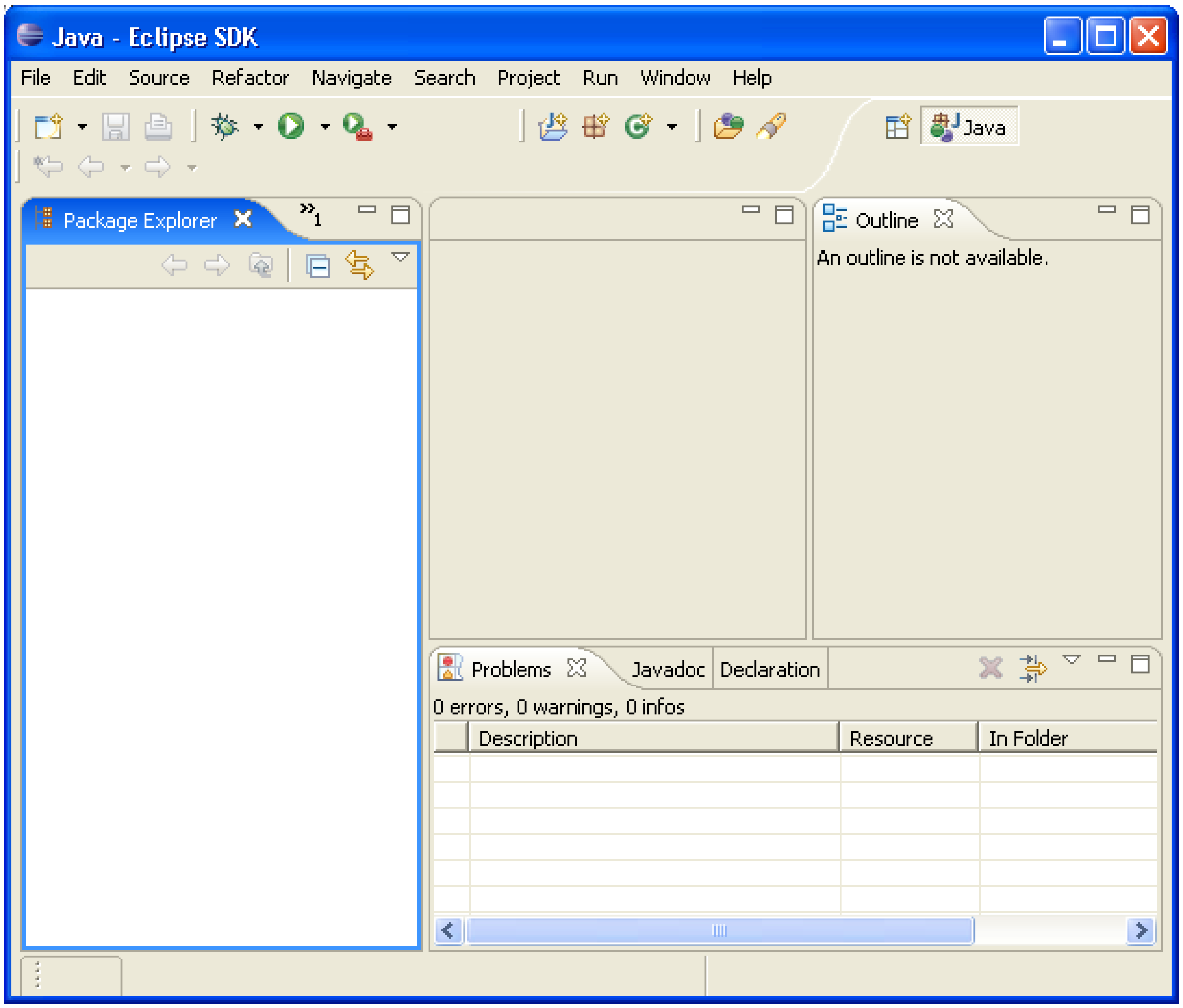
- **Platform Runtime**

  The *Platform Runtime* is the kernel that starts and runs the different components and takes care of the correct loading of plug-ins.

In addition to the basic Eclipse installation, the plug-ins extend the functionality of the Eclipse platform or other plug-ins. Plug-ins can be bundled as installable units. These installable units are called *features*. Each plug-in is connected to Eclipse via extension points, or to another plug-in, or both. Extension points are used to connect to plug-ins outside the Eclipse platform, but they also exist within Eclipse, as Eclipse itself is made of plug-ins.

Components

When you start the Eclipse Software Development Kit (SDK) for the first time, the workbench user interface and its components are displayed (after having closed the **Welcome** page), without any user-defined plug-ins. The Eclipse workbench is a platform for development tools. It provides the user interface structure for Eclipse and facilitates seamless integration of tools. The workbench consists of a collection of windows with menu bars, toolbars, shortcut bars and so-called perspectives. The name of the active perspective is shown in the title of the window. The following graphic shows an example of such a “bare” Eclipse workbench, using the Java perspective:



An outline is not available.

0 errors, 0 warnings, 0 infos

The workbench usually contains the following menus: **File**, **Edit**, **Navigate**, **Project**, **Window**, and **Help**. Other menus are plug-in dependent, or context-specific, based on the current perspective, editor or view. If you are a developer of plug-ins, you can develop and add new menus, editors, views or wizards.

In the following, the different components of the workbench are briefly introduced:

- Workspace
- Resources
- Wizards
- Views
- Editors
- Perspectives
- Preferences

- Properties

Workspace

As mentioned before, the workspace is the place in the file system where the different **resources** are stored. It consists of one or more projects. A project is a directory with several files and folders and has methods to build dependent resources.

