Contents

HTTP Server

** *Sergey Linev* * GSI, Darmstadt *

Chapter 1

HTTP server in ROOT

Idea of such server – provide direct access to the different data from running ROOT application. Any object can be streamed at the moment when request is coming and delivered to the browser. Main benefit of such approach – one do not need to create any temporary ROOT files for such task.

1.1 Starting HTTP server

To start http server, at any time create instance of the **THttpServer** class like:

```
serv = new THttpServer("http:8080");
```

This will starts civetweb-based http server with http port 8080. Than one should be able to open address "http://localhost:8080" in any modern browser (IE, Firefox, Chrome, Opera) and browse objects, created in application. By default, server can access files, canvases and histograms via gROOT pointer. All such objects can be displayed with JSRootIO graphics.

At any time one could register other objects with the command:

```
TGraph* gr = new TGraph(10);
gr->SetName("gr1");
serv->Register("graphs/subfolder", gr);
```

If objects content is changing in the applicaion, one could enable monitoring flag in the browser - than objects view will be regularly updated.

1.2 Configuring user access

By default http server is open for anonymous access. One could restrict access to the server only for authenticated users. First of all, one should create password file, using **htdigest** utility.

```
[shell] htdigest -c .htdigest domain_name user_name
```

It is recommened not to use special symbols in domain or user names. Several users can be add to the ".htdigetst" file. When server started, following arguments should be specified:

```
root [0] new THttpServer("http:8080/none?auth_file=.htdigest&auth_domain=domain_name");
```

After that browser will automatically request to input name/password for domain "domain_name"

1.3 Using FastCGI interface

FastCGI is a protocol for interfacing interactive programs with a web server like Apache, lighttpd, Microsoft ISS and many others.

When starting THttpServer, one could specify:

```
serv = new THttpServer("fastcgi:9000");
```

Example of configuration file for lighttpd server is:

```
server.modules += ( "mod_fastcgi" )
fastcgi.server = (
  "/remote_scripts/" =>
    (( "host" => "192.168.1.11",
       "port" => 9000,
       "check-local" => "disable",
       "docroot" => "/"
    ))
)
```

In this case, to access running ROOT application, one should open following address in the browser: http://lighttpd_hostname/remote_

In fact, FastCGI interface can run in parallel to http server. One just call:

```
serv = new THttpServer("http:8080");
serv->CreateEngine("fastcgi:9000");
```

One could specify debug parameter to be able adjust FastCGI configuration on the web server:

```
serv->CreateEngine("fastcgi:9000/none?debug=1");
```

All user access will be ruled by web server - for the moment one cannot restrict with fastcgi engine.

1.4 Integrate with existing applications

In many practical cases no any changes of existing code is required. Opened files (and all objects inside), existing canvas and histograms automatically scanned by the server and will be available to the users. If necessary, any object can be registered directly to the server with **`THttpServer::Register()`** call.

Central point of integration - when and how THttpServer get access to data from running application. By default it is done during gSystem->ProcessEvents() call - THttpServer uses synchronous timer, which is activated every 100 ms. Such approach works perfectely when running macros in interactive ROOT shell.

If application runs in compiled code and does not contains gSystem->ProcessEvents() calls, two method are available.

1.4.1 Asynchrounous timer

First method is to configure asynchronous timer for the server like:

```
serv->SetTimer(100, kFALSE);
```

Than timer will be activated even without gSystem->ProcessEvents() method call. Main advantage of such method that application code can be used as it is. Disadvantage - there is no control when communication between server and application is performed. It could happen just in-between of **`TH1::Fill()`** call and histogram object may be incomplete.

1.4.2 Explicit call of THttpServer::ProcessRequests() method

Second method is preferable - one just insert in the application regular calls of the THttpServer::ProcessRequests() method. Like:

```
serv->ProcessRequests();
```

In such case one can fully disable timer of the server:

```
serv->SetTimer(0, kTRUE);
```

1.5 Data access from command shell

Big advantage of http protocol that it supported not only in web browsers but also in many other applications. One could directly use http requests to access ROOT objects and data members from any kind of scripts.

If one starts server and register object like:

```
serv = new THttpServer("http:8080");
TNamed* n1 = new TNamed("obj", "title");
serv->Register("subfolder", n1);
```

One could request JSON representation of such object with the command:

```
[shell] wget http://localhost:8080/Objects/subfolder/obj/root.json
```

Than representation will look like:

```
{
  "_typename" : "JSROOTIO.TNamed",
  "fUniqueID" : 0,
  "fBits" : 50331656,
  "fName" : "obj",
  "fTitle" : "title"
}
```

One could access also class members of object like:

```
[shell] wget http://localhost:8080/Objects/subfolder/obj/fTitle/root.json
```

Result will be: “title”.

If access to the server restricted with htdigest method, it is recommended to use **curl** program while only curl correctly supports htdigest method. Command will look like:

```
[shell] curl --user "accout:password" http://localhost:8080/Objects/subfolder/obj/fTitle/root.json --digest -
```

Following requests can be performed: * root.bin - 20-byte header and zipped binary TBuffer content * root.json - ROOT JSON representation for object and objects members * root.xml - ROOT XML representation * root.png - PNG image * root.gif - GIF image * root.jpeg - JPEG image

For images one could specify h (height), w (width) and opt (draw) options. Like:

```
http://localhost:8080/Files/hsimple.root/hpx/root.png?w=500&h=500&opt=lego1
```

1.6 Using JSRootIO for ROOT files display

To be done