{:title “Creating JFrames using Swing APIs in Clojure (Part 1)” :layout :post :tags [“Clojure Guide Code”]}

Swing is a powerful java library and also at times be complex however thanks to Dave Ray's seesaw which you should look at if you plan on doing any GUI building with Clojure. I suggest reading the tutorial as well to get used to the basics. But here I want to show you some things that aren't quite mentioned specifically in the tutorial. So lets create a new leiningen project.

```
$ lein new hello-swing
```

This will create the leiningen project and now add the seesaw library to your projects.clj. Edit your project.clj file to include Seesaw. As well as adding where the main function will reside which will be in the hello-swing.core.

```
(defproject hello-swing "0.1.0-SNAPSHOT"
  :description "FIXME: write description"
  :url "http://example.com/FIXME"
  :license {:name "Eclipse Public License"
            :url "http://www.eclipse.org/legal/epl-v10.html"}
  :dependencies [[org.clojure/clojure "1.8.0"]
                 [seesaw "1.4.2"]] ;; add this
  :main hello-swing.core) ;; add this
```

Afterwards let start writing into our core.clj which is in the /src directory. Set up your core.clj as the following.

```
(ns hello-swing.core
  (:gen-class)) ;; In case you want to make this into a jar.

(use 'seesaw.core) ;; allows you to use the seesaw library. However highly discouraged in pr

(def jframe (frame :title "hello Frame"
                   :height 300
                   :width 300
                   :on-close :exit ;; Exits on close you can also set it to :hide which will
                   :content (label :text "Hello this is a label!"))) ;; A simple label with

(defn -main []
  (native!)     ;; Makes the frame appear correctly based on the OS. Which should be called
  (invoke-later ;; executes the body in the near future.
   (-> jframe show!))) ;; displays the jframe.
```

Now you can run the code by entering the following in the project directory.

```
$ lein run
```

You should see a frame like this.

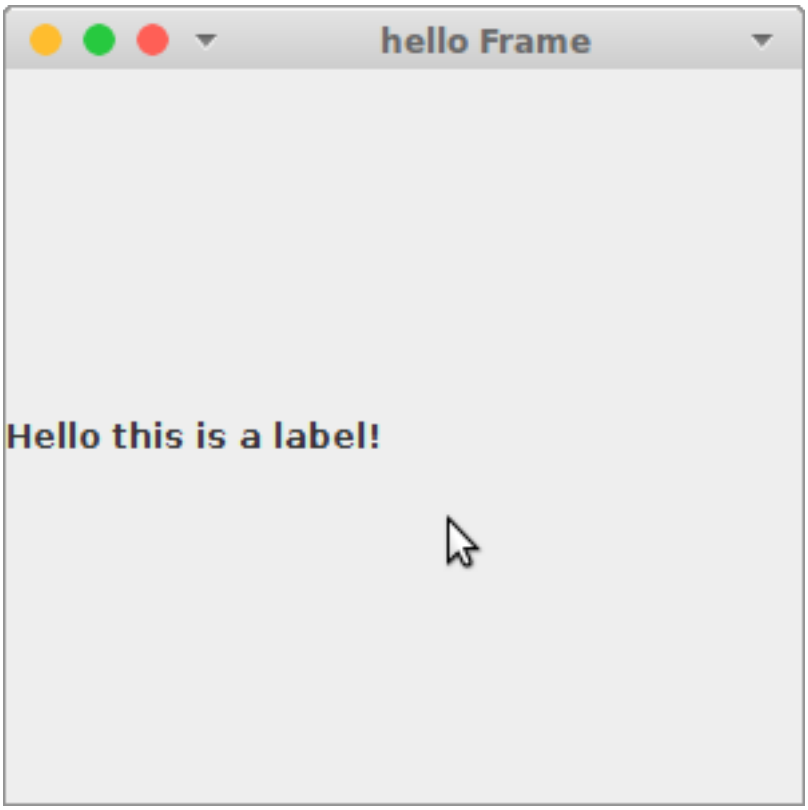

Figure 1: jframe

Now that you made a frame pop up lets decorate the frame to be a little nicer.
Adding a menubar to close it.

```
(ns hello-swing.core
  (:gen-class))

(use 'seesaw.core)

(defn handler [event]  ;; Handler that will be listening to the action commands being recie
  (let [e (.getActionCommand event)]
    (if (= e "Close Frame")
      (System/exit 0)))) ;; Exits the program with a exit status of 0.
```

```
(def close-frame (menu-item :text "Close Frame"
                            :tip "This will close the frame."
                            :listen [:action handler]))  ;; Assigning a function that will h

(def jframe (frame :title "hello Frame"
                   :menubar (menubar :items [(menu :text "File" :items [close-frame])]) ;; a
                   :height 300
                   :width 300
                   :on-close :exit ;; Exits on close
                   :content (label :text "Hello this is a label!"))) ;; Label

(defn -main []
  (native!)
  (invoke-later
   (-> jframe show!)))
```

I think that'll be all for today since that in itself is a lot specially if you never worked with GUI's before. You can mess around and make the menubar do all kinds of things just by changing the (System/exit 0) part.

In the next part we will be doing themes.