Example: A ``Hello World'' application

This section contains a step-by-step example how to create a ``stand-allone'' application using Ltk and SBCL. Fist of all, you need the application itself:

```
(defpackage :hello-world
  (:use :common-lisp :ltk)
  (:export #:main))

(in-package :hello-world)

(defun main ()
  (setf *debug-tk* nil)
  (with-ltk ()
    (let ((b (make-instance
              'button
              :text "Hello World!"
              :command (lambda ()
                         (do-msg "Bye!" "Hello World!")
                         (setf *exit-mainloop* t)))))
      (pack b))))
```

This may not be the worlds greatest application, but I shows the important steps. First rule is: whenever you write lisp code, put it in a package. While this seems overkill, it is the easiest solution to avoid symbol conflicts, and if your code grows you will need a package anyway.

Next, you want to build your application, here is a shell-script that will do that work:

```
sbcl --eval "(progn
               (compile-file \"ltk\")
               (load \"ltk\")
               (compile-file \"hello-world\")
               (load \"hello-world\")
               (save-lisp-and-die \"hello-world.core\"))"
```

This script compiles and loads both `ltk` and `hello-world`. Then it calls `save-lisp-and-die` to create the core file for the application. Put it in a file called`build-hello`, make it executable and run it to build the application. Once you have build your core, all what is left is running the application. For that, a small startup script will create the ``executable'' feeling:

```
sbcl --core hello-world.core --noinform\
     --eval "(progn (hello-world:main) (quit))"
```

If you put it in a file called `helloworld`, and make it executable, you can start your application just by typing `helloworld` at the shell prompt.

To deliver your application, you need to provide three files: `sbcl`, `hello-world.core` and `helloworld`. `sbcl` is the sbcl launching program, you can find its location by typing `which sbcl`. It is less than 300k in size, so just copy and deliver it with your custom core and startup scipt.

Thats all :).