Resources

Resources are items in the workspace, i.e. projects, folders, files and other dependent resources. These are all objects that will be or have been created with Eclipse. They are stored as normal files within the Eclipse workspace. A project holds several folders with files.

Wizards

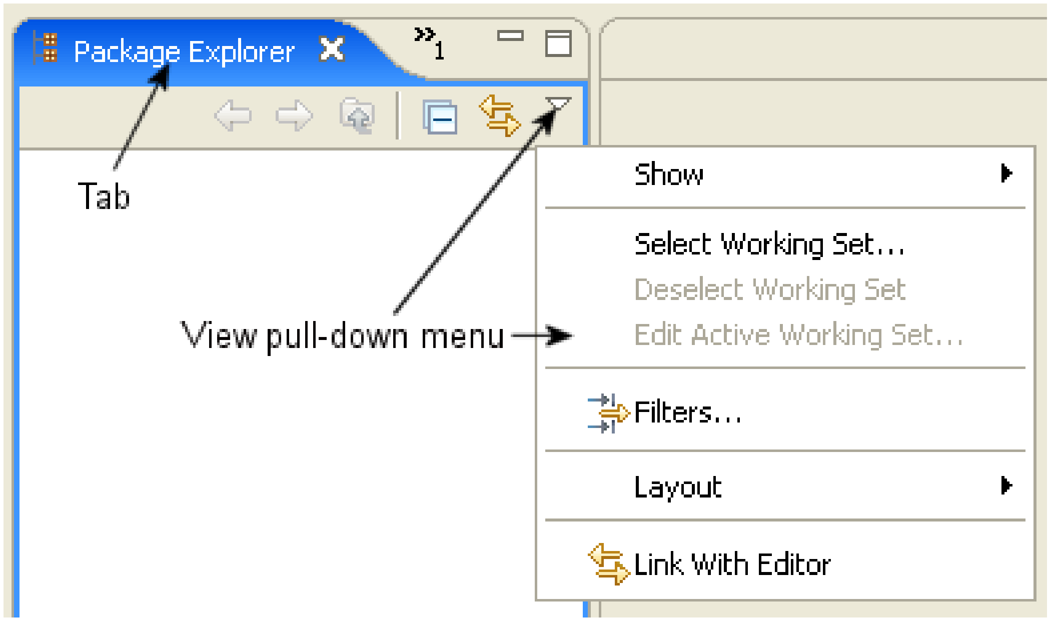
In Eclipse, most data is created using wizards. A wizard is an assistant that guides you step-by-step through a process, for example creating new resources or importing and exporting them.

Examples of wizards are:

- the New Project wizard;
- the New Class wizard;
- the New Package wizard;
- the Checkout wizard.

Views

A view is a visual component of the workbench that shows information, usually in a table or tree. It is used to navigate within a hierarchy of information, to open an editor or to display properties for the active editor. You choose **Window > Show View** to open the view with which you want to work. Several views can be stacked in a so-called tabbed notebook. To activate a view, you select its tab. Views also have their own context menus, which can be opened by right-clicking on the tab. Each view has a pull-down menu, which can be opened by selecting the down arrow to the right of the toolbar, below the tab. It contains functionality like sorting and filtering, which applies to the entire content of the view.



Examples of views are:

- Navigator;
- Package Explorer;
- Outline;
- Problems;
- Properties;
- Error Log.

Editors

An editor is another visual component of a workbench page. It is used to edit a document, to keep changes until the document is saved, or for browsing. Multiple editors may exist even for one document. There are content assistants, simple page and multiple page editors, and syntax high-lighters. Menus, toolbars and options in an editor are context-sensitive and change according to the environment. Eclipse has a list of registered editors, which are consulted first when you open a resource that needs an editor. If none of the editors in the list is suitable for the file type, the workbench checks automatically if any other editor from the underlying operating system is available (external editor). If an external editor is located, it will be launched.

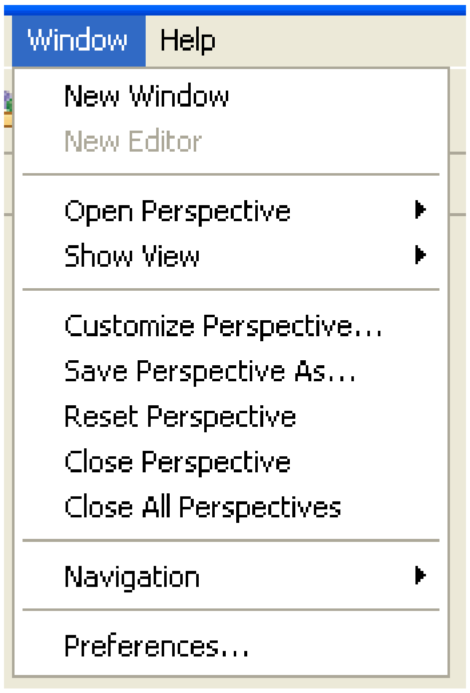
Examples of editors are:

- Java source editor;
- XML editor;
- Ant editor;

- Text editor;
- Plug-in editor.

