Subject: Re: Custom abort of HttpClient::Execute()
Posted by rylek on Thu, 23 Nov 2006 15:11:58 GMT
View Forum Message <> Reply to Message

You can abort the HTTP request using the progress callback

Gate2<int, int> progress

sent as one of the parameters to Execute. This callback is called periodically during all phases of the request (connection to server, writing request data, reading server response); as the name of the identifier implies, it's typically used to display progress bar.

In contrast to Callback2<int, int>, Gate2 has a return value. When you implement the progress callback, by returning true from the Gate function you tell the HttpClient to abort the request immediately.

A typical code snippet demonstrating its use within GUI context:

```
class DlgLoadURL : WithLoadURLLayout<TopWindow> {
public:
  typedef DlgLoadURL CLASSNAME;
  DlgLoadURL();

private:
  void Cancel();
  bool Progress(int pos, int size);

private:
  bool canceled;
};

DlgLoadURL::DlgLoadURL()
{
  CtrlLayoutOKCancel(...);
  cancel <<= THISBACK(Cancel);
}

DlgLoadURL::Cancel()
{
  canceled = true;
}

void DlgLoadURL::ExecuteHTTPRequest()
{
  canceled = false;
  HttpClient client;
  // fill in input request information
  client.Execute(THISBACK(Progress));
```

```
   // process result or error
}

bool DlgLoadURL::Progress(int pos, int size)
{
   // you can use pos & size to set up progress meter
   Ctrl::ProcessEvents();
   return canceled;
}
```

Note the call to Ctrl::ProcessEvents(); this is necessary to run the message loop during HTTP request execution (the outer message loop naturally doesn't run because from the point of view of the surrounding code the program is stuck within the HttpClient::Execute method). Without the message loop it wouldn't be possible to register user clicking on the Cancel button.

When the user clicks the Cancel button, the member variable 'canceled' is set to 'true'. The next time Progress is called, true is returned and the HTTP client terminates the request.

Practical remark: current version of the HTTP client calls the progress callback every few hudred milliseconds or so; when you use this method to cancel the pending request, expect an appropriate time lag. Getting rid of the lag would require either a multithreaded approach or marrying the socket interface with the Windows / Linux event loop, which is a direction we are still very reluctant to pursue.

Regards

Tomas
