written by Tom McAdam on 23 October 2012 in engineering, MetaBroadcast

optional maps

Some time before I went on holiday I wrote about how we use Optionals in our codebase. Today I'm going to briefly write about a recent addition Fred made which has Optional and Map playing nicely together.

the problem

We wanted to to have a `Map<K, Optional<V>>` whereby an `Optional.absent()` is returned if there is no value associated with a key, or the value otherwise. That way we don't need to wrap every `get` with an `Optional.fromNullable`, and not create an `Optional` on every invocation.

Firstly, a new interface extending Map with a couple of extra methods which let you add instances of `V` directly, without having to first create the `Optional` wrapper:

implementation: call in the decorator

Guava's ForwardingMap comes in very handy if you want to add some functionality to an existing collection implementation, taking the legwork out of the decorator pattern. In our case we're changing the behaviour of an ImmutableMap. We firstly created a `ForwardingOptionalMap`, which modifies the underlying map's behaviour for `get` — which now returns `Optional.absent` as necessary — and the various `put`-type methods, to ensure people can't add `null` values:

You'll have spotted that this is an abstract class and be wondering where the concrete class is. We currently have an `ImmutableOptionalMap`, and we're all set for other implementations, such as a mutable variant:

in action

Do let us know if you've got any comments or suggestions for improvements!