Perspectives

A perspective can be described as a container that holds several views and editors, bundled for a specific task. Views and editors can be dragged and dropped to other places in the workbench so that the environment fits your needs and you have your personal working perspective. Only one perspective is visible at any time. A perspective can be managed with the commands available in the **Window** menu:

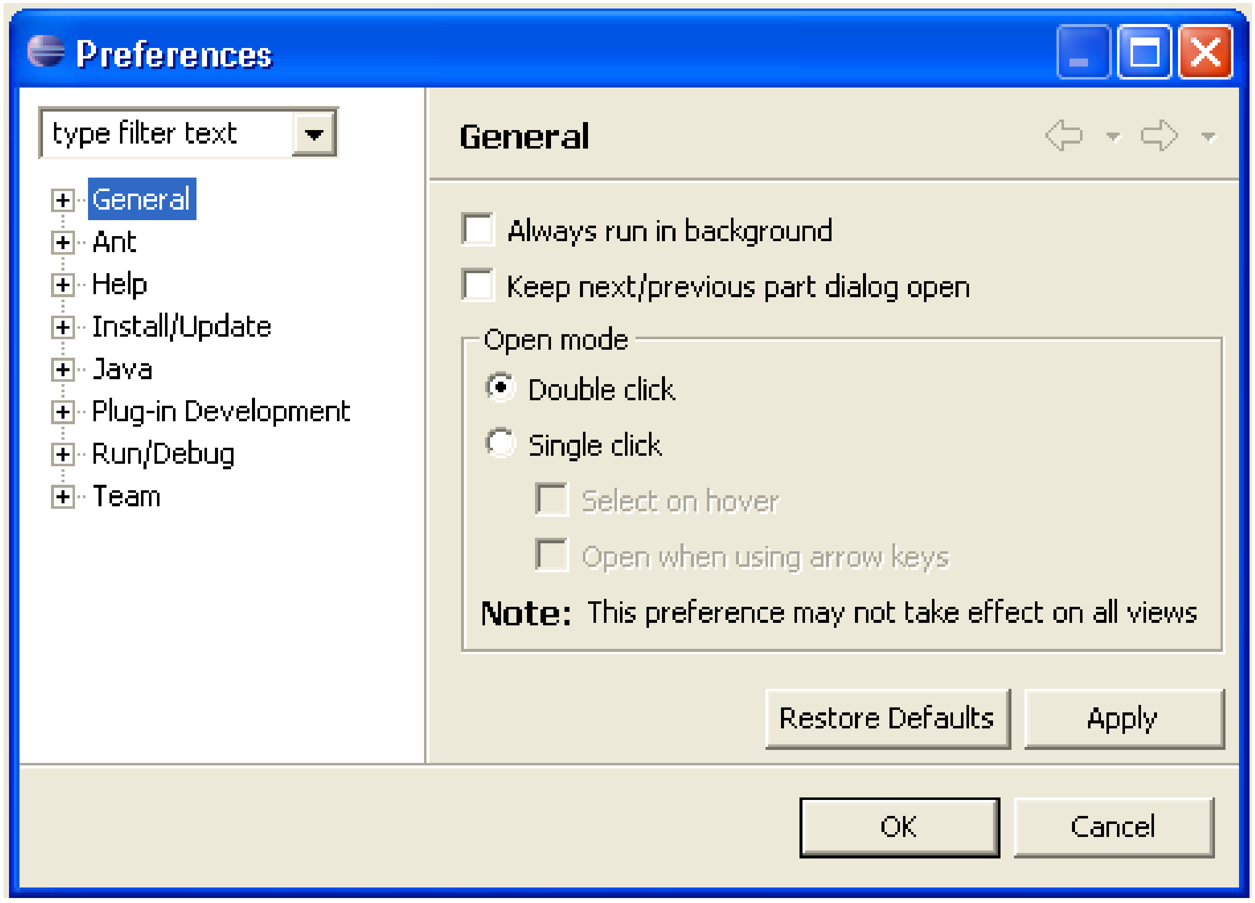


Examples are:

- Resource perspective;
- Java perspective;
- Debug perspective;
- Team synchronizing perspective.

Preferences

The **Preferences** dialog box sets the global preferences for various topics. It is available in the **Window** menu.



General

[ ] Always run in background

[ ] Keep next/previous part dialog open

Open mode

[● Double click]

