Users's Manual for Ada Web Server

Brent Seidel
Phoenix, AZ

August 6, 2024

Note that this is a draft version and not the final version for publication.

Contents

Chapter 1

Introduction

This Ada web server is a simple server designed to serve a few files and provide an interface to embedded systems. It is not intended to be a highly flexible high performance server–there are plenty of systems that provide that. It also does not currently support HTTPS or any authentication methods, so it should not be used in an application that needs security. However, it does not, by default refer to any external servers so it can be used on an isolated network.

This document describes the configuration of and modifications to the Ada web server. The amount of customization needed depends on the application. For some applications, no modifications of the software may be needed. In other applications, major modifications are needed.

1.1 License

Chapter 2

How to Obtain

This collection is currently available on GitHub at `https://github.com/BrentSeidel/Ada-Web-Server`

2.1 Dependencies

2.1.1 Ada Libraries

- `Ada.Characters.Latin_1`
- `Ada.Containers.Indefinite_Hashed_Maps`
- `Ada.Exceptions`
- `Ada.Sequential_IO`
- `Ada.Streams`
- `Ada.Strings`
- `Ada.Strings.Equal_Case_Insensitive`
- `Ada.Strings.Fixed`
- `Ada.Strings.Hash_Case_Insensitive`
- `Ada.Strings.Unbounded`
- `Ada.Text_IO`
- `Ada.Text_IO.Unbounded_IO`

2.1.2 Other Libraries

This library depends on the root package `BBS` available at `https://github.com/BrentSeidel/BBS-Ada` and through alire via "`alr get bbs`". Packages external to this project are marked with an asterisk.

- `BBS.web.files`
- `BBS.web.html`
- `BBS.web.http`
- `BBS.web`
- `GNAT.Sockets`*

Chapter 3

Usage Instructions

This is a library of routines intended to be used by some program. To use these in your program, edit your `*.gpr` file to include a line to `with` the path to `web_lib.gpr`. Then in your Ada code `with` in the package(s) you need and use the routines.

To use the example web server, go to the `example` directory and build the `web.gpr` project file in the `example` directory..

Chapter 4

API Description

4.1 Packages

4.1.1 `web.adb`

This is an example program that uses the web server library. It initializes the internal items table and starts the web server. If other initializations are needed, they could be added or called from here.

4.1.2 `BBS.web`

This package contains a number of items that are available for use in other packages. Items are put here instead of in `BBS.web.server` in order to avoid circular dependancies. The primary item from `BBS.web` that is expected to be used by user software is the `params` hash map. The package is instantiated as follows:

```
package params is new Ada.Containers.Indefinite_Hashed_Maps
      (Element_Type => String,
       Key_Type => String,
       Hash => Ada.Strings.Hash_Case_Insensitive,
       Equivalent_Keys => Ada.Strings.Equal_Case_Insensitive);
```

If your code needs access to parameters passed in a GET or POST request, this is how you will get access to those parameters. There are some examples where this is used in the `internal` and `svg` packages. Both parameters and headers use this hash map type.

Another hash map type is instantiated for internally generated items. First, every internal item procedure has the same signature so an access type (`user_proc` can be created as:

```
type user_proc is access procedure (s : GNAT.Sockets.Stream_Access;
                                    p : params.Map;
                                    h : params.Map);
```

The parameters are:

- `s` - The stream to write output to.

- `p` - Any passed parameters from the HTTP request
- `h` - HTTP request headers.

Then the hash map package is instantiated as:

```
package proc_tables is new Ada.Containers.Indefinite_Hashed_Maps
     (Element_Type => user_proc,
      Key_Type => String,
      Hash => Ada.Strings.Hash_Case_Insensitive,
      Equivalent_Keys => Ada.Strings.Equal_Case_Insensitive);
```

This map is populated and used in the `web_server` package.

4.1.3 `BBS.web.server`

This package contains the main web server software. Since this package calls the user code, any attempt by user code to call items in this package will create a circular dependency. Items that may be needed by user code should be in `web_common`. The item in this package that gets called with user code is the following procedure:

```
procedure server(internals : proc_tables.Map;
                 config_name : String;
                 port : GNAT.Sockets.Port_Type);
```

- `internals` - A map of the names of internal item and procedure
- `config_name` - The name of the configuration file
- `port` - The port to listen on
- This procedure enters an infinite loop and never returns.

This procedure should be called after all other initialization is done. It starts the web server and does not return. At some point, it may get turned into a task so that other processing in the main task can proceed along with it.

4.1.4 `BBS.web.html`

This package contains routines to support the generation of HTML.
Generate a simple HTML heading with the specified title.

```
procedure html_head(s : GNAT.Sockets.Stream_Access; title : String)
```

- `s` - The stream to write output to.
- `title` - The page title.

