MAGIC SOFTWARE ENTERPRISES LTD.

Magic eDeveloper

Tools Infrastructure Sample

04 6 5 4 3 2 1

January 2004

Table of Contents

Chapter 1

Introduction

Welcome to the Tools Infrastructure Sample guide. This document will guide you through a sample tool-type application to better acquaint you with the Tools Infrastructure concept and how you can make use of it.

The Speller Tool

The sample tool application is a Speller utility that lets the developer run a spell-checking procedure on individual tasks.

> **Disclaimer**
>
> The Speller utility is a sample utility and might not cover all aspects of spell checking or string representations in a program.

Prerequisites

The Speller utility utilizes Microsoft® Word 2000's spell-checker mechanism. Make sure Word is installed before running this utility.

The Speller utility uses new Tools Infrastructure enhancements that are available in eDeveloper 9.4 SP3.

The Sample's Flow

This document will guide you step-by-step through the sample utility. Any action you have to perform is prefixed by ❋. For example:

❋ Run Program #5.

Descriptions of the effects or implications of the actions appear between the actions and may be prefixed by 👁, when the effects or implications are visible. For example:

👁 A new dialog window is opened.

When the effects or implications are not visible, the description will be prefixed by ⓘ. For example,

ⓘ The object is released.

Notes and Tips

The Note and Tip sections, such as the one below, provide you with important information about the execution flow and feature implementation.

> **Note**
>
> Notes and Tips point out important aspects about the way the feature is executed or implemented, or the way the logic is written.

Further Reading

This sample does not cover all aspects of the Tools Infrastructure. For more information about the Tools Infrastructure feature, see the *eDeveloper Reference Guide*.

Chapter 2

Setup

You should follow the steps below to properly set up the Speller utility as an additional tool in your development environment.

File Placement

- Copy the Speller folder that is provided with the Sample package and place it in the Add_On folder of your eDeveloper directory.

Ini Modifications

- Open your Magic.ini file.
- Go to the [TOOLS_MENU] section.
- Copy the lines provided in the Speller.ini file located in the Speller folder and paste them into the [TOOLS_MENU] section.

> **Note**
>
> If the [TOOLS_MENU] section does not exist, create the section and then paste in the copied lines.

That's It

You are ready to begin executing and going through the sample.

Chapter 3

Running the Speller Utility

The purpose of the Speller utility is to search for text phrases in the program structure, and then let you view spelling mistakes so that you can correct them. After reviewing a program's spelling errors, you can update your program to reflect all modifications.

Activating the Utility

❋ Run the Mggenw.exe file.

❋ Open any application and import the Sample_Prog.exp file supplied with the Sample package.

ⓘ This file contains a simple online program with various phrases that contain spelling mistakes.

❋ Select the program and activate the Speller from the Tools pulldown menu as shown in Figure 3-1.

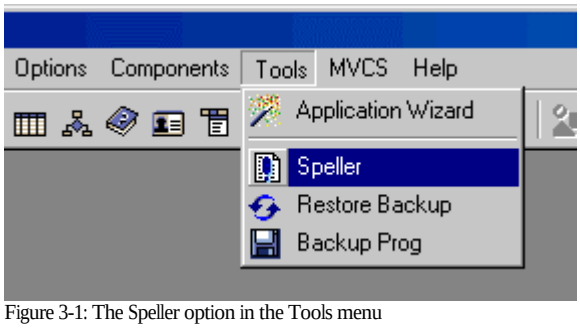

Figure 3-1: The Speller option in the Tools menu

ⓘ Now the Speller utility is activated and shows the first spelling error found.

Using the Speller

Using the speller utility is very simple. Each incorrect phrase is displayed in a dialog box that lets you fix any spelling mistakes found.

