WebTool

version 0.8

Typeset in LaTeX from SGML source using the DocBuilder-0.9 Document System.

Chapter 1

WebTool User's Guide

WebTool provides a easy way to use web based tools with Erlang/OTP. It configures and starts a webserver as well as all available tools.

1.1 WebTool User Guide

1.1.1 Introduction

WebTool provides an easy and efficient way to implement web based tools with Erlang/OTP. WebTool configures and starts the webserver and the various web based tools.

All tools that shall run under WebTool must have a *.tool file in the code path or in its priv directory. When WebTool starts it searches the code path for such files. For each `ebin` directory in the path, the `priv` directory is also searched. The *.tool files contain the configuration data for each web based tool.

1.1.2 Starting WebTool

Start WebTool by calling the function `webtool:start/0` or `webtool:start/2`. If `webtool:start/0` is used the start page of WebTool is available at *http://localhost:8888/* or *http://127.0.0.1:8888/*, and the directory containing the root directory for the webserver, is assumed to be `webtool-<vsn>/priv`.

Use `webtool:start/2` if the default path for the root directory, port, ip-number or server name can not be used. See the Reference Manual for webtool [page 8] for more information.

WebTool, with the default configuration as in `start/0`, can also be started with the `start_webtool` script which is available in the `priv` directory of the WebTool application. See the Reference Manual for start_webtool [page 6] for further information about this script. For Windows users, the batch file `start_webtool.bat` can be used for the same purpose.

1.1.3 Using WebTool

Start WebTool and point the browser to the corresponding URL. At the top of the page there is a frame with a link named *WebTool*. Click that link and a page where it is possible to start the available tools will appear in the main frame.

1.1.4 Start a web based tool

Click on the link labeled *WebTool* in the topmost frame, select the checkbox for each tool to start and click on the button labeled *Start*. A link to each tool that WebTool succeded to start will appear in the topmost frame.

1.1.5 Stop a web based tool

Click on the link labeled *WebTool* in the topmost frame. Select *Stop Tools* in the left frame. Select the checkbox for each tool to stop and click on the button labeled *Stop*.

1.1.6 Develop new web based tools

WebTool can be used as a framework when developing new web based tools.

A web based tool running under WebTool will typically consist of three parts.

- A *.tool file which defines how WebTool can find the tool's configuration data
- The Erlang code generating the web interface to the tool (HTML code)
- The tool itself.

In most cases it is a good idea to separate the code for creation of the html-pages and the code for the logic. This increases the readability of the code and the logic might be possible to reuse.

The *.tool file

When WebTool starts it searches the current path for `*.tool` files to find all available tools. The *.tool file contains a version identifier and a list of tuples which is the configuration data. The version identifier specifies the *.tool file version, i.e. not the version of webtool. Currently the only valid version is "1.2" and the only valid configuration tag is `config_func`. `config_func` specifies which function WebTool must call to get further configuration data for the tool. This means that a *.tool file generally must look like this:

```
{version,"1.2"}.
[{config_func,{Module,Function,Arguments}}].
```

`Module` is the name of the module where the callback function is defined. `Function` is the name of the callback function, and `Arguments` is the list of arguments to the callback function.

The configuration function

The *.tool file points out a configuration function. This function must return a list of configuration parameters (see the Reference Manual for webtool [page 8]).

The `web_data` parameter is mandatory and it specifies the name of the tool and the link to the tool's start page. All other parameters are optional.

If the tool requires any processes to run, the `start` parameter specifies the function that WebTool must call in order to start the process(es).

The `alias` parameters are passed directly on to the webserver (INETS). The webserver has three ways to create dynamic web pages CGI, Eval Scheme and Erl Scheme. All tools running under WebTool must use Erl Scheme.

Erl Scheme tries to resemble plain CGI. The big difference is that Erl Scheme can only execute Erlang code. The code will furthermore be executed on the same instance as the webserver.

An URL which calls an Erlang function with Erl Scheme can have the following syntax:

```
http://Servername:Port/ErlScriptAlias/Mod/Func<?QueryString>
```

An `alias` parameter in the configuration function can be an ErlScriptAlias as used in the above URL. The definition of an ErlScripAlias shall be like this:

{`alias`,{`erl_alias`,`Path`,[`Modules`]}}, e.g.

