Centipede

Splitting big jobs into many small steps

# Introduction

Centipede is a program designed to let you break repetitive tasks down into a sequence of small steps. This document will walk you through the usage, from the first time you start it to higher-level tasks such as programming your own actions (note: experience of programming in Python (see [www.python.org](http://www.python.org/)) or a .NET language is expected for this section.

# Contact

Any questions, bug reports, suggestions can be sent to [Richard.Lovely@robn.com](mailto:Richard.Lovely@robn.com).

# Installing

Extract the centipede archive to an easy to find location. Run the two included installers to ensure you have the required frameworks. Don’t worry too much about installation options – the defaults are fine for most users of Centipede.

# Terminology

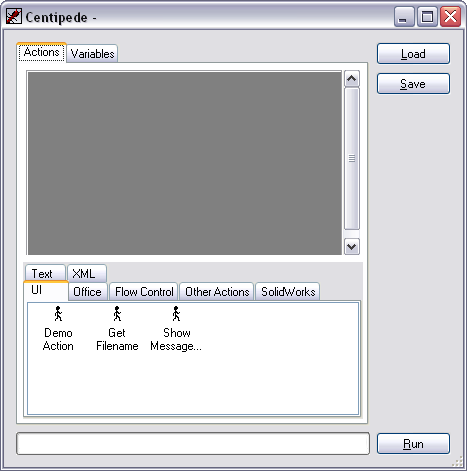
* **Action**: One specific thing done, this can be anything from opening a file, or making changes to it, or asking the user (you) for a specific value to use in a calculation. **Branch Actions** are also actions, but they are discussed below.
* **Argument**: These are values stored with the **actions** that determine exactly how the action works. This could include things like filenames, part names, and whether to over-write an existing file when saving.
* **Branch Action**: A **Branch Action** is an **action** that can decide which of two routes the **job** will go next. Normally jobs are processed from top to bottom, without skipping any out. A **branch action** gives you the option of jumping to any other action, anywhere in the job, whether before or after the current action. In combination with **Variable Actions**, this gives you very versatile control of job execution.
* **Job**: a list of **actions** executed in order to complete a task. Example jobs could be things like **Generate a Drawing of a shaft** or **Make a paint specification report**.
* **Plugin**: Almost all **actions** are created as plugins. Plugins can be installed by simply copying the dll file into the plugins folder. Plugins can be authored in any language that is compatible with the Microsoft .NET framework.
* **Python**: Python is a scripting language. It was chosen for its simplicity and ease of learning.
* **Variables**: In programming, a **Variable** is a specified location in memory used to store data for use later. In Centipede, the principle is similar: every piece of information that is needed by more than one action is stored in a variable, this includes files, numbers, strings, etc. They can easily be manipulated by **Variable Actions**.
* **Variable Actions**: an **Action** that uses simple **python** code to directly manipulate **variables**.

# First Run

The first time you run centipede, you will see the Favourites List. This will be empty as you don’t yet have any favourites setup. We will go through the process of adding favourites later.

# Creating a new Job

From the favourite jobs screen, create a new job by clicking *New*. You will then be taken to the Centipede Editor screen. From here, you can add, remove and edit actions, add, edit and remove variables, and run the job.

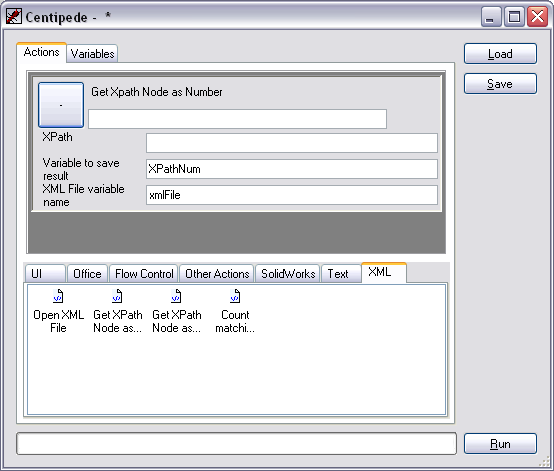


# Adding and Removing Actions

The available actions are listed under the tabs in the bottom half of the Actions tab. There are two ways of adding an action to the job: either double click the icon to add it to the end, or drag the icon into the action list, letting you choose where to put it. You can remove actions by right clicking in the action list, and choosing Delete from the pop-up menu.

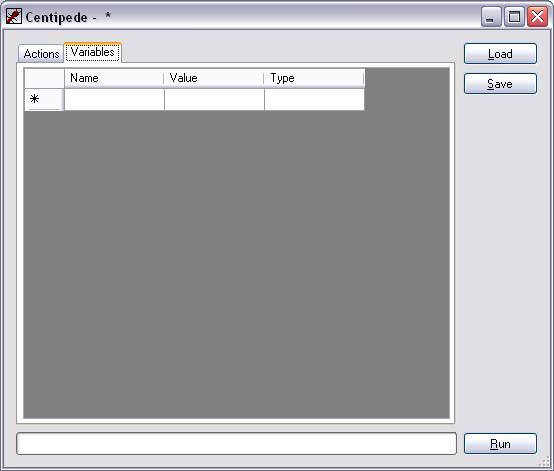
# Editing Actions

Most actions will have one or more arguments, and a comment. The comment is shown in the text box on every action in the list. To see the action’s arguments, click the + button. The argument list will appear below the comment text box. You can usually hover the mouse over an argument’s name to get information on the argument.