Generate a simple HTML heading with the specified title and style sheet.

```
procedure html_head(s : GNAT.Sockets.Stream_Access;
                    title : String; style : String)
```

- `s` - The stream to write output to.
- `title` - The page title.
- `style` - The URL for the style sheet to use.

Generate an ending for an HTML item using the file specified in `name`. This can be used to have a common footer for pages.

```
procedure html_end(s : GNAT.Sockets.Stream_Access; name: String)
```

- `s` - The stream to write output to.
- `name` - The name of the file containing the page ending.

4.1.5 `BBS.web.http`

This package contains routines to support HTTP. Currently GET and POST methods are supported with some minimal support for the OPTIONS method. GET methods are supported for all items while POST methods are only supported for internally generated items. If a file or an internally generated item with no parameters is being served, any supplied parameters are just ignored. Some of these routines are intended for use by user code and some are not.

Return code 200 OK for normal cases.

```
procedure ok(s : GNAT.Sockets.Stream_Access; txt: String)
```

- `s` - The stream to write output to.
- `txt` - Used as the MIME type for the page.

Return code 404 NOT FOUND for when the requested item is not in the directory.

```
procedure not_found(s : GNAT.Sockets.Stream_Access; item: String)
```

- `s` - The stream to write output to.
- `item` - The URL of the item that cannot be found.

Return code 500 INTERNAL SERVER ERROR generally when unable to open the file for the specified item. This means an item has been added to `config.txt` without adding the file to the system.

```
procedure internal_error(s : GNAT.Sockets.Stream_Access; file: String)
```

- `s` - The stream to write output to.
- `file` - The name of the file that caused the problem.

Return code 501 NOT IMPLEMENTED for a request for an internally generated item that is not yet implemented.

```
procedure not_implemented_int(s : GNAT.Sockets.Stream_Access; item: String)
```

- `s` - The stream to write output to.
- `item` - The URL of the item that is not implemented.

The `read_headers` procedure handles both GET, POST, and OPTIONS request and returns any passed parameters. Returned values will be the requested item in `item` and a dictionary containing the parameters in `params`. If there are no parameters, the dictionary will be empty. This routine is called by the web server to read the headers of the HTTP request. By the time any user code is called, the headers have already been read.

```
procedure read_headers(s       : GNAT.Sockets.Stream_Access;
                       sock    : GNAT.Sockets.Socket_Type;
                       method  : out request_type;
                       item    : out Ada.Strings.Unbounded.Unbounded_String;
                       headers : in out params.Map;
                       args    : in out params.Map;
                       dir     : dictionary.Map)
```

- `s` - The stream to read from.
- `sock` - The socket to read from.
- `method` - The requested method.
- `item` - The requested item.
- `headers` - A dictionary containing the rest of the request headers and values.
- `args` - A dictionary containing the request arguments and values.
- `dir` - A dictionary containing options to be returned for an OPTIONS request.

4.1.6 `BBS.web.internal`

This package contains routines to generate HTML or XML for internally generated items. The generated HTML or XML is written to the `Stream_Access` that needs to be passed in. The proper HTTP headers also need to be written. The routines in this package are only referenced (if referenced at all) from the internal items map. They do provide some examples of how to generate items. The internal routines can have access to parameters passed in the GET or POST request as well as the HTTP headers. If no parameters were passed, the parameters dictionary is empty.

Sends the count of transactions as an xml message.

```
procedure xml_count(s : GNAT.Sockets.Stream_Access;
                    h : params.Map;
                    p : params.Map);
```

- `s` - The stream to write output to.

- `h` - The dictionary containing the header items.
- `p` - The dictionary containing the argument items.

Sends the configuration data as a HTML table.

```
procedure html_show_config(s : GNAT.Sockets.Stream_Access;
                           h : params.Map;
                           p : params.Map);
```

- `s` - The stream to write output to.
- `h` - The dictionary containing the header items.
- `p` - The dictionary containing the argument items.

Sends the parameters provided as a HTML table.

```
procedure target(s : GNAT.Sockets.Stream_Access;
                 h : params.Map;
                 p : params.Map);
```

- `s` - The stream to write output to.
- `h` - The dictionary containing the header items.
- `p` - The dictionary containing the argument items.

Request that the configuration file be reloaded. This can be useful during development and debugging.

```
procedure html_reload_config(s : GNAT.Sockets.Stream_Access;
                             h : params.Map;
                             p : params.Map);
```

- `s` - The stream to write output to.
- `h` - The dictionary containing the header items.
- `p` - The dictionary containing the argument items.

4.1.7 `BBS.web.svg`

This package contains routines to generate SVG for internally generated items. The generated SVG is written to the `Stream_Access` that needs to be passed in. The proper HTTP headers also need to be written.