○ Single click

[ ] Select on hover

[ ] Open when using arrow keys

Note: This preference may not take effect on all views

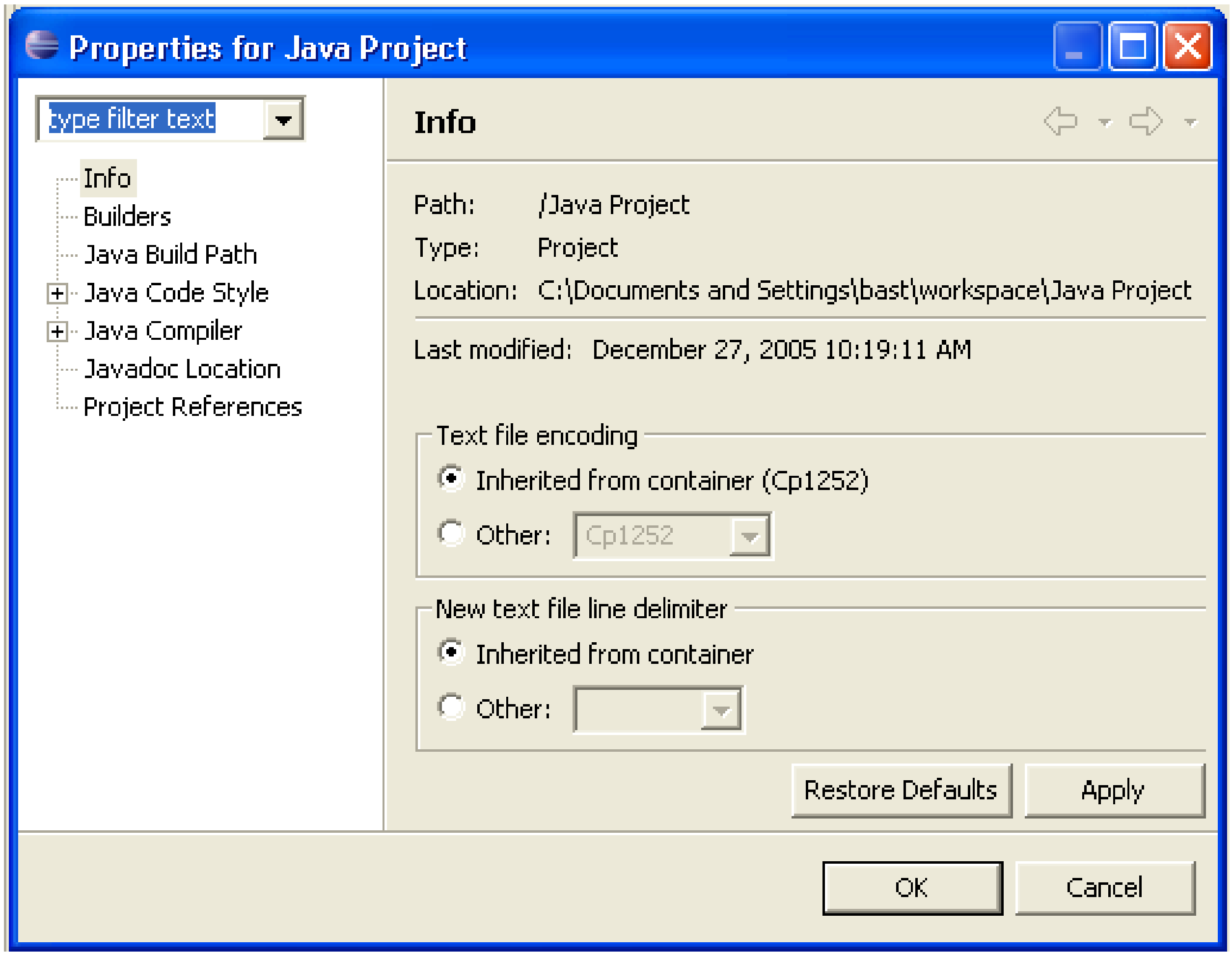
Examples of preferences are:

- Editor settings;
- Java compiler settings;
- Team settings.

The **Preferences** dialog box has a search facility (see the field **type filter text** in the graphic) and a history to navigate backwards and forwards through the pages.

Properties

The **Properties** dialog box shows and changes the properties of a resource or some other object in the active editor or view. The **Properties** command is available in the **File** menu or as the last command in the context menu of a resource.



Path: /Java Project
Type: Project
Location: C:\Documents and Settings\bast\workspace\Java Project

Last modified: December 27, 2005 10:19:11 AM

Text file encoding
[✓ Inherited from container (Cp1252)]
Other: Cp1252

New text file line delimiter
[✓ Inherited from container]
Other:

Examples of properties are:

- the properties of a file;
- the properties of a project.

The **Properties** dialog box has a search facility (see the field **type filter text** in the graphic) and a history to navigate backwards and forwards through the pages.

Using Eclipse-based Products

Developing plug-ins with Eclipse is one task, using Eclipse-based products is another one. For a user of Eclipse-based products, it is generally not necessary to have an in-depth knowledge of the Eclipse user interface, but it is helpful to have an idea of the main concepts, terms and components (as described above), as they keep reappearing in the user interface of the products. Here are some tips and tricks that apply to Software AG's Eclipse-based products.

Navigation

In most Eclipse-based products, navigation is done with the help of the **Navigator** view. It is usually displayed on the left side of a perspective and shows the available resources (projects, folders, files etc.) of the product. If you select a resource in the **Navigator** view and open the context menu, the available commands (for example, for copying, pasting, deleting etc.) are displayed.

Accomplishing tasks

To accomplish certain tasks like creating or editing resources, you select an item in the **Navigator** view, open the context menu and choose the desired command. The corresponding views and editors will usually open in a view on the right side of a perspective. You use them to interact with your product, for example, to enter, edit or add data. If your tasks require a step-by-step process, it is very probable that a wizard will open automatically and guide you through the process (for example, when importing or exporting resources). You just follow the instructions in the dialog boxes.

Logs and Infos

Information about what you are doing is normally available in information and log views at the bottom of your perspective, e.g. error logs, status information, etc.

Standard menus and commands

Products are integrated seamlessly into the Eclipse workbench. This means that you do not see where the standard Eclipse workbench ends and where the product-specific user interface starts. Eclipse-based products make use of standard Eclipse menus and commands, and they add their own functionality. So an Eclipse-based product usually still has the Eclipse “flavor”, but also its own components. As a consequence, product documentation describes only product functionality, and not the standard Eclipse functionality. The latter can be found in the standard Eclipse online help. If you miss the description of some functionality in the product documentation, it is thus very likely that you will find it in the Eclipse documentation (see the Eclipse help at ***http://www.eclipse.org/documentation/***).

Making life easier

Once you have established a working environment of views, editors and information windows, it is a good idea to save this environment as a customized perspective. To do so, you choose **Window > Save Perspective As**. You can re-open this perspective any time and thus do not have to create it again and again. This saves time and effort. If you want to restore the workbench to its default settings, you choose **Windows > Reset Perspective**.