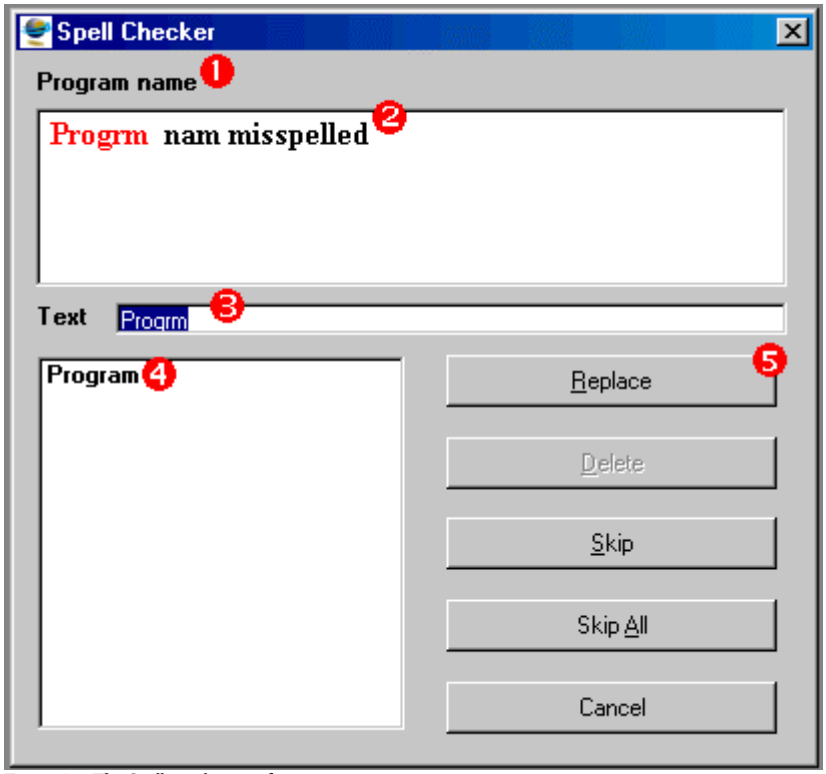

Figure 4-1: The Speller utility interface

- Element #1, as shown in Figure 4-1, describes the location in the program to which the text belongs.
- Element #2 displays the current phrase with the incorrect word marked in red. You can click in the text to make corrections.
- Element #3 is a text box in which you can type alternative text.
- Element #4 is a list of suggestions found for the incorrect word.
- Element #5 is a set of operational buttons you can use to handle incorrect words.

✱ Continue and correct incorrect phrases as desired.

Completing the Modifications

Once you finish correcting the last phrase, the utility prompts you to update the program to reflect the modifications.

✱ You should confirm the update.

Speller Procedure

The Speller utility has now completed its execution and the program is updated with the spelling modifications.

Backup and Restore

Every program handled by the Speller utility is backed up. This means that you can restore the program to the way it was before you made changes to the spelling.

Restore

- Remain parked on the program you just updated.
- Activate the Restore Backup option from the Tools menu.
- The utility prompts you to confirm the restore request.
- You should confirm the operation.
- The program reverts back to its original state before the spelling corrections were made.

Backup

In addition to the backup and restore ability, the Speller utility also lets you back up any program using the Backup Prog option in the Tools menu. You can restore the program at a later stage using the Restore Backup option.

Chapter 4

How it Works

You have just enhanced your development environment by adding an "Add-on tool". This tool was an ordinary eDeveloper application, which means that not only can you add additional proprietary tools, you can even create your own tools.

This chapter will guide you through the Speller tool's internal settings to give you a better understanding of how to create your own tools.

The Tool Concept

Before we look at how the Speller utility was defined, you should first understand the basic concept of a Tool and the tool application execution cycle.

A Tool

A tool is simply an eDeveloper application that is defined as a Tool option under the Tools menu, using the INI file.

- Open the Speller.ini file.
- The first line instructs eDeveloper to place a splitter line in the Tools menu.
- The second, third, and fourth entries instruct eDeveloper to place tool application entries under the Tools menu.
- The fourth comma-delimited segment describes the MFF application file: `Add_On\Speller\Speller_CTL.mff`.

> **Note**
>
> A tool application must be an MFF application file.

Pre-tool Activation

When a tool is activated and before it is actually executed, the engine performs a set of operations on the application in use, according to your operation configurations.

