SFML

* [Main Page](http://docs.google.com/index.htm)
* [Modules](http://docs.google.com/modules.htm)
* [Classes](http://docs.google.com/annotated.htm)
* [Files](http://docs.google.com/files.htm)
* [File List](http://docs.google.com/files.htm)
* [include](http://docs.google.com/dir_f3190241575fd2bd132a392ae6942f4a.htm)
* [SFML](http://docs.google.com/dir_692f376662c82a26cfe4cfa3aceebe24.htm)
* [Network](http://docs.google.com/dir_b9ac88db2949395b3130dd4ffb1be4e1.htm)

Http.hpp

1

2 //

3 // SFML - Simple and Fast Multimedia Library

4 // Copyright (C) 2007-2013 Laurent Gomila (laurent.gom@gmail.com)

5 //

6 // This software is provided 'as-is', without any express or implied warranty.

7 // In no event will the authors be held liable for any damages arising from the use of this software.

8 //

9 // Permission is granted to anyone to use this software for any purpose,

10 // including commercial applications, and to alter it and redistribute it freely,

11 // subject to the following restrictions:

12 //

13 // 1. The origin of this software must not be misrepresented;

14 // you must not claim that you wrote the original software.

15 // If you use this software in a product, an acknowledgment

16 // in the product documentation would be appreciated but is not required.

17 //

18 // 2. Altered source versions must be plainly marked as such,

19 // and must not be misrepresented as being the original software.

20 //

21 // 3. This notice may not be removed or altered from any source distribution.

22 //

24

25 #ifndef SFML\_HTTP\_HPP

26 #define SFML\_HTTP\_HPP

27

29 // Headers

31 #include <SFML/Network/Export.hpp>

32 #include <SFML/Network/IpAddress.hpp>

33 #include <SFML/Network/TcpSocket.hpp>

34 #include <SFML/System/NonCopyable.hpp>

35 #include <SFML/System/Time.hpp>

36 #include <map>

37 #include <string>

38

39

40 namespace sf

41 {

[46](http://docs.google.com/classsf_1_1Http.htm) class SFML\_NETWORK\_API [Http](http://docs.google.com/classsf_1_1Http.htm) : [NonCopyable](http://docs.google.com/classsf_1_1NonCopyable.htm)

47 {

48 public :

49

[54](http://docs.google.com/classsf_1_1Http_1_1Request.htm)  class SFML\_NETWORK\_API [Request](http://docs.google.com/classsf_1_1Http_1_1Request.htm)

55  {

56  public :

57

[62](http://docs.google.com/classsf_1_1Http_1_1Request.htm#a620f8bff6f43e1378f321bf53fbf5598)  enum [Method](http://docs.google.com/classsf_1_1Http_1_1Request.htm#a620f8bff6f43e1378f321bf53fbf5598)

63  {

[64](http://docs.google.com/classsf_1_1Http_1_1Request.htm#a620f8bff6f43e1378f321bf53fbf5598ab822baed393f3d0353621e5378b9fcb4)  [Get](http://docs.google.com/classsf_1_1Http_1_1Request.htm#a620f8bff6f43e1378f321bf53fbf5598ab822baed393f3d0353621e5378b9fcb4),

[65](http://docs.google.com/classsf_1_1Http_1_1Request.htm#a620f8bff6f43e1378f321bf53fbf5598ae8ec4048b9550f8d0747d4199603141a)  [Post](http://docs.google.com/classsf_1_1Http_1_1Request.htm#a620f8bff6f43e1378f321bf53fbf5598ae8ec4048b9550f8d0747d4199603141a),

[66](http://docs.google.com/classsf_1_1Http_1_1Request.htm#a620f8bff6f43e1378f321bf53fbf5598a4df23138be7ed60f47aba6548ba65e7b)  Head

67  };

68

80  [Request](http://docs.google.com/classsf_1_1Http_1_1Request.htm)(const std::string& uri = "/", Method method = Get, const std::string& body = "");

81

95  void setField(const std::string& field, const std::string& value);

96

107  void setMethod(Method method);

108

119  void setUri(const std::string& uri);

120

130  void setHttpVersion(unsigned int major, unsigned int minor);

131

142  void setBody(const std::string& body);

143

144  private :

145

146  friend class [Http](http://docs.google.com/classsf_1_1Http.htm);

147

157  std::string prepare() const;

158

169  bool hasField(const std::string& field) const;

170

172  // Types

174  typedef std::map<std::string, std::string> FieldTable;

175

177  // Member data

179  FieldTable m\_fields;

180  Method m\_method;

181  std::string m\_uri;

182  unsigned int m\_majorVersion;

183  unsigned int m\_minorVersion;

184  std::string m\_body;

185  };