Further Reading

If you are a new to Eclipse, this set of links will help you:

- ***http://www.eclipse.org/*** (the official Eclipse website)
- ***http://www.eclipse.org/articles/index.php*** (technical articles written by members of the development team and other members of the Eclipse community)
- ***http://marketplace.eclipse.org/*** (solutions for Eclipse)

The generally accepted Eclipse User Interface Guidelines can be found at the following location: ***http://wiki.eclipse.org/User_Interface_Guidelines***.

The very useful Eclipse online help is available at: ***http://www.eclipse.org/documentation/***.

Further information can be found in the following books:

- “The Java Developer's Guide to Eclipse” from Shavor, D'Anjou, Fairbrother, Kehn, Kellerman and McCarthy (Addison-Wesley)
- “Eclipse - Building Commercial Quality Plug-ins” from Clyberg and Rubel (Addison-Wesley)

Note: It is possible to use the Eclipse user interface using the keyboard only. See the Eclipse online help for detailed information.

2 What is NaturalONE?

General Information

NaturalONE is an Eclipse-based development environment for developing and maintaining Natural applications with web-based user interfaces and Natural services. It combines the functionality of several tools into a single development framework, and covers functionality across the entire product development lifecycle, including application development, testing, automated documentation of the sources in Predict, versioning, and deployment of the application into the production environment.

NaturalONE addresses developers who are used to work natively on mainframe, UNIX, Linux, OpenVMS or Windows platforms. Owing to the graphical user interface and many features for fast Natural source development (such as code assist), developers gain productivity using this environment. Developers who are already familiar with Eclipse can comfortably edit Natural sources in an environment to which they are already used; the environment always behaves in the same way, no matter whether Natural or Java applications are developed.

Basic Functionality for Natural Application Development

The basic functionality of NaturalONE is described in *Using NaturalONE*.

NaturalONE uses the standard Eclipse functionality and adds its own perspective, views and editors to the Eclipse workbench. In the Eclipse workspace, all files are organized in projects. A direct connection to a Natural server is not required for editing the Natural sources. However, for executing and debugging a Natural application, NaturalONE establishes a connection to the appropriate Natural runtime.

The following topics are covered below:

- Powerful Tool Set
- Powerful Natural Environment
- Direct Development on a Natural Server

Powerful Tool Set

Applications can be developed or maintained in the Eclipse environment.

Editors

To support developers and gain productivity during the development life-cycle, specific Natural editors are available:

- **Source Editor**
  The source editor uses a real incremental Natural parser. It supports the developer with syntax coloring of the Natural source and content assist for fast code writing. Both Natural programming modes (reporting mode and structured mode) are supported. For the development of international applications, bidirectional language support is enabled.

  NaturalONE makes use of the Natural source editor for data area editing. Data areas are defined with the `DEFINE DATA` statement. With the assistance of the Natural parser, syntactically correct data areas can be developed quickly and easily.
- **Map Editor**
  With the map editor, Natural maps can be defined graphically. This also includes bidirectional language support. It is possible to navigate through the parts of a map on a graphical basis using the **Outline** view. Inline rules or Predict rules can also be edited.
- **DDM Editor**
  DDMs which have been generated by Predict can be adopted for your application. It is also possible to create DDMs from scratch. This can be done for all types supported by Natural: Adabas, SQL, Tamino, VSAM and more. In the **Outline** view, the DDM structure is visualized in a hierarchical manner.

Debugging and Execution

Debugging and execution of applications is possible. Watchpoints, breakpoints, etc. make bug fixing less complex.

XML

With the XML toolkit, it is possible to generate functionality for the processing of XML documents. DTDs or schemas can be used for generating Natural data areas, parser implementations and serializers for XML documents, and vice versa.

Database Retrieval

The data browser provides fast access to Adabas or SQL databases. With just a few mouse clicks, it is possible to write a database retrieval and to display the retrieved data in the **Report Data** view. It is easy to check the content of the database and to test whether the application works correctly.

Powerful Natural Environment

Offloading the application into the Eclipse workspace gives you several advantages. The Natural builder keeps the dependencies of the application. This is visualized in the **Dependencies** view. The builder keeps track of the source changes. Owing to the label decorations, you can easily see which Natural objects have not yet been saved locally, not yet been compiled on the Natural server or not yet been versioned in your version control system. When the preferences are set properly, the builder recatalogs the appropriate objects on the Natural server using the Natural parameters that are defined in the Eclipse environment.

Basically, the builder supports two different workspace structures: one which is very Natural-related (that is, the application is based on the conventional Natural library structure), and another which allows you to define folders which follow the naming conventions of the underlying file system (Windows or Linux). With the latter workspace structure, the folders are mapped to "real" Natural libraries. Thus, you can structure your Natural applications in the workspace in a more logical way than with the conventional library structure. Folders can be nested.

Besides using folders in the workspace, it is also possible to use alternative (long) file names for the objects. These file names also have to follow the rules of the underlying file system. There is a mapping between the alternative file name and the Natural object name, where the Natural object name always follows the Natural naming conventions.

Where useful, wizards (for example, for creating new projects or new objects) are used.

Error messages are edited with the error message editor. They are also stored in the Eclipse workspace.

To gain productivity, features which support often repeated actions are available. For example, you can externalize code fragments into separate Natural objects. Or you can rename Natural objects using the refactoring feature. When you create a Natural source from scratch, a skeleton according to the Natural object type is automatically generated.

You can customize the NaturalONE environment to your specific needs. In the Natural preferences, you can change the behavior of the Natural builder, you can define the platform for which the Natural parser is to check the Natural syntax, or you can create your own code templates.

