ToiletPaper #143

Actor System for Python

Author: Hannes Lerchl / Senior Software Architect / New Business

✗ Problem

Actor Systems have been around for quite a while (emerged in the 1970s in the context of AI research) and provide a reliable method for highly parallel program execution. The best-known systems today are probably Erlang, Elixir, and Akka. Oh, how I wish that there would be such a system to handle the more and more frequent Python back-end requests…

✓ Solution

Look over there! In fact, there is an Actor System for Python: Thespian. A mature and well-documented library that provides a reasonable base:

- “Actor spawning” and “message passing” is taken over.
- Timeouts can be realized via delayed messages.
- In the background, a separate OS process is created for each actor. This makes it somewhat heavier, but as a side effect, the GIL is bypassed.
- Actors can be stopped by message (this also stops the process).
- Log messages are collected and merged.
- With the `ActorTypeDispatcher` there is a base class, which sorts the received messages by type and calls specialized methods (which need to be written).
- The system can also “monitor” file descriptors and sockets (via `select`) without blocking the whole actor.
- There is a “dead letter handling” for messages with disappeared recipients.
- There is a feedback (as a message), if the own message has caused another actor to stumble (i.e.: exception)
- You can register actors globally under a name (with the usual side effects).
- A feature called “actor troupes” allows horizontal scaling of actors.
- Shell commands can be executed by an actor and the output can be reported (wrapper around `subprocess.Popen`).

➔ Example

```python
from thespian.actors import *

class Hello(Actor):
  def receiveMessage(self, message, sender):
    # a message could be anything that python can pickle/unpickle
    self.send(sender, 'Hello, World!')

    # the actor keeps running and listening for messages

if __name__ == "__main__":
  # the actor system is created as well as a new actor.
  # this actor will run as its own OS-process on the same machine
  hello = ActorSystem().createActor(Hello)

  # send a message to the actor and wait (1 second) for a response
  print(ActorSystem().ask(hello, 'hi', 1))

  # command the actor to exit
  ActorSystem().tell(hello, ActorExitRequest())
```

+ Further Aspects

- What is this guy talking about? The actor model!
- Documentation can be found in the Thespian user's guide.
- When Carl Hewitt invented the actor model with his typewriter: “Viewing control structures as patterns of passing messages”.
- What’s missing is an ecosystem for “generic patterns” like Erlang/OTP offers.