Registry

Janko Thyson

Thursday, October 29, 2014

```
## [1] TRUE
```

Whenever a reactive object is set via `setReactiveS3()`, the object that is *actually* created is an instance of class `ReactiveObject.S3` even though it does not appear so: the instance itself remains invisible and only the value of its field `value` is made visible to the user/system and can thus subsequently be accessed and manipulated through the name/ID provided in `id` in the call to `setReactiveS3()`

```
setReactiveS3(id = "x_1", 10)
x_1
```

```
## [1] 10
```

```
class(x_1)
```

```
## [1] "numeric"
```

Usually, due to the way `makeActiveBinding()` works, we would not be able to access this hidden object once the function returns as it is only stored internally. In order to keep the object accessible, `setReactiveS3()` stores it in an internal registry.

The registry can be accessed via

```
getOption("reactr")$.registry
```

```
## <environment: 0x0000000007c541d0>
```

or via the convenience function `getRegistry()`

```
registry <- getRegistry()
```

As mentioned, the actual content consists of the respective **invisible** objects that were created in the call to `setReactiveS3()` which are assigned to names that correspond to the UIDs of its **visible** parts

```
ls(registry)
```

```
## [1] "2fc2e352f72008b90a112f096cd2d029"
```

```
x_1_hidden <- getFromRegistry(id = "x_1")
x_1_hidden
```

```
## <environment: 0x0000000009b89230>
## attr(,"class")
## [1] "ReactiveObject.S3" "environment"
```

```r
class(x_1_hidden)
```

```
## [1] "ReactiveObject.S3" "environment"
```