You can version your applications in a version control system. A Natural application consists of the Natural objects, configuration parameters, error messages, etc. A version control system enables you to merge the application pieces from several developers. After merging the sources in the repository of the version control system, the application is ready for deployment onto the designated platform. The deployment wizard ensures that the appropriate parts of the application are deployed onto the Natural server. See also ***Using a Version Control System***.

Out of the box, it is possible, for example, to execute, debug or test Natural applications or parts of them. The local Natural runtime is already configured for usage with NaturalONE. By default, no additional configuration is required. With the local Natural runtime, an EntireX broker environment is available and an RPC server is started. This allows you to develop RPC-based applications or to make use of web services.

Direct Development on a Natural Server

If you are used to work with Natural Studio and Natural's Single Point of Development (SPoD) concept, you will find comparable functionality in NaturalONE. The **Natural Server** view makes visible the information from the Natural servers which may be located on different platforms (on a mainframe, UNIX, Linux, OpenVMS or Windows platform). Using this view, you can edit your Natural sources directly on a Natural server, and catalog or even execute them directly on the server. Information about the server configuration is available; this is similar to the output of the Natural system commands `SYSPROD`, `SYSPROF`, `SYSFILE` and `UNLOCK`.

Natural for Ajax / Ajax Developer

Besides the basic functionality for Natural application development, NaturalONE enables you to create rich internet applications which use the Ajax (Asynchronous JavaScript and XML) technology. This feature, the Ajax Developer, is always installed together with the basic functionality, thus providing an integrated development and runtime environment for Natural for Ajax applications.

Ajax Developer includes a number of tools for creating and maintaining complex graphical user interfaces. Its central tool is the Layout Painter which is used to define layouts for HTML pages.

For detailed information, see *Natural for Ajax* and *Ajax Developer*.

Optional Components

In addition to the above-mentioned application development functionality which is always installed with NaturalONE, you can also install optional components for NaturalONE. Some of these optional components require that additional software is installed on a server.

The headings below correspond to the names that are used in the product selection tree of the Software AG Installer.

- Application Testing
- Lifecycle Manager
- Mainframe Tools
- Natural Construct
- Natural Engineer
- Predict

- Service Development

Application Testing

When you select **Application Testing** in the installer, the following context menu will be available in the **Navigator** view:

- **Testing**
  Used to directly test or create unit tests for various Natural objects, including subprograms, maps, subroutines and business services. For detailed information, see the *Application Testing* documentation.

  This optional component requires that EntireX is installed.

Lifecycle Manager

This optional component requires that CentraSite is installed. It runs in CentraSite Control, which makes use of a browser, and is used to deploy Natural applications and Natural for Ajax applications (or specific versions thereof) to different environments and to keep track of the installed applications in each environment.

When you select **Lifecycle Manager** in the installer, several files are copied to your NaturalONE installation. For detailed information on how to proceed after that, see *System Requirements* and *Installation and Configuration* in the *Lifecycle Manager* documentation.

Mainframe Tools

This optional component includes Mainframe Navigation which allows you to access and manipulate objects stored on a mainframe from Eclipse. These objects include datasets and members, as well as system objects such as active jobs or the console under the z/OS operating system. With Mainframe Navigation, the objects are displayed in a tree structure and can be browsed and edited in Eclipse. On the mainframe server, Mainframe Navigation is supported by Natural ISPF.

When you select **Mainframe Tools** in the installer, the **Mainframe Navigation** view will be available. For detailed information, see the *Mainframe Navigation* documentation.

Mainframe Navigation requires that additional software is installed on a server. See *Installation and Configuration* in the *Mainframe Navigation* documentation.

Natural Construct

When you select **Natural Construct** in the installer, the following context menu will be available in the **Navigator** view:

- **Code Generation > New Using Construct Model**
  Allows you to use your existing Natural Construct models in NaturalONE. You can also create new Natural Construct models or code frames. For detailed information, see *Using Natural Construct* in the *Code Generation* documentation.

  This optional component requires that additional software is installed on a server. See *Requirements* in the *Using Natural Construct* section of the *Code Generation* documentation.

Natural Engineer

When you select **Natural Engineer** in the installer, the following context menu will be available in the **Natural Server** view:

- **Natural Engineer**
  Used to convert Natural objects containing `INPUT` statements for use with Natural for Ajax pages. For detailed information, see the *Natural Engineer* documentation.

  This optional component requires that additional software is installed on a server. See *Setting Up a Natural Engineer Environment* in the *Natural Engineer* documentation.

Predict

When you select **Predict** in the installer, the following context menu will be available in the **Natural Server** view:

- **Predict Description and Generation**
  Used to document Natural sources in Predict via NaturalONE, manipulate and retrieve data stored by Predict on the server, and manage external objects (generation and administration). For detailed information, see the *Predict Description and Generation* documentation.

  This optional component requires that additional software is installed on a server. See *Setting Up a Predict Environment* in the *Predict Description and Generation* documentation.

Service Development

When you select **Service Development** in the installer, the following context menus will be available in the **Navigator** view:

- **Business Services**
  Used to create, maintain and test business services. For detailed information, see the *Business Services* documentation.

  This optional component requires that EntireX is installed. It also requires that additional software is installed on a server. See *Before You Start* in the *Business Services* documentation.
- **Code Generation**
  Used to generate Natural subprograms and data areas. For detailed information, see the *Code Generation* documentation.

3 Different Modes for Developing Natural Applications

General Information

NaturalONE has two modes for developing Natural applications. One, the Natural server mode, is similar to Natural Studio when used as a development client in a SPoD environment. The other, the so-called local mode, is the preferred development mode in NaturalONE; this mode allows you to work in a way that an Eclipse user expects.

