Ruby master - Feature #6236

WEBrick::HTTPServer swallows Exception

03/31/2012 06:18 PM - regularfry (Alex Young)

Status:	Closed	
Priority:	Normal	
Assignee:	nahi (Hiroshi Nakamura)	
Target version:	2.0.0	

Description

At the moment when using WEBrick you've always got to remember to define a signal handler to be able to kill the server when you're done with it. This is annoying and makes it more painful to use than it should be, because if you realise you've forgotten to define a trap("INT") handler after you've started the server, all you can do is kill -9 the process. This also catches out people learning the library more than it should. It shouldn't be the web server's job to take over process management, but that's what it ends up doing.

The reason this happens is because webrick/server.rb uses rescue Exception around its accept loop. This is more broad than it should be. The attached patch replaces this with a rescue StandardError, and causes other Exception subclasses to be logged and re-raised. This makes WEBrick::HTTPServer somewhat more friendly to use at the command-line.

If you Ctrl-c out of a server.start loop with this patch applied, you can't restart the server because it leaves internal state lying around, but I think it's still an improvement over the current situation.

Associated revisions

Revision 8c5c5a22 - 04/11/2012 08:28 PM - drbrain (Eric Hodel)

- lib/webrick/server.rb (module WEBrick::GenericServer): A server will now continue only when a StandardError subclass is raised. For other exception types the error will be logged at the fatal level and the server will safely stop. Based on a patch by Alex Young. [ruby-trunk - Feature #6236]
- test/webrick/test_server.rb: Test for new exception handling behavior. Join the server thread instead of busy-waiting for it to shut down to remove race conditions.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@35303 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 35303 - 04/11/2012 08:28 PM - drbrain (Eric Hodel)

- lib/webrick/server.rb (module WEBrick::GenericServer): A server will now continue only when a StandardError subclass is raised. For other exception types the error will be logged at the fatal level and the server will safely stop. Based on a patch by Alex Young. [ruby-trunk - Feature #6236]
- test/webrick/test_server.rb: Test for new exception handling behavior. Join the server thread instead of busy-waiting for it to shut down to remove race conditions.

Revision 35303 - 04/11/2012 08:28 PM - drbrain (Eric Hodel)

- lib/webrick/server.rb (module WEBrick::GenericServer): A server will now continue only when a StandardError subclass is raised. For other exception types the error will be logged at the fatal level and the server will safely stop. Based on a patch by Alex Young. [ruby-trunk - Feature #6236]
- test/webrick/test_server.rb: Test for new exception handling behavior. Join the server thread instead of busy-waiting for it to shut down to remove race conditions.

Revision 35303 - 04/11/2012 08:28 PM - drbrain (Eric Hodel)

- lib/webrick/server.rb (module WEBrick::GenericServer): A server will now continue only when a StandardError subclass is raised. For other exception types the error will be logged at the fatal level and the server will safely stop. Based on a patch by Alex Young. [ruby-trunk - Feature #6236]
- test/webrick/test_server.rb: Test for new exception handling behavior. Join the server thread instead of busy-waiting for it to shut down to remove race conditions.

Revision 35303 - 04/11/2012 08:28 PM - drbrain (Eric Hodel)

- lib/webrick/server.rb (module WEBrick::GenericServer): A server will now continue only when a StandardError subclass is raised. For other exception types the error will be logged at the fatal level and the server will safely stop. Based on a patch by Alex Young. [ruby-trunk - Feature #6236]

- test/webrick/test_server.rb: Test for new exception handling behavior. Join the server thread instead of busy-waiting for it to shut down to remove race conditions.

Revision 35303 - 04/11/2012 08:28 PM - drbrain (Eric Hodel)

- lib/webrick/server.rb (module WEBrick::GenericServer): A server will now continue only when a StandardError subclass is raised. For other exception types the error will be logged at the fatal level and the server will safely stop. Based on a patch by Alex Young. [ruby-trunk - Feature #6236]
- test/webrick/test_server.rb: Test for new exception handling behavior. Join the server thread instead of busy-waiting for it to shut down to remove race conditions.

