THttpServer class in ROOT

Sergey Linev

Experiment Electronics, GSI, Darmstadt

Introduction

THttpServer class implements http server for arbitrary ROOT-based applications. It is based on *Civetweb* embeddable http server and provides direct access to all objects registered to the server. Support of *FastCGI* allows to integrate it with standard web servers like *Apache* or *lighttpd*.

Server provides access to objects, data members and collections in different formats: binary, JSON, XML. One also could execute object methods or commands registered to the server.

JavaScript ROOT used to implement generic user interface. With any modern web browsers one could list, display and monitor objects available on http server. Different possibilities are provided to integrate dynamic web elements into other HTML pages.

Simple to use

In many practical cases one just needs to create an instance of *THttpServer*:

```
root [0]  .x $ROOTSYS/tutorials/hsimple.C
root [1]  serv = new THttpServer("http:8080");
```

Objects like histograms or canvases from *gROOT* directory will be automatically visible to the server. At any time one could register objects directly:

```
root [2] gr = new TGraph(10);
root [3] gr->SetName("gr1");
root [4] serv->Register("/subfolder",gr);
```

Read-only (default) mode can be disabled:

```
root [5] serv->SetReadOnly(kFALSE);
```

Several examples are provided in $ROOTSYS/tutorials/http/ subfolder.

TBufferJSON class

TBufferJSON converts objects (or selected data members) into JSON (JavaScript Object Notation) format, which can be parsed by standard JavaScript methods. Handling of STL containers is implemented. Produced data can be directly used with JSROOT.

TBufferJSON allows to keep complex ROOT I/O completely on the server side. Any custom streamer can be equipped to work with *TBufferJSON* - see *TCanvas::Streamer()* for practical example.

```
{
  "_typename" : "TAttText",
  "fTextAngle" : 0,
  "fTextSize" : 5.0e-02,
  "fTextAlign" : 11,
  "fTextColor" : 1,
  "fTextFont" : 62
}
```

```
{
"_typename" : "TH1F",
"fUniqueID" : 0,
"fBits" : 50331656,
"fName" : "hpx",
"fTitle" : "This is the px distr
"fLineColor" : 602,
...
```

http://bit.ly/1xinYu7

Data monitoring

In standard interface for *THttpServer* regular update of objects drawing will be performed when monitoring flag is enabled. There is also special “draw.htm” page for each object, which can be integrated into arbitrary HTML page using iframe tag:

```
<iframe width="500" height="400"
  href="http://localhost:8080/Objects/hpx/
        draw.htm?monitoring=1000"></iframe>
```

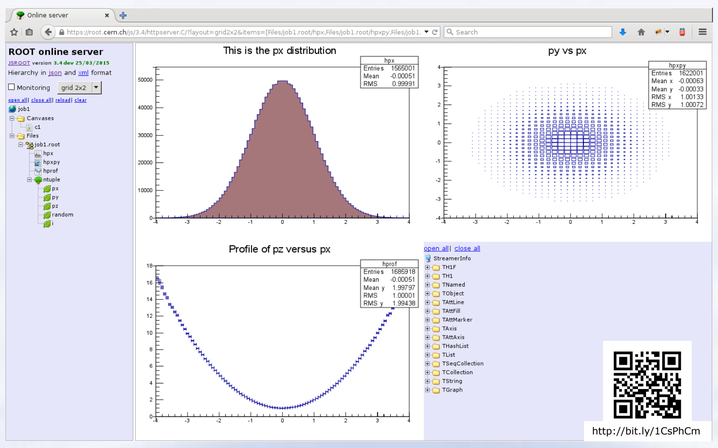
http://bit.ly/1yiepGM

One also can create and regularly update JSON files in arbitrary ROOT application. Such files can be served by normal web server and displayed with JavaScript ROOT like in demo:

```
https://root.cern.ch/js/3.4/demo/demo.htm
```

http://bit.ly/1EHnFXl

User interface



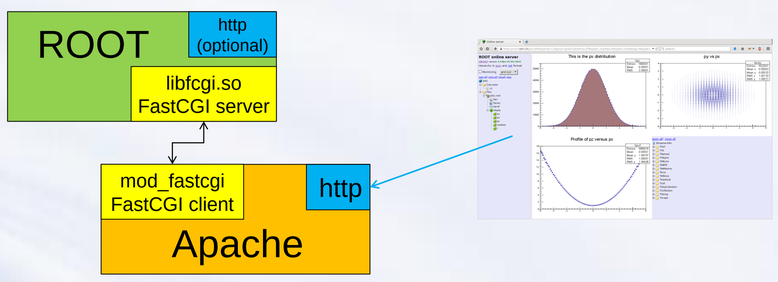
http://bit.ly/1CsPhCm

Snapshot of running *httpserver.C* macro from tutorials

FastCGI support

FastCGI is a protocol for interfacing interactive programs with standard web servers like *Apache*, *lighttpd*, *Microsoft IIS* and many others. It avoids complexity of http in ROOT and benefits from common web infrastructure: user management, access configuration, firewall. To start do:

```
root [6] serv->CreateEngine("fastcgi:9000");
```

