ScriptEclipse

User Manual v1.3.0

Table of contents

Table of contents

Description

ScriptEclipse is a plugin for the Eclipse Platform that allows developers to add functionality to Eclipse by simple scripting. The supported script languages are Python, Ruby, Groovy and JavaScript.

Each script will become an entry in the ScriptEclipse menu bar automagically and you can access it there or via a shortcut. It will not get easier to write all those nifty little features you always wanted, but didn't want to write a plugin for!

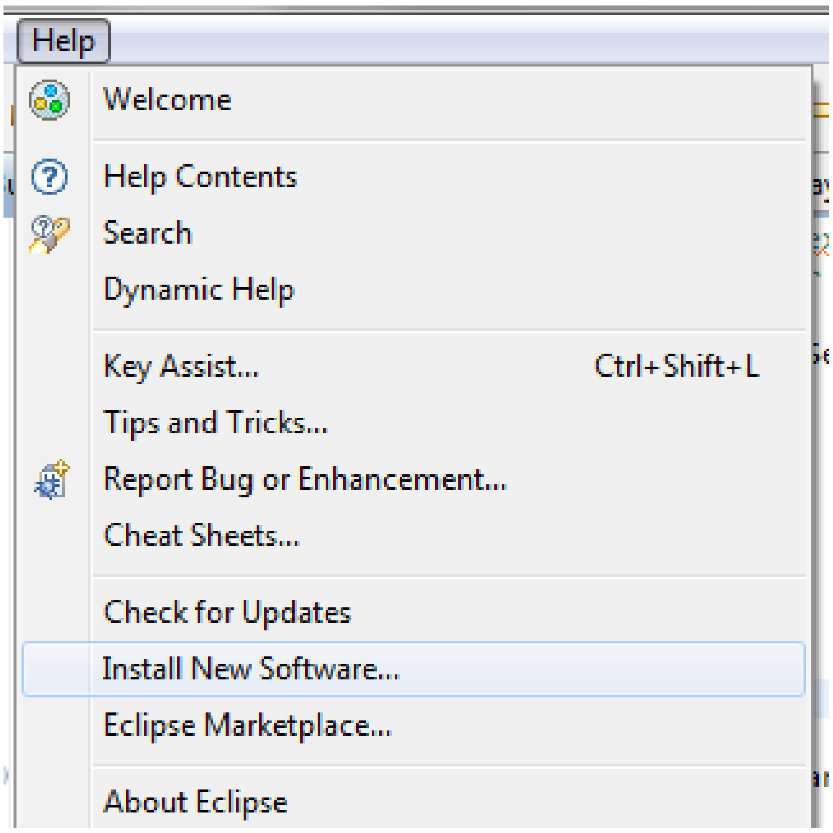
Supported are all products based on Eclipse >=3.5, including Eclipse itself, Rapid Application Developer, Zend Studio, Aptana and many more.

Installation

There are two ways to install the ScriptEclipse.

Installation via update site

Click Help → Install New Software...



Enter http://viplugin.com/scripteclipse into the textfield and press Add...
Select and install the ScriptEclipse plugin.

Installation via zip file

Download the latest version as zip file from the homepage http://viplugin.com/scripteclipse (download box is on the right side). Extract the zip file to the dropins folder of your Eclipse installation. Restart Eclipse (if running)

Usage

After installation you will see that you have a new menu called „ScriptEclipse“. The first time there is only one menu item called „Create _SCRIPTS_ Project“. Click on this and the plugin will generate a new Project with a few example scripts that should wet your appetite for exploring all the possibilities.

Every time you click on the ScriptEclipse menu it will be rebuilt, the scripts scanned again and hooks installed (see MetaData chapter).

stdout and **stderr** from your scripts are redirected to a console within your Eclipse workbench.