- You can see an indication of the pre-activation operations file in the tools entry line in the sixth delimited segment: `Add_On\Speller\Speller_Pre.txt`.
- Open the `Speller_Pre.txt` file.
- This file may contain several operations, where each line represents an operation.
- In the example provided, the pre-activation operation instructs the engine to perform an Export procedure. The details of this operation show that:
  - The export file name is: `Add_on\Speller\OriginalExport.tmp`
  - The repository to be exported is the program repository: `Repository=P`
  - The range of programs is the current program: `Range=@`

> **Note**
>
> To learn more about the additional options of the Export operation and other operations, see the *eDeveloper Reference Guide*.

- The Export operation means that when the tool is activated, eDeveloper will export the current program to the defined file location.
- This export file will be used later on by the tool-type application to extract details of the program and to create a modified export file.

> **Tip**
>
> Constructing the Tools menu is quite easy. To learn more about the Tools menu, see the *eDeveloper Reference Guide*.

Running the Tool Application

After the pre-activation operations are completed, the tool application is executed and performs the tasks it is defined to perform.

In the next chapter you will read about the Speller application in more depth.

Post-tool Activation

After the tool application is closed, the application that was previously in use is automatically re-opened and the engine executes all the operations defined in the post-operation file.

- See the reference to the pre-activation operations file in the tools entry line in the seventh delimited segment: `Add_On\Speller\Speller_Post.txt`.

> **Note**
>
> Usually the post-activation operations are a derivative of the tool application's flow, and therefore the post operation file is usually created dynamically by the tool application.

More Information

That concludes the flow of a tool application. See the next chapter to find out what the sample tool actually does.

Chapter 5

The Tool Application

The main idea behind the Tools Infrastructure is that a tool application is a regular eDeveloper application which you can design and develop yourself.. The Speller utility is a specific application for a specific use, but by understanding how it works, you can use the information to develop your own tools.

❋ Create a new application and import the Speller.exp file.

> **Note**
>
> The Speller application uses different programming methodologies to provide the spell-checking mechanism. This document only discusses the points that are relevant to the Speller utility as a tool-type application.

Initialization

Just like any other application, the Task Prefix of the Main Program is used to initialize the application flow.

Global Values

One of the services a tool application receives from the engine is a set of values that provide more details of the application in which the tool was activated.

❋ Go to line #7 in the Task Prefix of the Main Program.

❋ Looking at lines #7 and #8 you can see how the GetParam function is used for pre-defined global values. The tool application retrieves information such as the number of the entry on which the developer was parked before activating the tool, or the prefix of the application in use.

❋ Go to line #5 in the Task Prefix of the Main Program.

❋ Some of the pre-defined global values, such as MG_ToolEntry provide information on the tool application settings.

Several Tools in a Single MFF

Looking at the [TOOLS_MENU] line provided by the Speller.ini file you can see that all three tool entries activate the same tool application. You might be wondering how the same application runs different flows for different tool entries. This is done using the MG_ToolEntry global value.

❋ Go to line #18 in the Task Prefix of the Main Program.

👁 There are three Raise Event operations that initiate each flow of the tool according to the tool entry line describing each tool.

👁 The tool entry line information was previously retrieved using the MG_ToolEntry global value. In this case the name of the entry was used to determine the application flow.

The Speller Main Flow

If the tool entry line has been identified as the Main Speller tool entry, the Speller Main event is triggered and its handler executes the appropriate flow.

❋ Go to the Speller Main handler.

👁 Lines #1 - #6 initialize the flow by deleting the memory tables and backing up the export file.

> **Note**
>
> The export file is the export file created by the pre-activation operations file.

Processing the Export File

❋ Go to line #8 in the Speller Main handler.

👁 This operation calls a program that processes the export file.

👁 This called program simply runs on the export file lines, stores them in a memory table, locates and extracts phrases from it, and stores the phrases in a second memory table.

👁 Have a look at the called program to see how this processing is performed.

> **Important**
>
> Processing of the export file in this way is specific to the Speller utility. Other tool applications can handle the export file differently if any handling is even required.

