tempuhs:

Status update, December 2014

Alexander Berntsen & Stian Ellingsen, plaimi, 2014

What's new

What's new

- tempuhs library

tempuhs:

Status update, December 2014

- Users
- Permissions
- Roles
- User attributes
- Complete restructuring of code

tempuhs:

Status update, December 2014

```
Role json
→ name              Text
→ namespace         UserId
→ UniqueRole        namespace name
→ rubbish           UTCTime                   Maybe
deriving Show

User json
→ name              Text
→ UniqueUser        name
→ rubbish           UTCTime                   Maybe
deriving Show
```

tempuhs:

Status update, December 2014

```
UserAttribute json
→ user                  UserId
→ name                  Text
→ value                 Text
→ UniqueUserAttribute   user name
deriving Show

UserRole json
→ user UserId
→ role RoleId
→ UniqueUserRole        user role
→ rubbish               UTCTime         Maybe
deriving Show
```

tempuhs:

Status update, December 2014

Permissionset json

→ timespan	TimespanId	
→ role	RoleId	
→ own	Bool	
→ read	Bool	
→ write	Bool	
→ rubbish	UTCTime	Maybe
→ UniquePermissionset	timespan role	

deriving Show

tempuhs:

Status update, December 2014

- We still think this was maybe a good idea
- The frontend engineer seems to think it's OK
- & even if the overall structure has some bad ideas, most of it is quite general and reusable anyway

tempuhs:

Status update, December 2014

What's new

- tempuhs server

tempuhs:

Status update, December 2014

October

- TimespanAttributes may be inserted together with Timespans
- Timespan modification is more flexible – TimespanAttributes may be modified as part of the same request & query as modifying the Timespan itself
- Rewrite the tests in the name of The Right Thing

tempuhs:

Status update, December 2014

November

- Even more flexible Timespan modification – omitted fields are not overwritten with the same value
- Attribute filtering for Timespans
- Rewrite everything a few times in the name of The Right Thing

November (cont)

- Polymorphic rubbishing with Lens
- Lenses for all database fields
- Oh yeah and we did some boring stuff too
- Like implementing that Users & Roles & Permissions stuff
- It works real well as far as we can tell

tempuhs:

Status update, December 2014

December

- More flexible rubbishing
- Hard deletes
- We do HTTP PATCH and HTTP PUT correctly now (The Right Thing etc.)
- UserAttributes (To users what TimespanAttributes are to Timespans)

December (Cont)

- Rather than inserting a bunch of copypasta crap for UserAttributes, we, once again, did The Right Thing
- And it's awesome
- But the types are pretty scary

tempuhs:

Status update, December 2014

-
  ```
  cmpMaybe :: forall
                 t (query :: * -> *) (expr :: * -> *)
                 backend (query1 :: * -> *) (expr1 :: * -> *)
                 backend1 typ.
                 (E.Esqueleto query1 expr1 backend1
                 ,E.Esqueleto query expr backend
                 ,E.PersistField typ
                 ,E.PersistField t)
              => (expr1 (E.Value (Maybe typ))
                  -> expr (E.Value (Maybe t))
                  -> expr1 (E.Value Bool))
                  -> expr1 (E.Value (Maybe typ))
              -> Maybe t
              -> expr1 (E.Value Bool)
  ```

- Is this even srs??

December (Cont)

- Flexible timespans that expand and contract based on children
- This is very non-trivial, and mutually recursive and ugh
- Optimally we would have liked a research team and five years
- Instead we had a few hacks and a very long Friday

Flexible timespans

- A timespan that is not rubbish may be a flexible timespan.
- A timespan with a clock with a name of "<~>" is a flexible timespan.
- A flexible timespan's beginMin is equal to the smallest beginMin of all of its immediate descendants' beginMin.
- A flexible timespan's endMax is equal to the biggest endMax of all of its immediate descendants' endMax.