Revision 35303 - 04/11/2012 08:28 PM - drbrain (Eric Hodel)

- lib/webrick/server.rb (module WEBrick::GenericServer): A server will now continue only when a StandardError subclass is raised. For other exception types the error will be logged at the fatal level and the server will safely stop. Based on a patch by Alex Young. [ruby-trunk - Feature #6236]
- test/webrick/test_server.rb: Test for new exception handling behavior. Join the server thread instead of busy-waiting for it to shut down to remove race conditions.

Revision c26ea74a - 04/14/2012 02:30 AM - naruse (Yui NARUSE)

- lib/webrick/server.rb (WEBrick::GenericServer#start):
  partially revert r35315.

- test/webrick/test_server.rb (test_start_exception):
  received signal is delivered to the main thread, so it is needed to
  emulate it. patched by Eric Hodel. [ruby-core:44348] [Feature #6236]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@35323 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 35323 - 04/14/2012 02:30 AM - naruse (Yui NARUSE)

- lib/webrick/server.rb (WEBrick::GenericServer#start):
  partially revert r35315.

- test/webrick/test_server.rb (test_start_exception):
  received signal is delivered to the main thread, so it is needed to
  emulate it. patched by Eric Hodel. [ruby-core:44348] [Feature #6236]

Revision 35323 - 04/14/2012 02:30 AM - naruse (Yui NARUSE)

- lib/webrick/server.rb (WEBrick::GenericServer#start):
  partially revert r35315.

- test/webrick/test_server.rb (test_start_exception):
  received signal is delivered to the main thread, so it is needed to
  emulate it. patched by Eric Hodel. [ruby-core:44348] [Feature #6236]

Revision 35323 - 04/14/2012 02:30 AM - naruse (Yui NARUSE)

- lib/webrick/server.rb (WEBrick::GenericServer#start):
  partially revert r35315.

- test/webrick/test_server.rb (test_start_exception):
  received signal is delivered to the main thread, so it is needed to
  emulate it. patched by Eric Hodel. [ruby-core:44348] [Feature #6236]

Revision 35323 - 04/14/2012 02:30 AM - naruse (Yui NARUSE)

- lib/webrick/server.rb (WEBrick::GenericServer#start):
  partially revert r35315.

- test/webrick/test_server.rb (test_start_exception):
  received signal is delivered to the main thread, so it is needed to
  emulate it. patched by Eric Hodel. [ruby-core:44348] [Feature #6236]

Revision 35323 - 04/14/2012 02:30 AM - naruse (Yui NARUSE)

- lib/webrick/server.rb (WEBrick::GenericServer#start):
  partially revert r35315.

- test/webrick/test_server.rb (test_start_exception):
  received signal is delivered to the main thread, so it is needed to
  emulate it. patched by Eric Hodel. [ruby-core:44348] [Feature #6236]

Revision 35323 - 04/14/2012 02:30 AM - naruse (Yui NARUSE)

- lib/webrick/server.rb (WEBrick::GenericServer#start):
  partially revert r35315.

- test/webrick/test_server.rb (test_start_exception):
  received signal is delivered to the main thread, so it is needed to
  emulate it. patched by Eric Hodel. [ruby-core:44348] [Feature #6236]

History

#1 - 04/01/2012 12:24 AM - drbrain (Eric Hodel)

I like this idea, but why not rescue Interrupt separately from Exception instead? This preserves the exception-tolerance of the server while still allowing ^{C} to work

#2 - 04/02/2012 04:36 PM - regularfry (Alex Young)

I couldn't see that it was reasonable for WEBrick to expect to handle any of the Exception subclasses that aren't StandardErrors. This problem is caused precisely because WEBrick tried to handle something it shouldn't have, and I'd expect a similar problem with other Exceptions.

For this immediate problem, yes, rescue Interrupt would work just as well. I think it would be masking a wider issue. My rule of thumb is that you should rescue Exception precisely once in any given program, and *never* in a library. That being said, I'd be interested to see WEBrick code which expects to keep working under a rescue Exception but would break under rescue StandardError. It may just be that I'm insufficiently imaginative to see where it's useful :-)

#3 - 04/10/2012 02:32 PM - mame (Yusuke Endoh)

- Tracker changed from Bug to Feature

- Assignee set to drbrain (Eric Hodel)

This is not a bug, is it? Moving to the feature tracker.

And there is no maintainer of WEBrick. I tentatively assign this to drbrain.
Drbrain, if you think the patch is good enough, and if there is no objection, I think you can commit it.