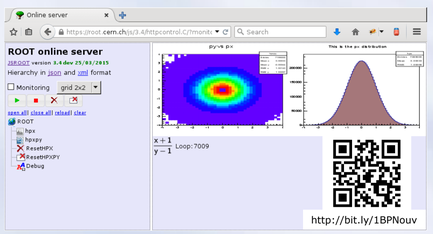


Command interface

One could register commands to the server:

```
root [7] serv->RegisterCommand("/Start", "bFillHist=kTRUE;");
root [8] serv->RegisterCommand("/ResetHPX","/hpx/->Reset();");
```

Commands appear in the browser hierarchy and can be executed by the user. One could use **cmd.json** request to invoke commands from the shell. Example can be found in *httpcontrol.C* macro.



http://bit.ly/1BPNouv

Snapshot of running *httpcontrol.C* macro from tutorials

http protocol to ROOT

THttpServer implements various http requests:

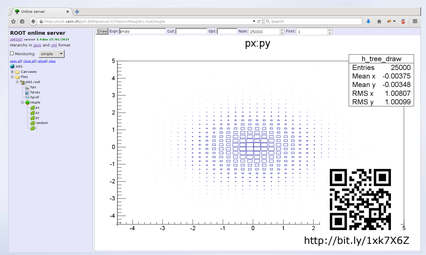
h.json	objects hierarchy description (*TRootSniffer*)
h.xml	objects hierarchy in XML
root.json	object data in JSON format (*TBufferJSON*)
root.bin	object data in binary format (*TBufferFile*)
root.xml	object data in XML format (*TBufferXML*)
root.png	object drawing on *TCanvas*
exe.json	objects method execution
cmd.json	execution registered to server commands

These requests are used to implement web GUI, but can be also invoked with http clients like ***wget*** or ***curl***

```
wget http://localhost:8080/hpx/exe.json?method=GetTitle
```

TTree::Draw in the browser

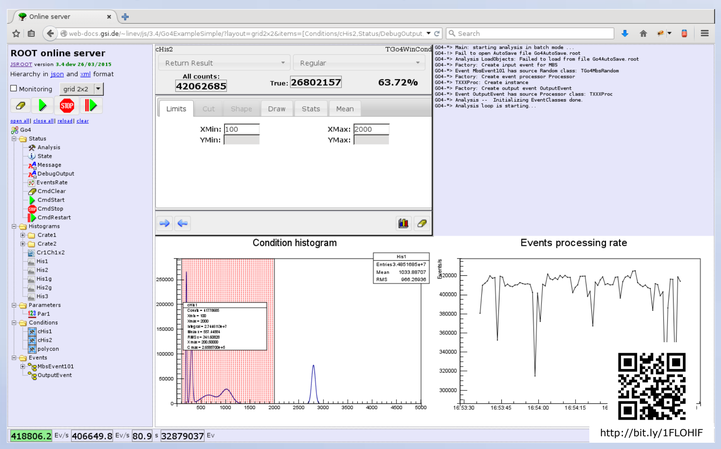
A special UI is provided allowing *TTree::Draw()* execution on remote ROOT process via http protocol. One could specify draw and cut expressions, number of events and see produced histogram in the browser. Here **exe.json** request is used, which returns histogram produced by *TTree::Draw* in JSON format.



http://bit.ly/1xk7X6Z

Use for go4 analysis

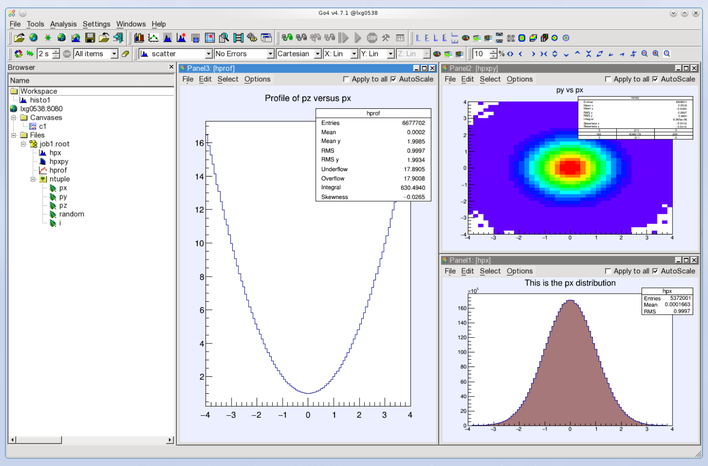
THttpServer is used in go4 framework (http://go4.gsi.de) to monitor and modify analysis objects. Drawing and objects editors for custom go4 classes are implemented, one is able to configure and control analysis remotely via http.



http://bit.ly/1FLOHIF

go4 GUI as browser

Based on ROOT and Qt, the go4 GUI implements a generic http client for *THttpServer*. One benefits from native ROOT graphics in case of web browser performance penalties.



Screenshot of go4 GUI connected to running *httpserver.C* macro

Conclusion

- Available in ROOT versions 5 & 6
- Provides generic remote user interface for ROOT applications
- Command-line tools can be used
- Many possibilities to integrate with existing ROOT-based applications
- Used in Go4 framework
- Documentation and demo on **https://root.cern.ch/js/**

CHEP 2015, 13-17 April 2015, Okinawa, Japan, http://chep2015.kek.jp/