{`alias`,{`erl_alias`,"/testtool",[helloworld]}}

The following URL will then cause a call to the function helloworld:helloworld/2 (if WebTool is started with default settings i.e. servername “localhost” and port 8888):

`http://localhost:8888/testtool/helloworld/helloworld`

Note that the module `helloworld` must be in the code path of the node running WebTool.

Functions that are called via the Erl Scheme must take two arguments, `Environment` and `Input`.

- `Environment` is a list of key/value tuples.
- `Input` is the part of the URL after the “?”, i.e. the part of the URL containing name-value pairs. If the page was called with the URL:
  `http://localhost:8888/testtool/helloworld/helloworld?input1=one&input2=two`
  `Input` will be the string `"input1=one&input2=two"`. In the module `httpd` in the INETS application there is a function `parse_query` which will parse such a string and return a list of key-value tuples.

An `alias` parameter in the configuration function can also be a normal path alias. This can e.g. be used to point out a directory where HTML files are stored. The following definition states that the URL `http://localhost:8888/mytool_home/` really points to the directory `/usr/local/otp/lib/myapp-1.0/priv`:

{`alias`,{"/mytool_home","/usr/local/otp/lib/myapp-1.0/priv"}}

See the INETS documentation, especially the module `mod_esi`, for a more in depht coverage of Erl Scheme.

A small example

A Hello World example that uses Erl Scheme would look like this. Note that this example does not have a process running and thus does not need a `start` parameter in the configuration function.

helloworld.erl:

```
-module(helloworld).
-export([config_data/0]).
-export([helloworld/2]).

config_data()->
    {testtool,
     [{web_data,{"TestTool","/testtool/helloworld/helloworld"}},
      {alias,{erl_alias,"/testtool",[helloworld]}}]}.

helloworld(_Env,_Input)->
    [header(),html_header(),helloworld_body(),html_end()].

header() ->
    header("text/html").

header(MimeType) ->
    "Content-type: " ++ MimeType ++ "\r\n\r\n".

html_header() ->
    "<HTML>
       <HEAD>
          <TITLE>Hello world Example </TITLE>
       </HEAD>\n".

helloworld_body()->
    "<BODY>Hello World</BODY>".

html_end()->
    "</HTML>".
```

To use this example with WebTool a *.tool file must be created and added to a directory in the current path, e.g. the same directory as the compiled `helloworld.beam`.

testtool.tool:

```
{version,"1.2"}.
[{config_func, {helloworld,config_data,[]}}].
```

When `helloworld.erl` is compiled, start WebTool by calling the function `webtool:start()` and point your browser to *http://localhost:8888/*. Select WebTool in the topmost frame and start TestTool from the web page. Click on the link labeled *TestTool* in the topmost frame.

WebTool Reference Manual

Short Summaries

- Command start_webtool [page 6] – WebTool Start Script
- Erlang Module webtool [page 8] – WebTool is a tool used to simplify the implementation of web based tools with Erlang/OTP.

start_webtool

The following functions are exported:

- `start_webtool application [ browser ]` Start a WebTool Application

webtool

The following functions are exported:

- `start()-> {ok,Pid}| {stop,Reason}`
  [page 8] Start WebTool.
- `start(Path,Data)->{ok,Pid}|{stop,Reason}`
  [page 8] Start WebTool with default configuration.
- `stop()->void`
  [page 8] Stop WebTool.
- `debug_app(Module)->void`
  [page 8] Debug a WebTool application.
- `stop_debug()->void`
  [page 9] Stop debugging an application and format the trace log.
- `Module:Func(Data)-> {Name,WebData}|error`
  [page 9] Returns configuration data needed by WebTool to configure and start a tool.

start_webtool

Command

The start_webtool script starts WebTool, a WebTool application and a web browser pointing to this application.

Exports

start_webtool application [browser]

Starts WebTool, the given WebTool Application and a web browser pointing to this application.

If no argument is given, a list of available applications is displayed, e.g.

```
>start_webtool
Starting webtool...
WebTool is available at http://localhost:8888/
Or  http://127.0.0.1:8888/

Usage: start_webtool application [ browser ]

Available applications are: [orber,appmon,crashdump_viewer,webcover]
Default browser is 'iexplore' (Internet Explorer) on Windows or else 'netscape'
```

To start any of the listed applications, give the application name as the first argument, e.g.

```
>start_webtool webcover
Starting webtool...
WebTool is available at http://localhost:8888/
Or  http://127.0.0.1:8888/
Starting webcover...
Sending URL to netscape...done
```

The WebTool application WebCover is then started and the default browser is used. The default broser is Internet Explorer on Windows or else Netscape.