--
Yusuke Endoh mame@tsg.ne.jp

#4 - 04/12/2012 05:28 AM - drbrain (Eric Hodel)

- Status changed from Open to Closed

- % Done changed from 0 to 100

This issue was solved with changeset r35303.
Alex, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- lib/webrick/server.rb (module WEBrick::GenericServer): A server will now continue only when a StandardError subclass is raised. For other exception types the error will be logged at the fatal level and the server will safely stop. Based on a patch by Alex Young. [ruby-trunk - Feature #6236]
- test/webrick/test_server.rb: Test for new exception handling behavior. Join the server thread instead of busy-waiting for it to shut down to remove race conditions.

#5 - 04/12/2012 05:31 AM - drbrain (Eric Hodel)

- Target version set to 2.0.0

I extended Alex's patch to include safe shutdown and to allow ^{C} to work from within accept_client.

Since each connection is handled in a separate thread, my concern about exception-tollerance was not founded.

#6 - 04/12/2012 06:00 AM - nahi (Hiroshi Nakamura)

- Assignee changed from drbrain (Eric Hodel) to nahi (Hiroshi Nakamura)

WEBrick has a maintainer now :)

#7 - 04/12/2012 07:53 AM - mame (Yusuke Endoh)

Oops, sorry. Nahi-san, could you please review drbrain's commit (r35303) ?

2012/4/12 nahi (Hiroshi Nakamura) nakahiro@gmail.com:

> Issue #6236 has been updated by nahi (Hiroshi Nakamura).
>
> Assignee changed from drbrain (Eric Hodel) to nahi (Hiroshi Nakamura)
>
> WEBrick has a maintainer now :)

--
Yusuke Endoh mame@tsg.ne.jp

#8 - 04/12/2012 10:53 AM - naruse (Yui NARUSE)

- Status changed from Closed to Assigned

r35303 looks buggy.

WERBrick::GenericServer#stop's is a method to deny a new connection.
Even if the method is called, the server is still alive.
To kill the server, calling shutdown is required.

In other words, the status of a server is changing like
Stop -> Running -> Shutdown -> Stop
(the name and the functions is confusing arround shutdown/stop...)

The relation between methods and status is

- start: Stop -> Running
- stop: Running -> Shutdown
- shutdown: Running -> Shutdown -> Stop

This effects especially when ServerType is Thread.
In such case it must wait after calling shutdown until the server status become Stop.

A patch is following:

diff --git a/lib/webrick/server.rb b/lib/webrick/server.rb

index 22b91ad..2eabf5d 100644
--- a/lib/webrick/server.rb
+++ b/lib/webrick/server.rb
@@ -143,6 +143,7 @@ module WEBrick
end

```
     ensure
```

- [redacted]
  ```
     @logger.info "going to shutdown ..."
     thgroup.list.each{|th| th.join if th[:WEBrickThread] }
     call_callback(:StopCallback)
  ```

  @@ -157,7 +158,7 @@ module WEBrick

  def stop
  if @status == :Running

- [redacted]

- [redacted]

  end
  end

#9 - 04/12/2012 06:23 PM - regularfry (Alex Young)

On 11/04/12 21:31, drbrain (Eric Hodel) wrote:

> Issue #6236 has been updated by drbrain (Eric Hodel).
>
> Target version set to 2.0.0
>
> I extended Alex's patch to include safe shutdown and to allow ^{C} to work from within accept_client.

Fantastic, thanks for that.

--
Alex

#10 - 04/13/2012 05:25 AM - drbrain (Eric Hodel)

=begin
I misunderstood the intended use of start/stop/shutdown, so I think your patch should be applied.

I assumed you could start the server again after stopping it like:

start -> stop -> start -> stop -> shutdown

since stop does not close the listening sockets, so having the :Shutdown state made no sense.

