Wt - Bug #7440

WTimer results in excessive browser CPU and memory usage when connection is lost

02/04/2020 01:19 PM - Christopher Bilberg

Status:	New	**Start date:**	02/04/2020
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

We are using a WTimer to trigger updates in our application, however when the connection to the server is lost the WTimer triggers the CPU and memory to explode.

If I run the following code:

```
#include <Wt/WApplication>
#include <Wt/WTimer>
#include <Wt/WText>

class TimerApplication : public Wt::WApplication
{
public:
    TimerApplication(const Wt::WEnvironment& env) : Wt::WApplication(env) {
      Wt::WText* text = new Wt::WText(root());
      Wt::WTimer* timer = new Wt::WTimer(this);
      timer->setInterval(10);
      timer->timeout().connect(std::bind([ = ]() {
        text->setText("Count is " + std::to_string(count++));
      }));
    timer->start();
    }
private:
  int count = 0;
};

Wt::WApplication *createApplication(const Wt::WEnvironment& env)
{
    return new TimerApplication(env);
}

int main(int argc, char **argv)
{
    return Wt::WRun(argc, argv, &createApplication);
}
```

The browser will for instance use around 20 % CPU when there is connection to the server, however as soon as the connection is lost, this goes up to 60-80 % and keeps increasing together with the memory usage. Eventually this will cause the whole browser to become unresponsive. The browser itself keeps printing net::ERR_CONNECTION_REFUSED errors, which makes sense, however I don't see why this basically has to make the entire browser freeze.

We are running Wt 3.4.1 and I used Google Chrome 79 for testing.

Is there a way to get around this? As for instance stop sending requests after it has been refused a certain number of times.

History

#1 - 02/05/2020 09:57 AM - Roel Standaert

10ms is of course a very short interval, but I trust that is just to demonstrate?

I think it could be valuable to have some sort of last resort handling of situations like this (server is not responding, and we're accumulating many events).

Specifically in your case, you may want to just use a single shot timer that is restarted every time it times out. That way there can never be more than one timeout event.

What I would generally recommend if you want to trigger updates (that originate from the server) is to use server push to do so, either when an event that has to update the interface happens, or by using a server side timer (WIOService::schedule or WServer::schedule (the latter appears to be currently undocumented) could be used for that).