Flexible timespans

- A timespan may not have a descendant as a parent.
- A timespan may not have itself as a parent.

tempuhs:

Status update, December 2014

-
  ```
  isFlexibleProp :: Timespan -> Bool
  isFlexibleProp Timespan{timespanRubbish = Nothing}= False
  isFlexibleProp _                                  = True
  ```

-
  ```
  isFlexProp :: Clock -> Bool
  isFlexProp c = (clockName c == "<~>")
  ```

-
  ```
  beginMinProp :: [Timespan] -> ProperTime
  beginMinProp = minimum . map timespanBeginMin
  ```

-
  ```
  endMaxProp :: [Timespan] -> ProperTime
  endMaxProp = maximum . map timespanEndMax
  ```

-
  ```
  parentCycleProp :: Eq a => a -> [a] -> Bool
  parentCycleProp = not .: elem
  ```

December (Cont)

- Authentication and authorisation
- tempuhs-server receives a signed request from a client
- It authenticates the client, which then becomes appropriately authorised (per its permissions)
- This might be secure, BRB gonna go test it and stuff

tempuhs:

Status update, December 2014

What's new

- オートちゃん

Authentication & authorisation

- Every free solution is at least slightly terrible
- Privacy & security is srs and should preferably not be terrible
- The solution??

tempuhs:

Status update, December 2014

Alexander Berntsen & Stian Ellingsen 2014

tempuhs:

Status update, December 2014

オートちゃん

- HMAC-based authentication (HMAC is really good)
- Standard HTTP headers for authentication
- かわいいです！

tempuhs:

Status update, December 2014

オートちゃん

- Users add trusted services as clients, and tells オートちゃん what permissions they should have (read/write)
- オートちゃん gives its clients an ID and secret key
- The clients use these to sign their requests
- オートちゃん then authorises the client to transform the user's data, respecting the permissions

tempuhs:

Status update, December 2014

オートちゃん

- OBTW permissions is a tempuhs construct
- オートちゃん doesn't really care about permissions per-se, it just supports flags of whatever kind
- It's completely generic and generally doesn't care too much about anything; except for verifying that requests are properly signed, and that the correct authorisation is performed

Future plans

tempuhs:

Status update, December 2014

- We should formalise how mytimelines.org uses tempuhs-server, rather than having a bunch of ad-hoc conventions that might make sense maybe sometimes
- Implement some documentation system for flexible API documentation generation
- We can do more interesting timespans, like timecycles
- Timespans may store spatial information & filter events per spatial data

tempuhs:

Status update, December 2014

- We need to be able to express more types of <u>relationships</u>
- And we need to be able to express timespan <u>weight</u> more sensibly
- This presents a twofold challenge
  - How do we <u>express</u> these things?
  - How do we <u>deal</u> with these things?
- The naïve relationships we have presently, are OK... presently
- The weighting simply isn't

tempuhs:

Status update, December 2014

- We can store semantic information about the timespans
- Users may tag the timespans with tags
- This can be used to filter the timespans
- Semantic data may also help us weight the visualisation, and lets us recommend users to add timespans (history is a set of agreed upon lies)
- Semantic data may also be inferred from timespans & their relationships

tempuhs:

Status update, December 2014

- Time spans need a time specification and a DSL to implement it
- Clocks need a conversion DSL
- Magical common indexing unicorn
- Split オートちゃん into several small modules
- Formally verify オートちゃん to be correct

tempuhs:

Status update, December 2014

- The scary types need to go
- The functions aren't even typesafe!
- We're going to look at chucking out Esqueleto and use Opaleye instead
- This will be quite a bit of effort, but very worth it
- Compiletime detection of ill-formed SQL queries? Amazing!

tempuhs:

Status update, December 2014

- https://secure.plaimi.net/works/tempuhs.html
- https://github.com/plaimi/tempuhs
- https://github.com/plaimi/tempuhs-server
- https://github.com/plaimi/authochan