Ruby 101

Advanced Ruby: Threads

Advanced Ruby:

Threads provide:

concurrent processing

ability to execute more than one thing "at once"*

*Allow multiple sections of code to get a share of processing time

Uses:

One task will take awhile, but others can continue

ex Loading a file into memory, while displaying progress

ex Processing one chunk of data, while working with another

ex Processing a large amount of data, while displaying the results as they come in

Advanced Ruby:

Create a thread

```
thread_name = Thread.new do
      …
end
```

Advanced Ruby:

Threads can access:

- variables
- methods
- classes
- etc...

Threads can have their own:

- variables
- methods
- classes
- etc ...

Advanced Ruby:

When working with concurrent threads:

A thread likely will take a break from it's processing:

sleep *seconds* => puts the thread to sleep for seconds
may or may not be at the time that another thread needs to process

Thread.stop => puts the thread to sleep until it is manually re-started

Thread.pass => lets the scheduler know that its a good time to switch threads

.wakeup => marks thread as ready for scheduling

.run => marks thread and invokes the scheduler

.join => pauses the current thread and executes the thread instance until exit of optional limit time elapses

.exit => terminates the thread

Advanced Ruby:

Thread with sleep:

```
my_thread = Thread.new do
    while true do
        puts "thread here"
        sleep 0.1
    end
end
time = 0
while time < 30 do
    puts "main thread here"
    sleep 1
    time += 1
end
```

Advanced Ruby:

Thread with stop and run:

```
my_thread = Thread.new do
    while true do
        puts "thread here"
        Thread.stop
    end
end
time = 0
while time < 30 do
    puts "main thread here"
    my_thread.run
    sleep 1
    time += 1
end
```

Advanced Ruby:

Exceptions in threads

By Default => Terminate thread only

.abort_on_exception= => (class and instance method)

if set to true, all threads exit on exception

if false only the thread where it occurred

$DEBUG = true

Advanced Ruby:

Accessing data outside the thread: BEWARE!

```
my_var = ""

my_thread = Thread.new do
    10.times do
        my_var += "tock"
        Thread.pass
    end
end

10.times do
    my_var += "tick"
    puts "Value: #{my_var}"
    Thread.pass
end
```

Advanced Ruby:

What state is my thread?

.status

"sleep"	=> the thread is asleep
"run"	=> the thread is current running
"aborting"	=> the thread has been signalled to exit
false	=> the thread has existed normally
nil	=> the thread has exited abnormally

Advanced Ruby:

Thread Local Variables
.thread_variables
.thread_variable_get
.thread_variable_set

Thread.current => the current thread object

```
thread = Thread.new do
    Thread.current.thread_variable_set "my_var", 1
    Thread.stop
    Thread.current.thread_variable_set "my_var", 2
end

puts thread.thread_variable_get "my_var"
thread.run
puts thread.thread_variable_get "my_var"
```