I changed :Shutdown to :Stop in GenericServer#stop because my test would hang waiting for the server to reach :Stop state after stop was called which would not bring the server to the :Stop state (due to a swallowed Exception). I changed TESTWEBrick::start_server to use Thread.join instead of a busy-loop this will no longer be a problem.
=end

#11 - 04/13/2012 05:01 PM - naruse (Yui NARUSE)

- Status changed from Assigned to Closed

This ticket looks intended to allow ^{C} shutting down the server.
So current change to shutdown other than StandardError is too wide, it should be only Interrupt.

drbrain (Eric Hodel) wrote:

> I misunderstood the intended use of start/stop/shutdown, so I think your patch should be applied.
>
> I assumed you could start the server again after stopping it like:
>
> start -> stop -> start -> stop -> shutdown
>
> since stop does not close the listening sockets, so having the :Shutdown state made no sense.

> I changed :Shutdown to :Stop in GenericServer#stop because my test would hang waiting for the server to reach :Stop state after stop was called which would not bring the server to the :Stop state (due to a swallowed Exception). I changed TESTWEBrick::start_server to use Thread.join instead of a busy-loop this will no longer be a problem.

Using Thread.join itself is not wrong, but on such case the change will break existing code.
Anyway I fixed it in r35315.

#12 - 04/13/2012 05:53 PM - regularfry (Alex Young)

On 13/04/12 09:01, naruse (Yui NARUSE) wrote:

> Issue #6236 has been updated by naruse (Yui NARUSE).
>
> Status changed from Assigned to Closed
>
> This ticket looks intended to allow ^{C} shutting down the server.
> So current change to shutdown other than StandardError is too wide, it should be only Interrupt.

You've reverted to catching non-StandardError Exceptions. Do you
disagree with my argument here?

> I couldn't see that it was reasonable for WEBrick to expect to handle any of the Exception subclasses that aren't StandardErrors. This problem is caused precisely because WEBrick tried to handle something it shouldn't have, and I'd expect a similar problem with other Exceptions.
>
> For this immediate problem, yes, rescue Interrupt would work just as well. I think it would be masking a wider issue.

If so, why?

--
Alex

#13 - 04/14/2012 07:50 AM - drbrain (Eric Hodel)

- File webrick.signal_exception.patch added

Since process-management libraries use other signals than SIGINT to terminate processes, such as SIGTERM, I think WEBrick should rescue and exit on any signal, not just Interrupt, so SignalException is better. See the attached patch.

Sending SIGINT to $$ raises an exception in the main thread (the test thread) not the WEBrick server thread which does not exercise the same code path. The test should exercise that WEBrick shuts down correctly upon ^{C,} not that assert_raises catches signal exceptions on the main thread and that the ensure in TestWEBrick#start_server works.

Alex, at r35315 besides NoMemoryError and SignalException, only StandardError subclasses can be raised from the WEBrick accept loop as no user code is run in the accept loop. If, for example, user code causes a SystemStackError, that error will only terminate the connection thread and will not propagate to the WEBrick server thread.

#14 - 04/14/2012 11:21 AM - naruse (Yui NARUSE)

- Status changed from Closed to Assigned

regularfry (Alex Young) wrote:

> On 13/04/12 09:01, naruse (Yui NARUSE) wrote:
>
> > Issue #6236 has been updated by naruse (Yui NARUSE).
> >
> > Status changed from Assigned to Closed
> >
> > This ticket looks intended to allow ^{C} shutting down the server.
> > So current change to shutdown other than StandardError is too wide, it should be only Interrupt.
>
> You've reverted to catching non-StandardError Exceptions. Do you
> disagree with my argument here?

Oh sorry, I missed it and now I agree.
I'll revert it.

> Sending SIGINT to $$ raises an exception in the main thread (the test thread) not the WEBrick server thread which does not exercise the same code path. The test should exercise that WEBrick shuts down correctly upon ^{C,} not that assert_raises catches signal exceptions on the main

thread and that the ensure in TestWEBrick#start_server works.

I'll merge your patch.

#15 - 04/14/2012 11:30 AM - naruse (Yui NARUSE)

- Status changed from Assigned to Closed

This issue was solved with changeset r35323.
Alex, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- lib/webrick/server.rb (WEBrick::GenericServer#start):
  partially revert r35315.

- test/webrick/test_server.rb (test_start_exception):
  received signal is delivered to the main thread, so it is needed to
  emulate it. patched by Eric Hodel. [ruby-core:44348] [Feature #6236]

Files

standarderror.patch	626 Bytes	03/31/2012	regularfry (Alex Young)
webrick.signal_exception.patch	1.26 KB	04/14/2012	drbrain (Eric Hodel)