

EiffelRSS

Eiffel Programming Tips

Michael Käser <kaeserm@student.ethz.ch>

Martin Luder <luderm@student.ethz.ch>

Thomas Weibel <weibelt@student.ethz.ch>

Abstract

This document contains various handy tips for programming with ISE Eiffel.

Contents

1	Downloading files	1
2	Portable file access	2
3	Detecting the underlying operating system	2
4	Accessing features statically	3
5	Links	3

1 Downloading files

With EiffelNet:

```
local
  http_source: HTTP_PROTOCOL
  file_destination: FILE_PROTOCOL
  http_address: HTTP_URL
  file_address: FILE_URL
  transaction: SINGLE_TRANSACTION
do
  -- create the connection to the source
  create http_address.make ("http://eiffelrss.berlios.de\
→/Main/AllRecentChanges?action=rss")
  create http_source.make (http_address)

  -- print http-url-string of source
  io.put_string ("http location: ")
  io.put_string (http_source.location)
  io.put_new_line

  -- create the connection to the local file
  create file_address.make ("eiffelrss.rss")
  create file_destination.make (file_address)

  -- print file-url-string of file
  io.put_string ("file location: ")
  io.put_string (file_destination.location)
  io.put_new_line

  -- create the transaction
  create transaction.make (http_source, file_destination\
→)

  -- test if the transaction is correct
  io.put_string ("transaction is correct: " + \
→transaction.is_correct.out)
  io.put_new_line

  -- download the file
  transaction.execute

  -- test if the transaction succeeded
  io.put_string ("transaction succeeded: " + transaction\
→.succeeded.out)
  io.put_new_line
end
```

Multiple files can be downloaded with class `MULTIPLE_TRANSACTION`, which is a container of `TRANSACTION`.

2 Portable file access

With class `PLAIN_TEXT_FILE`:

```
local
  input_file , output_file: PLAIN_TEXT_FILE
  — Files for input and output
do
  — Open files
  create input_file.make_open_read (". / path / to / input . txt \
  → ")
  create output_file.make_create_read_write (". / path / to / \
  → output . txt ")

  — Read from input
  input_file.read_line
  io.put_string (input.last_string + "%N")

  — Write to output
  output_file.put_string (input.last_string)
  output_file.put_new_line

  — Close files
  input_file.close
  output_file.close
end
```

This works for Unix and Windows. Make sure to only use slashes (/) in the path, backslashes (\) don't work under Unix.

3 Detecting the underlying operating system

The Gobo libraries feature the class `KL_OPERATING_SYSTEM` to detect the underlying operating system:

```
local
  operating_system: KL_OPERATING_SYSTEM
do
  create operating_system
  io.put_string ("DotNet? " + operating_system.is_dotnet \
  → .out + "%N")
  io.put_string ("Unix? " + operating_system.is_unix.out \
  → + "%N")
```

```
io.put_string ("Windows? " + operating_system.\
→is_windows.out + "%N")
end
```

4 Accessing features statically

To access a feature from a class without creating it, just write:

```
feature {CLASS_NAME}. feature_name
```

For example:

```
create logfile.make_filename_threshold ("logfile.log", \
→feature {LOGFILE}. Developer)
```

5 Links

Links to external documentation, libraries etc.

- [Reflection](#)¹
- [Eiffel Software Directory](#)²
- [ETest - Eiffel Unit Testing Tool](#)³
 - [ETest Tutorial](#)⁴
- [EiffelVision2 Tutorial](#)⁵

¹<http://fluri.computerscience.ch/index.php?page=projects&subpage=m>

²<http://eiffelzone.com/index.html>

³<http://www.cs.yorku.ca/eiffel/etest/>

⁴<http://www.cs.yorku.ca/eiffel/prism/estudio55.htm>

⁵<http://www.cs.yorku.ca/eiffel/vision2/>