When you are used to work natively on a Natural server or with Natural Studio in a SPoD environment, you face a "paradigm shift" when switching to local mode. In local mode, the sources are no longer stored or modified directly on the Natural server. The central place for keeping the sources is now the Eclipse workspace which is connected to a version control system.

More information on these development modes is provided in the topics below.

Local Mode

This is the preferred way when working with Eclipse.

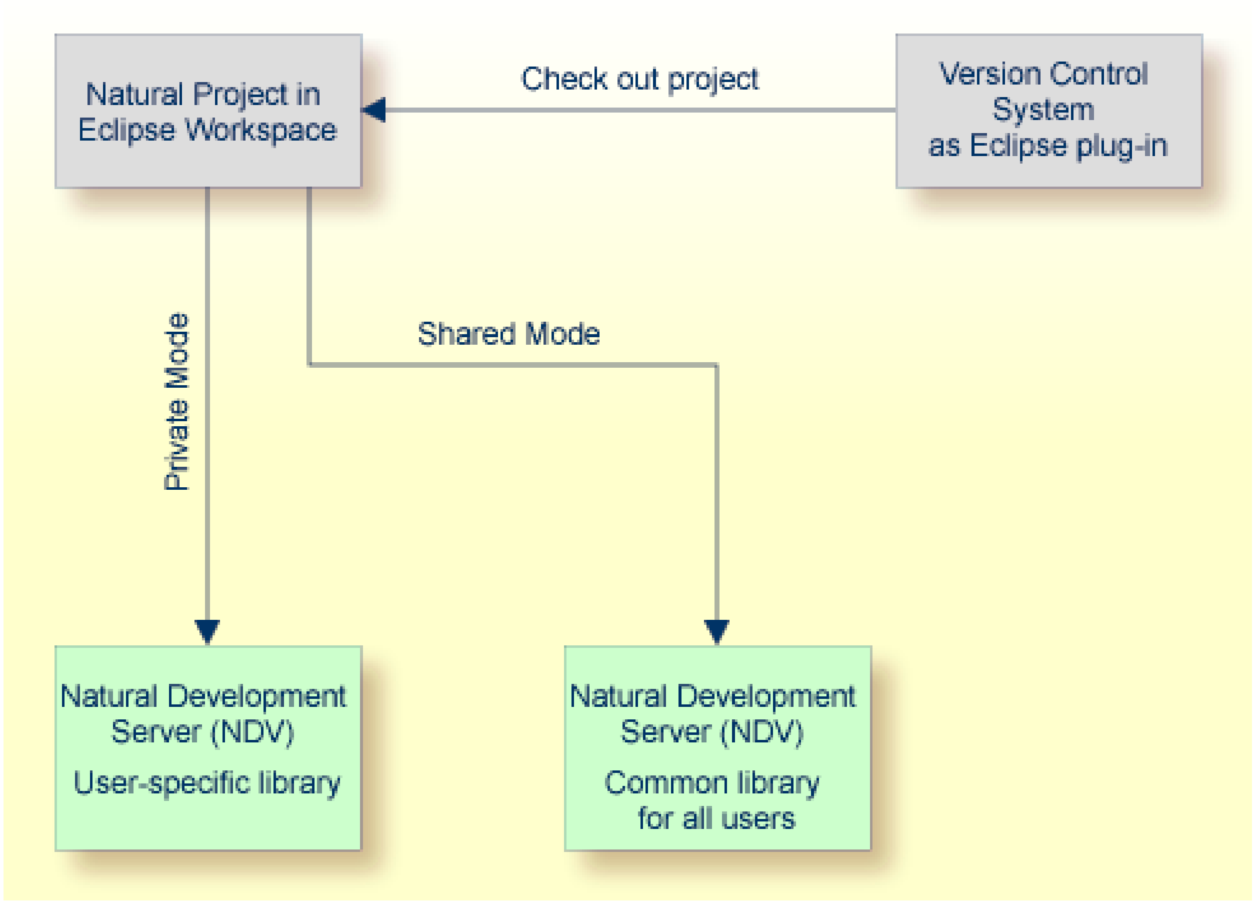
As a rule, you download a library from a Natural server into a Natural project in the local Eclipse workspace and then make your changes to the source code. In order to build the project, you have to update the appropriate Natural server, that is, you upload the changes to the server and catalog them there.

When several developers are working on the same Natural application, the Natural application is now spread over several Eclipse workspaces on different PCs. Two different modes in which the build is to be performed can be defined in the project properties:

- **Shared Mode**
  The build is performed in a common library on the Natural server that is shared by all users. This is the default mode.
- **Private Mode**
  The build is performed in a user-specific private library on the Natural server. This avoids a situation where one developer overwrites the changes of another developer.

For detailed information on these modes, see *Steplibs* in *Changing the Project Properties*.

Since the Eclipse features or plug-ins (for example, a plug-in for a version control system) expect the sources to be stored in the Eclipse workspace, you gain additional productivity in local mode because you can make use of third-party tools from other vendors.


Natural Project in Eclipse Workspace
Check out project
Version Control System as Eclipse plug-in
Private Mode
Shared Mode
Natural Development Server (NDV)
User-specific library
Natural Development Server (NDV)
Common library for all users

NaturalONE provides team support so that several developers can work in parallel with one application at the same time. Therefore, the configuration files are kept with the project and can thus be versioned in the version control system. Compiler options are always stored with a project.