# Variables

On the variables tab, you can see all the variables in the job. It should be empty if this is a new job – it will be populated as needed when the job is run. You can also create custom variables by clicking in an empty row and typing. Deleting variables is the same as in the action tab – right click, and choose Delete.

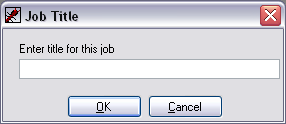


# Embedded Variables

Certain arguments can use variables. To do this, surround the argument name with curly braces, {like this}.

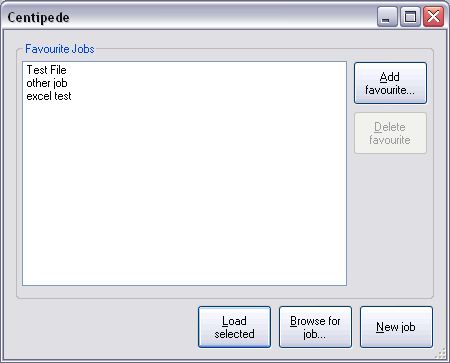
# Saving jobs

When you click Save in the editor, you will first be asked for a name for the job. This will be shown in the title bar of the editor and in the favourites list if you choose to add the job.



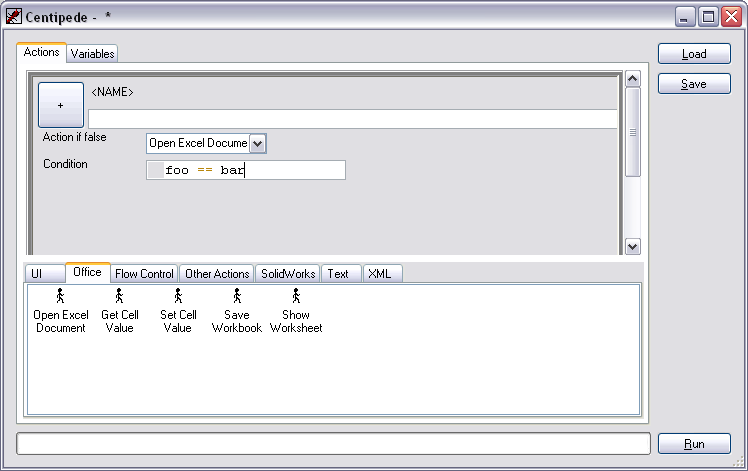
# Favourite Jobs

To add a job to the favourites list, click Add…, and find the job file. To load a favourite either double-click it in the favourites list, or select it and click Load Selected. You can load a job without adding it to the favourites list by clicking Browse for Job…



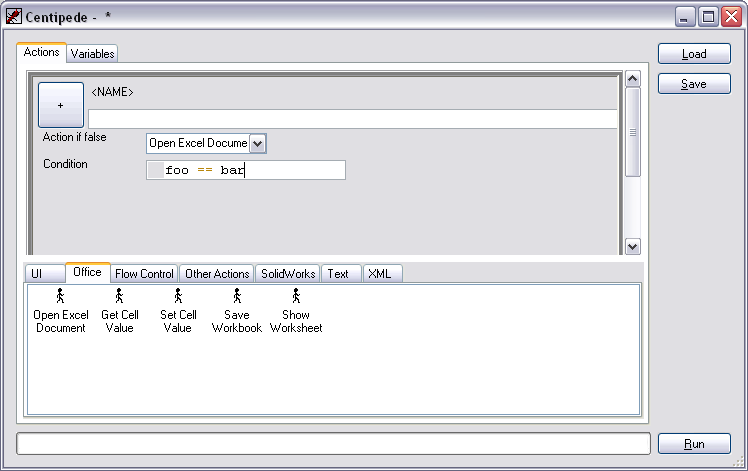
# Branch Actions

A branch action has two arguments, a condition, and Action if False. Condition expects a Boolean expression – one that can either be true or false, similar to the condition used in an IF formula in Excel. It can use any variable, and doesn’t need braces to do so. Note that “a is equal to b” is written as a==b, with a double equals. Using a single = is a syntax error. When the action is processed, the condition will be evaluated. If it is calculated as True, nothing special will happen. If, on the other hand, it is False, execution will instead move to the action selected in Action if False.



# Variable Actions

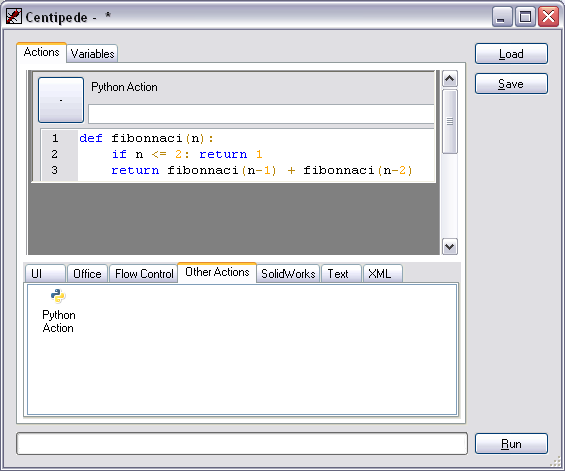
A variable action has two arguments: an expression and a destination variable. When the action is processed, the expression is calculated, and the answer is stored in the named variable. You can use any variable (including the destination variable) in either argument, again without braces.



# Advanced topics

## Python Actions

If you are familiar with the Python programming language, this is an extremely powerful action, allowing almost anything to be accomplished.



## Authoring Plugins

Included in the Centipede zip archive is a CentipedeAction Visual Studio project template. This is compatible with Visual Studio 2010 and later, and contains everything you need to know.