To use another browser, give the browser's start command as the second argument, e.g.

```
>start_webtool webcover mozilla
Starting webtool...
WebTool is available at http://localhost:8888/
Or  http://127.0.0.1:8888/
Starting webcover...
Sending URL to mozilla...done
```

If the given browser name is not known to WebTool, WebTool will run it as a command with the start URL as the only argument, e.g.

```
>start_webtool webcover mybrowser
Starting webtool...
WebTool is available at http://localhost:8888/
Or  http://127.0.0.1:8888/
Starting webcover...
Starting mybrowser...
```

Here the command "`mybrowser http://localhost:8888/webcover`" is executed.

See Also

webtool(3) [page 8]

webtool

Erlang Module

WebTool makes it easy to use web based tools with Erlang/OTP. WebTool configures and starts the webserver httpd.

Exports

`start()-> {ok,Pid}| {stop,Reason}`

Start WebTool with default data, i.e. port 8888, ip-number 127.0.0.1, and server-name `localhost`. If port 8888 is in use, port 8889 is tried instead. If 8889 is also in use, 8890 is tried and so on. Max number of ports tried is 256.

The `mime.types` file and WebTool's own HTML files are assumed to be in the directory `webtool-<vsn>/priv/root/conf`.

`start(Path,Data)->{ok,Pid}|{stop,Reason}`

Types:

- Path = string() | standard_path
- Data = [Port,Address,Name] | PortNumber | standard_data
- Port = {port,PortNumber}
- Address = {bind_address,IpNumber}
- Name = {server_name,ServerName}
- PortNumber = integer()
- IpNumber = tuple(), e.g. {127,0,0,1}
- ServerName = string()
- Pid = pid()

Use this function to start WebTool if the default port, ip-number,servername or path can not be used.

`Path` is the directory where the `mime.types` file and WebTool's own HTML files are located. By default this is `webtool-<vsn>/priv`, and in most cases there is no need to change this. If `Path` is set to `standard_path` the default will be used.

If `Data` is set to `PortNumber`, the default data will be used for ip-number (`127.0.0.1`) and server name (`localhost`).

`stop()->void`

Stop WebTool and the tools started by WebTool.

`debug_app(Module)->void`

Types:

- Module = atom()

Debug a WebTool application by tracing all functions in the given module which are called from WebTool.

`stop_debug()->void`

Stop the tracing started by `debug_app/1`, and format the trace log.

CALLBACK FUNCTIONS

The following callback function must be implemented by each web based tool that will be used via WebTool. When started, WebTool searches the Erlang code path for *.tool files to locate all web based tools and their callback functions. See the WebTool User's Guide [page 1] for more information about the *.tool files.

Exports

`Module:Func(Data)-> {Name,WebData}|error`

Types:

- Data = term()
- Name = atom()
- WebData = [WebOptions]
- WebOptions = LinkData | Alias | Start
- LinkData = {web_data,{ToolName,Url}}
- Alias = {alias,{VirtualPath,RealPath}} | {alias,{erl_alias,Path,[Modules]}
- Start = {start,StartData}
- ToolName = Url = VirtualPath = RealPath = Path = string()
- Modules = atom()
- StartData = AppData | ChildSpec | Func
- AppData = {app,AppName}
- ChildSpec = {child,child_spec()}
  See the Reference Manual for the module supervisor in the STDLIB application for details about child_spec().
- Func = {func,{StartMod,StartFunc,StartArg}, {StopMod,StopFunc,StopArg}}
- AppName = StartMod = StartFunc = StopMod = StopFunc =atom()
- StartArg = StopArg = [term()]

This is the configuration function (`config_func`) which must be stated in the `*.tool` file.

The function is called by WebTool at startup to retrieve the data needed to start and configure the tool. `LinkData` is used by WebTool to create the link to the tool. `Alias` is used to create the aliases needed by the webserver. `Start` is used to start and stop the tool.

See Also

start_webtool(1) [page 6], WebTool User's Guide [page 1]

Index of Modules and Functions

Modules are typed in *this way*.
Functions are typed in `this way`.