Send SVG code to display a thermometer showing the value parameter. This procedure handles getting and checking the parameters. The value is clamped to be between `min` and `max`. If any exceptions occur in parsing the parameters or if `min` is greater than `max`, a red "X" will be presented as the graphic to indicate an error condition. Using the default configuration, an example request is `/Thermometer?min=0&max=100&value=50`

```
procedure thermometer(s : GNAT.Sockets.Stream_Access;
                      h : params.Map;
                      p : params.Map);
```

- `s` - The stream to write output to.
- `h` - The dictionary containing the header items.
- `p` - The dictionary containing the argument items.

The following parameters are supported:

- `min` – The minimum displayed value
- `max` – The maximum displayed value
- `value` – The value to display.

Send SVG code to display a round dial with a pointer to the appropriate value. The value is clamped to be between `min` and `max`. If any exceptions occur in parsing the parameters or if `min` is greater than `max`, a red "X" will be presented as the graphic to indicate an error condition. Using the default configuration, an example request is `/Dial?min=0&max=100&value=50`

```
procedure dial(s : GNAT.Sockets.Stream_Access;
               h : params.Map;
               p : params.Map);
```

- `s` - The stream to write output to.
- `h` - The dictionary containing the header items.
- `p` - The dictionary containing the argument items.

The following parameters are supported:

- `min` – The minimum displayed value
- `max` – The maximum displayed value
- `value` – The value to display.

4.1.8 `BBS.web.files`

This package consists of a spec and body and is used to support the serving of binary and text files. The visible routines are:

This procedure sends a binary file to the client with headers. The file name is contained in the parameter `name` and the MIME type of the file is contained in the parameter `mime`.

```
procedure send_binary_with_headers(s : GNAT.Sockets.Stream_Access;
                                   mime : String; name : String)
```

- `s` - The stream to write output to.
- `name` - The name of the binary file to send.
- `mime` - The MIME type of the file.

This procedure sends a text file to the client with headers. The file name is contained in the parameter `name` and the MIME type of the file is contained in the parameter `mime`.

```
procedure send_text_with_headers(s : GNAT.Sockets.Stream_Access;
                                 mime : String; name : String)
```

- `s` - The stream to write output to.
- `name` - The name of the binary file to send.
- `mime` - The MIME type of the file.

This procedure sends a text file to the client without headers. The file name is contained in the parameter `name`. No HTTP headers are sent.

```
procedure send_text_without_headers(s : GNAT.Sockets.Stream_Access;
                                    name : String)
```

- `s` - The stream to write output to.
- `name` - The name of the binary file to send.

4.2 Modifications

To add or change internally generated items, the code will need to be modified and recompiled. In both cases, the first place to look in the software is the `build_internal_map` procedure in the `web.adb` example file.

4.2.1 Modifications to Existing Items

First, identify the routine to modify by looking at the `build_internal_map` procedure. Then the appropriate files can be edited and the routine located. Once the routine is located, any necessary modifications can be made.

4.2.2 Adding New Items

The very first thing is to decide what you can your item to do. The existing software has examples of routines that generate HTML, XML, and SVG. These can be used as models. These are contained in the `BBS.web.internal` (for HTML and XML) and `BBS.web.svg` (for SVG) packages. If you need to interface with other hardware or software, it will probably be best to add new packages for these interfaces.

Chapter 5

Example Program

5.1 `web.adb`

This is an example program that uses the web server library. It initializes the internal items table and starts the web server. If other initializations are needed, they could be added or called from here.

5.2 Configuration

Generally before being used, the web server needs to be configured. An example configuration is included and may be modified as needed.

5.3 Configuration File

The primary means of configuration is the `config.txt` file in the repository root. This file is used to translate from the URL requested to the actual page served. Both external files and internally generated responses are supported. Note that only URLs specified in this file will be served. Anything else will get a 404 NOT FOUND error.

The format of the configuration file is fairly simple. A line that has a pound sign, "#" (octothorp) as the first character is a comment. Non comment lines consist of three fields separated by a single space.

- The first field is the requested URL minus the server specification. Due to the nature of URLs, the first character will always be a slash, "/". The web server uses a simple dictionary data structure for the URLs so the structure is technically flat. However, any sort of hierarchical structure can be simulated.

- The second field identifies the item to be served. It may be a file or it may be an arbitrary string to identify an internally generated item. Files served are passed unchanged and both text and binary files are supported.

- The third field identifies the MIME type of the file being served. If the third field is "`internal`", the item being served will be generated internally. The value of the second field is used to select the proper internal routine to call.

5.3.1 Modifying the Software

To change the port used by the server, specify a different value when calling `BBS.web.server.server`.

To change the number of tasks (threads) available for serving, change the `num_handlers` constant in `BBS.web.server`. The default value is 10, which should be adequate. If memory is tight, it can be reduced. If higher performance is needed, this number can be increased.