APIT Lab Week 6

Dr Simon Rogers

Jan 2018

Introduction and aims

To use a `SwingWorker` object to create a countdown timer.

Tasks

- Create a `JFrame` including a `JPanel` with a single `JTextField` and a `Start` button.
- When the user clicks on the `Start` button, your code should create and start (using `.execute()`) a new `SwingWorker` object that converts the value in the `JTextField` into an integer (don't worry about error checking) and counts down that number of Seconds.
- Some help:
  - The `SwingWorker` object (or more correctly, the object you make that will inherit from `SwingWorker`) should take as an input (via its constructor) the number of seconds. It should then call `Thread.sleep(1000)` for each of these seconds.
  - As it runs it should update the value in the text field (i.e. 10...9...8...7...etc). Whilst you *can* do this directly from within the `doInBackground()` method of the `SwingWorker` it is bad to do so! The correct way is to call `publish` each second and then write a `process` method that does the `JTextField` update.
  - Think before you start about what needs to know about what...i.e. in order to be able to order an update on the `JTextField` the `SwingWorker` needs a reference to it (or to a method you write in the `JFrame` class that does it (better)).
- Additional tasks:
  - Add a `Stop` button that stops the countdown. Note that `SwingWorkers` have a `.cancel()` method and, from within, they can check `isCancelled()`.
  - Wrap all the timer stuff (i.e. the JPanel etc) into a single class (making a class that extends `JPanel` is probably the neatest way). You can now easily add multiple timers to your `JFrame`. If you like, you can stick a button on your `JFrame` which, when clicked, adds another counter to the `JFrame`. Note that once you have made the `JFrame` visible, if you add anything you wont see it until you call `revalidate()` followed by `repaint()`.