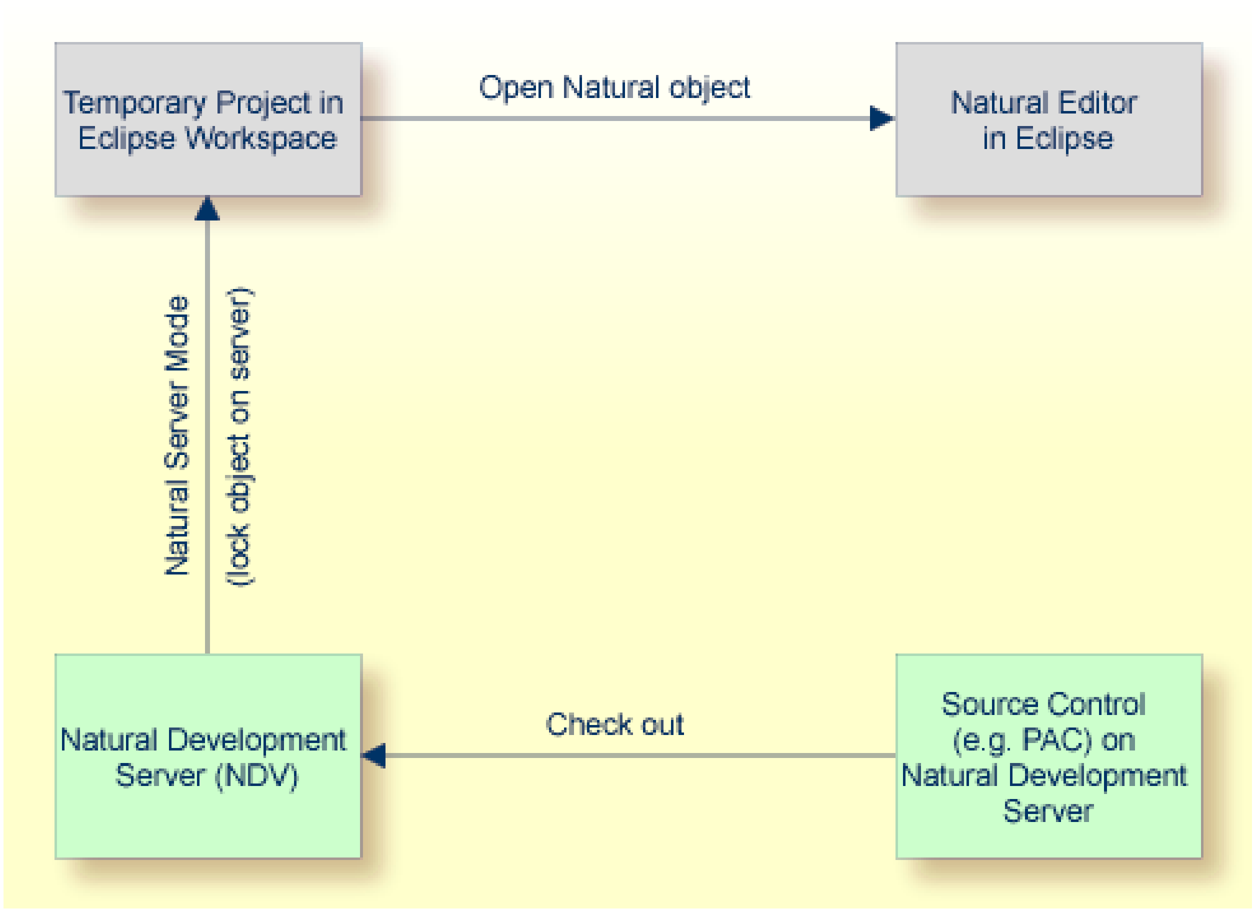
With NaturalONE, the versioned source is always considered to be the original source. You have to make sure that you do not copy a source to the Natural server and then manipulate the source on the server. If this happens, team support will fail. See also ***Using a Version Control System***.

Natural Server Mode

This is the classical way when working with Natural Studio. This mode mimics the SPoD behavior known from Natural Studio.

In this case, you do not have to download libraries to the Eclipse workspace. Instead, you work virtually directly with the objects on a Natural server. "Virtually", because the objects are temporarily downloaded to the Eclipse workspace for editing purposes. As long as you edit a source, it is locked on the Natural server in order to prevent concurrent modifications by different users.

Since the Natural sources remain on the Natural server, versioning is also done on the Natural server.


Temporary Project in Eclipse Workspace
Open Natural object
Natural Editor in Eclipse
Natural Server Mode
(lock object on server)
Natural Development Server (NDV)
Check out
Source Control (e.g. PAC) on Natural Development Server

Note: In Natural server mode, there is no distinction between private mode and shared mode.

4 Using a Version Control System

With NaturalONE, you have the possibility to version Natural applications in the repository of a version control system. For this purpose, the following tools are supported:

- **CVS**
  CVS is shipped with Eclipse. The *Workbench User Guide* in the Eclipse help (***http://www.eclipse.org/documentation/***) provides detailed information on how to set up a CVS repository and on team programming with CVS.
- **Subversion (SVN)**
  If you want to use Subversion, you have to install Subclipse which is a separate plug-in for Eclipse. This plug-in can be downloaded from ***http://subclipse.tigris.org/***. Make sure to download a Subclipse version which supports your current Eclipse version. Subversion is considered to be the successor of CVS.

Both versioning tools are client/server applications. Both require an Eclipse plug-in as the user interface and a server part (the versioning repository) which keeps the application.

Both plug-ins have their own perspectives. Each of these perspectives has to be enabled using the **Open Perspective** command.

For more information, see the documentation of your preferred version control system.