WebTool

version 0.7

Typeset in LaTeX from SGML source using the DOCBUILDER 3.2.2 Document System.

Chapter 1

WebTool User's Guide

WebTool provides a easy way to use web based tools with Erlang/OTP. WebTool configures and starts the webserver and the various web based tools.

1.1 WebTool User Guide

1.1.1 Introduction

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

All tools that shall be managed by webtool must have a *.tool file in the current path. When WebToool starts it searches the path for such files to find 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 to start WebTool the start page of WebTool is availible at the url: *http://localhost:8888/* or *http://127.0.0.1:8888/*. If `webtool:start/0` is used to start WebTool the directory containing the root directory for the webserver, is assumed to be the directory: `webtool-vsn/priv`.

The `start/2` function has three variants; use one of this if the default path for the priv directory, the default port, ip-number or servername not can be used. See WebTool Reference Manual for more information.

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 on the link and a page where it is possible to start the available tools will appear in the main frame. Select the tools to start and click on the button labeled *Start*.

1.1.4 Add new Web based Tools

When WebTool starts it searches the current path for `*.tool` files, to get configuration data for the different tools. The *.tool consists of a version tuple and a list of tuples, Today WebTool only uses the `config_func` tuple, which describes the function call WebTool must do to get the configuration data for the tool.

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

- `Module`, The name of the module where the callback function is defined.
- `Function`, The name of the callback function.
- `Argument`, A list of the arguments to the callback function.

1.1.5 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.

1.1.6 Stop a Web Based Tool

If a web based tool no longer is used, 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*.

1.1.7 Develop new Web based tools

Developing web based tools to Erlang is easy and straightforward. The use of WebTool minimize the need of knowledge about the webserver in Erlang/OTP.

Design of tool

Web based Erlang tools is best built around the generic behaviours There are many reasons for this, among other things it will be easier to develop a web based tool that holds a state and to maintain the code.

Since the webserver can get queries from many users the module that get the incomming requests from the different users it is best built around the generic behavior `gen_server`.

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

Erl Scheme

The built in webserver httpd has three ways to create dynamic web pages CGI, Eval Scheme and Erl Scheme. All web based tools using the framework WebTool must use the Erl Scheme.

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

Assume that there are a module `mytool` and a function `startpage` that shall be called via the Erl Scheme. Assume also that the following line was in the configuration file that was used when starting the webserver:

`ErlScriptAlias /tools [mytool]`

Then the url will be:

http://myserver/tools/mytool/startpage

or in more general form:

http://servername:port/ErlScriptAlias/module/function

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

- Environment is a list of key/value tuples, see *mod_esi* for more information.
- Input is the part of the url after the ?, that is the part of the url containing name value pairs. If the page was called with the url:

http://myserver/tools/mytool/startpage?input1=one&input2=two

input will be the string `"input1=one&input2=two"` in the module `httpd` there is a function `parse_query` that parse the Input and returns a list of key/value tuples. See the INETS documentation for a more in depht coverage of the Erl Scheme.

A small example

A Hello World example that use the Erl Scheme would look like this.

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


helloWorld(Env,Input)->
    [header(),html_header(),helloWorldBody(),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".

helloWorldBody()->
```

```
        "<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, and a callback function that returns the data needed by WebTool to configure the tool must be added.

The callback function for this example will look like this:

```
configData()->
{testTool,
[{web_data,{"TestTool","/testtool/helloworld/helloWorld"}},
{alias,{erl_alias,"/testtool",["helloworld"]}}]}.
```

Create a *.tool file for the tool, the file must be in the current path. The file will look something like this.

```
{version,1.2}.
[{helloworld,configData,[]}].
```

Start WebTool by calling the function `webtool:start()`. Point your browser to the url *http://127.0.0.1:8888/*. 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

- Erlang Module **webtool** [page 6] – WebTool is a tool used to simplify the use of web based tools with Erlang/OTP.

webtool

The following functions are exported:

- `start()-> {ok,Pid}| {stop,Reason}`
  [page 6] Start WebTool.
- `start(Path,Data)->{ok,Pid}|{stop,Reason}`
  [page 6] Start WebTool.
- `stop()->void`
  [page 6] Stop WebTool.
- `Module:Func(Data)-> {Name,WebData}|error`
  [page 7] Returns configuration data needed by WebTool to configure and start a tool.

webtool

Erlang Module

WebTool makes it easy to use web based tools with Erlang/OTP. WebTool configure and start the webserver httpd. If WebTool not has write access to it's priv directory copy it to another location and use `webtool;start/2` to start WebTool.

Exports

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

Start WebTool with default, port 8888, ip-number 127.0.0.1, and server-name localhost. The configuration data is assumed to be in the directory `webtool-vsn/priv`.

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

- Path = String()|standard_path
- Data = [Port,Address,Name]|standard_Data|PortNumber
- Port = {"Port",Portnr}
- Address = {"BindAddress",Ipnumber}
- Name = {"ServerName",ServerName}
- Portnr = Ipnumber = ServerName = String()
- PortNumber = integer()
- Pid = pid()
- Reason

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

If `Data` is set to `PortNumber`, the ip-number and server- name will be retreived by calling `inet:gethostname/0` and inet:getaddr/2.

If `Path` is set to `standard_path` the configuration data is assumed to be in the directory `webtool-vsn/priv`. If `Data` is set to `standard_data` the default port ip-number and servername is used.

`stop()->void`

Stop WebTool and the tools started by WebTool.

CALLBACK FUNCTIONS

The following callback function must be implemented by each web based tool that will be used via WebTool. WebTool search the path for *.tool files to find the callback function, see WebTool user's Guide for more information.

Exports

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

- 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_spec()
- Func = {{StartMod, StartFunc, StartArg}, {StopMod, StopFunc, StopArg}}
- AppName = StartMod = SartFunc = StopMod = StopFunc =atom()
- StartArg = StopArg = [term()]
-

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 server.

for more information about `child_spec` see supervisor:start_child/3.

See Also

WebTool User's Guide.

Index of Modules and Functions

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