From within the Python scripts you can use all Eclipse classes from the following packages:

```
org.eclipse.jface.text,
 org.eclipse.ui.workbench.texteditor,
 org.eclipse.ui.editors,
 org.eclipse.jdt;resolution:=optional,
 org.eclipse.jdt.core;resolution:=optional,
 org.eclipse.jdt.debug;resolution:=optional,
 org.eclipse.jdt.launching;resolution:=optional,
 org.eclipse.jdt.ui;resolution:=optional,
 org.eclipse.ui.views;resolution:=optional,
 org.eclipse.ui.views.log;resolution:=optional,
 org.eclipse.ui.views.properties.tabbed;resolution:=optional,
 org.eclipse.osgi.services;resolution:=optional,
 org.eclipse.osgi.util;resolution:=optional,
 org.junit;resolution:=optional,
 org.eclipse.ltk.core.refactoring;resolution:=optional,
 org.eclipse.ltk.core.refactoring.source;resolution:=optional,
 org.eclipse.swtbot.eclipse.core;resolution:=optional,
 org.eclipse.swtbot.eclipse.finder;resolution:=optional,
 org.eclipse.swtbot.eclipse.spy;resolution:=optional,
 org.eclipse.swtbot.forms.finder;resolution:=optional,
 org.eclipse.swtbot.go;resolution:=optional,
 org.eclipse.swtbot.junit4_x;resolution:=optional,
```

org.eclipse.swtbot.swt.finder;resolution:=optional

Additionally there is the EclipseUtils class with the following methods:

```
    /**
     * Call an Eclipse Action
     * @param action the id of the action you want to call
     * @return true if the call was successfull
     */
    boolean callHandlerAction(String action)

    /**
     * create a new untitled text file and open it
     */
    void createNewFile()

    /**
     *
     * @return the currently active editor in the workbench
     */
    ITextEditor getActiveEditor()


    /**
     *
     * @return all files in the currently seleted project. The project
needs to be selected in the project explorer.
     */
    IFile[] getAllFilesInCurrentProject()

    /**
     *
     * @return the position of the cursor in the active editor
     */
    int getCursorPosition()
```

```java
/**
 *
 * @return the selected text in the active editor
 */
String getSelectedText()

/**
 *
 * @return the length of the current text selection
 */
int getSelectionLength()

/**
 *
 * @return the offset of the current text selection
 */
int getSelectionOffset()

/**
 *
 * @return the full text in the active editor
 */
String getTextOfActiveEditor()

/**
 * replace text in the active editor
 * @param offset the offset where to start the replacement
 * @param length the length of the replacement
 * @param text the text to replace it with
 */
void replaceTextOfActiveEditor(int offset, int length, String text)


/**
 * show the ScriptEclipse console view
 */
public static void showConsole()

/**
 * select the word under the cursor
 */
public static void selectWordUnderCursor()
```

```
/**
 * send a keypress event (e.g. acknowledge dialogs)
 * @param key the key to send
 */
public static void keypress(String key)
```

Metadata in Scripts

The scripts support the following Metadata:

@@name SUBMENU/NAME

> SUBMENUE the submenue under ScriptEclipse to put this script in
> NAME will be set as text in the menu item
> e.g. @@name My first script
> e.g. @@name Ruby/My first ruby script

@@shortcut SHORTCUT

> Defines the accelerator to activate the script
> e.g. @@shortcut SHIFT+ALT+F3

@@dontparse

> Defines that this file shall not be part of the menu.
> Comes in handy when you have helper scripts

@@callOnEditorOpened

> Defines that the script will be automatically executed when a new editor is opened.
> When you define this for the first time, you have to click on the ScriptEclipse menu, so that the scripts are reloaded and the hook is installed.

Tutorial Line-by-Line

```python
"""
Search the current selected text in google
@@name Google selection (Python)
@@shortcut SHIFT+ALT+G
"""

""" Import necessary classes from Eclipse, Java and ScriptEclipse """
from org.eclipse.ui import PlatformUI
from java.net import URL
from com.mbartl.scripteclipse import EclipseUtils

""" Open an internal or external brwoser """
internalBrowser = True
""" get the current selected text in the currently opened editor """
selection = EclipseUtils.getSelectedText()

""" if there's no selection do nothing """
if selection != "":
    """ build the URL to google including the searchterm """
    url = URL("http://www.google.com/search?q=" + selection)

    """ create a browser and open the URL """
    url = URL("http://www.google.com/search?q=" + selection)
    browserSupport = PlatformUI.getWorkbench().getBrowserSupport()
    if internalBrowser:
        browserSupport.createBrowser("browser").openURL(url)
    else:
        browserSupport.getExternalBrowser().openURL(url)
```

Have a look at the example scripts. It's really easy ;)

Credits

This product includes the following software:

Jakarta Codec

Jython

Groovy

Jruby

The license can be found in the doc folder within the plugin.