Singletons in Haskell

If you are deep into Haskell you've probably come across the singletons library. You may have shied away, because it looked scary, but really the main concept is actually quite straight-forward.

TIP	If you are willing to spend a bit of time to understand the ins and outs of this library, you should read this paper by Richard A. Eisenberg and Stephanie Weirich.

Why does this library exist?

Haskell doesn't have first class dependent types. This library still allows you to write dependent functions.

The main idea

A singleton type is a type with exactly one constructor.

To write dependent functions in Haskell you have to use the singleton type to ...

1. promote a value to the type level.
2. demote a type-level constructor to value-level.

IMPORTANT	Now look at this table!

Table 1. Singleton for `Succ Nat`

kinds		promoted kind: `Nat`	
types	regular ADT: `Nat`	type-level constructor: `'Succ 'Zero`	singleton type: `SNat ('Succ 'Zero)`
terms	regular value constructor: `Succ Zero`		singleton constructor: `SSucc SZero`

The singleton type allows you to convert between types and terms.

TIP	If you understood this quite well, you have already grasped the main point of this library. If you want to actually use the library to write Dependent Haskell, I once again recommend this paper to get a bit more comfortable with the matter.

Why do we need a library, if the concept is so simple?

With the singletons library you can use *TemplateHaskell* to generate the singleton types for you.

CAUTION	Writing the singletons manually is really quite annoying!

It also gives names to common idioms, so it will actually help you to gain an understanding for dependent types in Haskell.