Tip

Some tool applications may collect information about the application they are defined to handle by exporting an export document instead of a regular internal export file. You can customize the information provided in the export document by using a document template file.

Running the Spelling Checker

After the phrases have been extracted, the Check Spellings program is called.

Creating a Modified Export File

❋ Go to line #17 in the Speller Main handler.

👁 If the spelling mistakes were found and the user confirms the modification, program #5 is called.

👁 Program #5 runs on the export lines to produce a new export file.

Concluding the Flow

❋ Go back to the Task Prefix of the Main Program, line #27.

👁 The flow is concluded by calling program #6, which creates the post-activation operations file, and raises the Close Application event to close the tool application and return to the application in use.

Restore and Backup

The Restore Backup and Backup Program flows are much simpler.

❋ In the Main Program, go to the Speller Restore handler.

👁 This flow checks for the existence of the backup file, and if it exists prompts the user for confirmation.

👁 The flow is concluded like the main flow, by calling program #6 to create the post operation file, and then closing the application.

❋ Go to the Speller Backup handler.

👁 This flow copies the export file as a backup file.

👁 The flow is concluded like the main flow, by calling program #6 to create the post operation file, and then closing the application.

The Post-activation Operations

The post-activation operations are usually an important part of the tool application, especially if the tool is defined to affect the application in use.

❋ Open any other application, park on a program and activate the Backup Prog tool option.

❋ Open the Add_on\Speller\Speller_Post.txt file located in the eDeveloper directory.

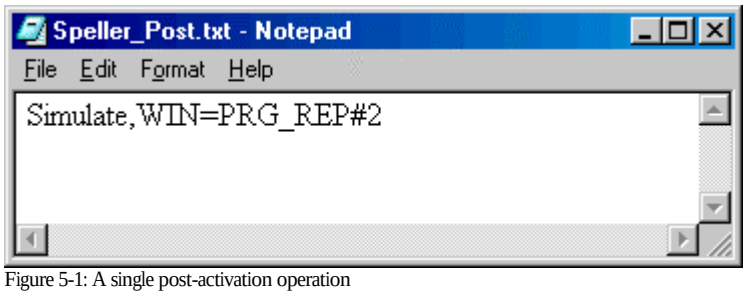

Figure 5-1: A single post-activation operation

👁 The file should appear like the one in Figure 5-1, showing a single operation line that instructs the engine to park on the entry that was previously selected.

> **Note**
>
> When the application is reopened after the tool is closed, the application does not retain its last location. However, using the Simulate option you can configure the application to park on the last location or at any other location.

❋ Park on the program you just backed up, and activate the Restore Backup tool option.

❋ Open the Add_on\Speller\Speller_Post.txt file located in the eDeveloper directory.

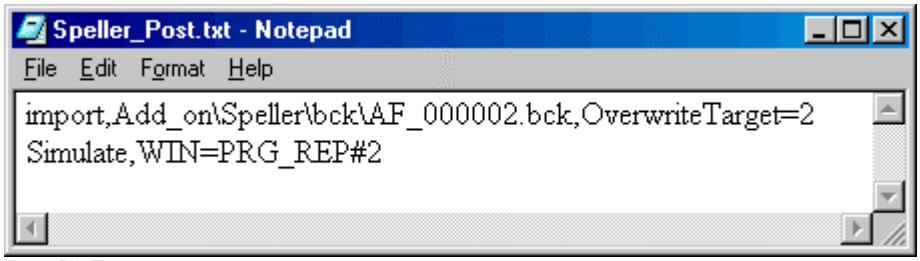

Figure 5-2: Two post-activation operations

👁 The file should appear like the one in Figure 5-2, showing two operation lines that instruct the engine to import the backup file by overwriting the parked program and then to park on the entry that was previously selected.

Chapter

6

Summary

This concludes the Tools Infrastructure Sample document. In this guide we saw how you can create your own tool-type applications that you can include in your development environment. We showed how you can use eDeveloper's Tools Infrastructure by using the Speller application as an example.

We hope that this guide helped you to understand more about the Tools Infrastructure feature and the concepts behind it. You can use this feature to create and share numerous tool applications that will enhance your development environment.