Subject: Core Http API change proposals - HttpStatus and HttpMethod as enum class
Posted by Klugier on Fri, 01 Jun 2018 23:29:05 GMT
View Forum Message <> Reply to Message

Hello,

I think U++ should provide better way of dealing with Http internals such as methods and status code. Currently method is implement using String and status code is implemented using regular int. It is not perfect, because the int can be out of rage of http status code. The same thing happens with method - it can be for example "WrongMethod" that doesn't exist in the standard.

My proposal for the change is to provide enum classes that will handle such problems:

```
// RFC compatibility
enum class HttpMethod {
    GET,
    HEAD,
    ...
};

enum class HttpStatus {
    ...
    BAD_REQUEST = 400,
    ...
    NOT_FOUND = 404,
    ...
};
```

We should consider changing current API or provide implicit conversion to int and String for compatibility reasons. The usage of the API will look like this (Basing on Mireks REST example):

```
void RESTProcess(TcpSocket& r)
{
 HttpHeader hdr;
 if(hdr.Read(r)) {
  VectorMap<String, String> urlvar;

  String uri = hdr.GetURI();
  int q = uri.Find('?');
   if(q >= 0) {
    ParseUrlVars(urlvar, ~uri + q + 1);
    uri.Trim(q);
   }

   String req, n;
   q = uri.Find('/');
   if(q >= 0) {
    n = uri.Mid(q + 1);
    req = Filter(n, CharFilterAlphaToLower);
   }

   switch (hdr.GetMethod()) { // Switch support is now possible...
                 case HttpMethod::GET:
                    if(req == "order") { }
                    break;
                 case HttpMethod::POST:
                    if(req == "order") { }
                    break;
                 case HttpMethod::PUT:
                    if(req == "order") { }
                    break;
                 case HttpMethod::DELETE:
                    if(req == "order") { }
                    break;
                 default:
                    break;
              }
 }
 HttpResponse(r, false, HttpStatus::BAD_REQUEST, "INVALID REQUEST");
}
```

General idea of this change is to make the API easier and safer to use for the library client.

Sincerely,
Klugier

Subject: Re: Core Http API change proposals - HttpStatus and HttpMethod as enum class
Posted by mirek on Sat, 16 Jun 2018 06:38:56 GMT
View Forum Message <> Reply to Message

I am afraid that HTTP is used much more loosely that you think. It is quite common for internal interfaces to introduce new method names and new error codes - that is why using String for method name is important.

Subject: Re: Core Http API change proposals - HttpStatus and HttpMethod as enum class
Posted by Klugier on Sun, 17 Jun 2018 19:02:45 GMT

Hello,

I agree with your, however I think we can still make it better for the final user. Instead of defining enum class we can define consts values. For example:

```
class HttpMethod {
public:
    static const String Get = "GET";
    static const String Head = "HEAD";

    HttpMethod() = delete;
};
```

The same thing we can do with Http status code:

```
class HttpStatus {
public:
    static const int Continue = 100;
    ...

    static const int OK = 200;


    String ToString(int status) {
        // Can returned string basing on status code.
    }

    HttpStatus() = delete;
};
```

I do not like the current approach when you need to explicitly write String in your code. This is more risk prone for the final user, because he can make spelling mistake in the code. In my approach it is verified on compilation level.

My idea is basing on following go standard library implementation:
- https://golang.org/src/net/http/method.go
- https://golang.org/src/net/http/status.go

Sincerely,
Klugier

Subject: Re: Core Http API change proposals - HttpStatus and HttpMethod as enum

class
Posted by mirek on Mon, 18 Jun 2018 05:38:34 GMT
View Forum Message <> Reply to Message

Klugier wrote on Sun, 17 June 2018 21:02Hello,

I agree with your, however I think we can still make it better for the final user. Instead of defining enum class we can define consts values. For example:

```
class HttpMethod {
public:
    static const String Get = "GET";
    static const String Head = "HEAD";

    HttpMethod() = delete;
};
```

The same thing we can do with Http status code:

```
class HttpStatus {
public:
    static const int Continue = 100;
    ...

    static const int OK = 200;


    String ToString(int status) {
        // Can returned string basing on status code.
    }

    HttpStatus() = delete;
};
```

I do not like the current approach when you need to explicitly write String in your code. This is more risk prone for the final user, because he can make spelling mistake in the code. In my approach it is verified on compilation level.

My idea is basing on following go standard library implementation:
- https://golang.org/src/net/http/method.go
- https://golang.org/src/net/http/status.go

Sincerely,
Klugier

With status codes, this is outright dangerous. With methods, not worth it. I doubt that users would prefer to write HttpMethod::Get instead "GET".

What might make sense, although barely, is to add "IsHEAD", "IsGET" methods to HttpHeader - at least we could add faster comparison (than operator==(String, const char *)), but it is still not worth it.