186

[191](http://docs.google.com/classsf_1_1Http_1_1Response.htm)  class SFML\_NETWORK\_API [Response](http://docs.google.com/classsf_1_1Http_1_1Response.htm)

192  {

193  public :

194

[199](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8)  enum [Status](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8)

200  {

201  // 2xx: success

[202](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8a0158f932254d3f09647dd1f64bd43832)  Ok = 200,

[203](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8a0a6e8bafa9365a0ed10b8a9cbfd0649b)  Created = 201,

[204](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8ad328945457bd2f0d65107ba6b5ccd443)  Accepted = 202,

[205](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8aefde9e4abf5682dcd314d63143be42e0)  NoContent = 204,

[206](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8a77327cc2a5e34cc64030b322e61d12a8)  ResetContent = 205,

[207](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8a0cfae3ab0469b73dfddc54312a5e6a8a)  PartialContent = 206,

208

209  // 3xx: redirection

[210](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8add95cbd8fa27516821f763488557f96b)  MultipleChoices = 300,

[211](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8a2f91651db3a09628faf68cbcefa0810a)  MovedPermanently = 301,

[212](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8a05c50d7b17c844e0b909e5802d5f1587)  MovedTemporarily = 302,

[213](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8a060ebc3af266e6bfe045b89e298e2545)  NotModified = 304,

214

215  // 4xx: client error

[216](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8a3f88a714cf5483ee22f9051e5a3c080a)  BadRequest = 400,

[217](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8ab7a79b7bff50fb1902c19eecbb4e2a2d)  Unauthorized = 401,

[218](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8a64492842e823ebe12a85539b6b454986)  Forbidden = 403,

[219](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8affca8a8319a62d98bd3ef90ff5cfc030)  NotFound = 404,

[220](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8a12533d00093b190e6d4c0076577e2239)  RangeNotSatisfiable = 407,

221

222  // 5xx: server error

[223](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8adae2b2a936414349d55b4ed8c583fed1)  InternalServerError = 500,

[224](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8a6920ba06d7e2bcf0b325da23ee95ef68)  NotImplemented = 501,

[225](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8aad0cbad4cdaf448beb763e86bc1f747c)  BadGateway = 502,

[226](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8ac4fffba9d5ad4c14171a1bbe4f6adf87)  ServiceNotAvailable = 503,

[227](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8a215935d823ab44694709a184a71353b0)  GatewayTimeout = 504,

[228](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8aeb32a1a087d5fcf1a42663eb40c3c305)  VersionNotSupported = 505,

229

230  // 10xx: SFML custom codes

[231](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8a0af0090420e60bf54da4860749345c95)  InvalidResponse = 1000,

[232](http://docs.google.com/classsf_1_1Http_1_1Response.htm#a663e071978e30fbbeb20ed045be874d8a7f307376f13bdc06b24fc274ecd2aa60)  ConnectionFailed = 1001

233  };

234

241  [Response](http://docs.google.com/classsf_1_1Http_1_1Response.htm)();

242

255  const std::string& getField(const std::string& field) const;

256

268  Status getStatus() const;

269

278  unsigned int getMajorHttpVersion() const;

279

288  unsigned int getMinorHttpVersion() const;

289

302  const std::string& getBody() const;

303

304  private :

305

306  friend class [Http](http://docs.google.com/classsf_1_1Http.htm);

307

317  void parse(const std::string& data);

318

320  // Types

322  typedef std::map<std::string, std::string> FieldTable;

323

325  // Member data

327  FieldTable m\_fields;

328  Status m\_status;

329  unsigned int m\_majorVersion;

330  unsigned int m\_minorVersion;

331  std::string m\_body;

332  };

333

338  [Http](http://docs.google.com/classsf_1_1Http.htm)();

339

354  [Http](http://docs.google.com/classsf_1_1Http.htm)(const std::string& host, unsigned short port = 0);

355

371  void setHost(const std::string& host, unsigned short port = 0);

372

391  Response sendRequest(const Request& request, [Time](http://docs.google.com/classsf_1_1Time.htm) timeout = [Time::Zero](http://docs.google.com/classsf_1_1Time.htm#a8db127b632fa8da21550e7282af11fa0));

392

393 private :

394

396  // Member data

398  [TcpSocket](http://docs.google.com/classsf_1_1TcpSocket.htm) m\_connection;

399  [IpAddress](http://docs.google.com/classsf_1_1IpAddress.htm) m\_host;

400  std::string m\_hostName;

401  unsigned short m\_port;

402 };

403

404 } // namespace sf

405

406

407 #endif // SFML\_HTTP\_HPP

408

409

Copyright � Laurent Gomila  ::  Documentation generated by [doxygen](http://www.doxygen.org/)  ::