Voyage

The adventure of persisting object-models

A glimpse of Seaside

- Stateful
- Component model
- Maybe aged, but still the best :)

Continuations

I still prefer:

```
n1 := self request: 'Number 1'.
n2 := self request: 'Number 3'.
self inform: 'Value: ', (n1 + n2)
```

Continuations

Over:

```
someBehaviorOn: html
        ...
        html inputText
                onEnter: (html jQuery ajax
                        serializeThis
                        onSuccess: ((self canvas jQuery: #id) load
                                html: [ :canvas |
                                        self someOtherBehaviorOn: canvas ])))

someOtherBehaviorOn: html
        ...
```

A glimpse of Pier

- A CMS
- Component based
- You can embed almost anything
- You can decorate almost anything

A perfect world

A perfect world

- Objects collaborate inside your image, creating a perfect choreography of interacting elements...

A perfect world

- Objects collaborate inside your image, creating a perfect choreography of interacting elements...
- INSIDE your image

Why to persist?

- Space
- Reliability
- Scalability

Persisting Pier

- A complex* object model
- Looking for
  - Scalability
  - Update capabilities
  - Backup

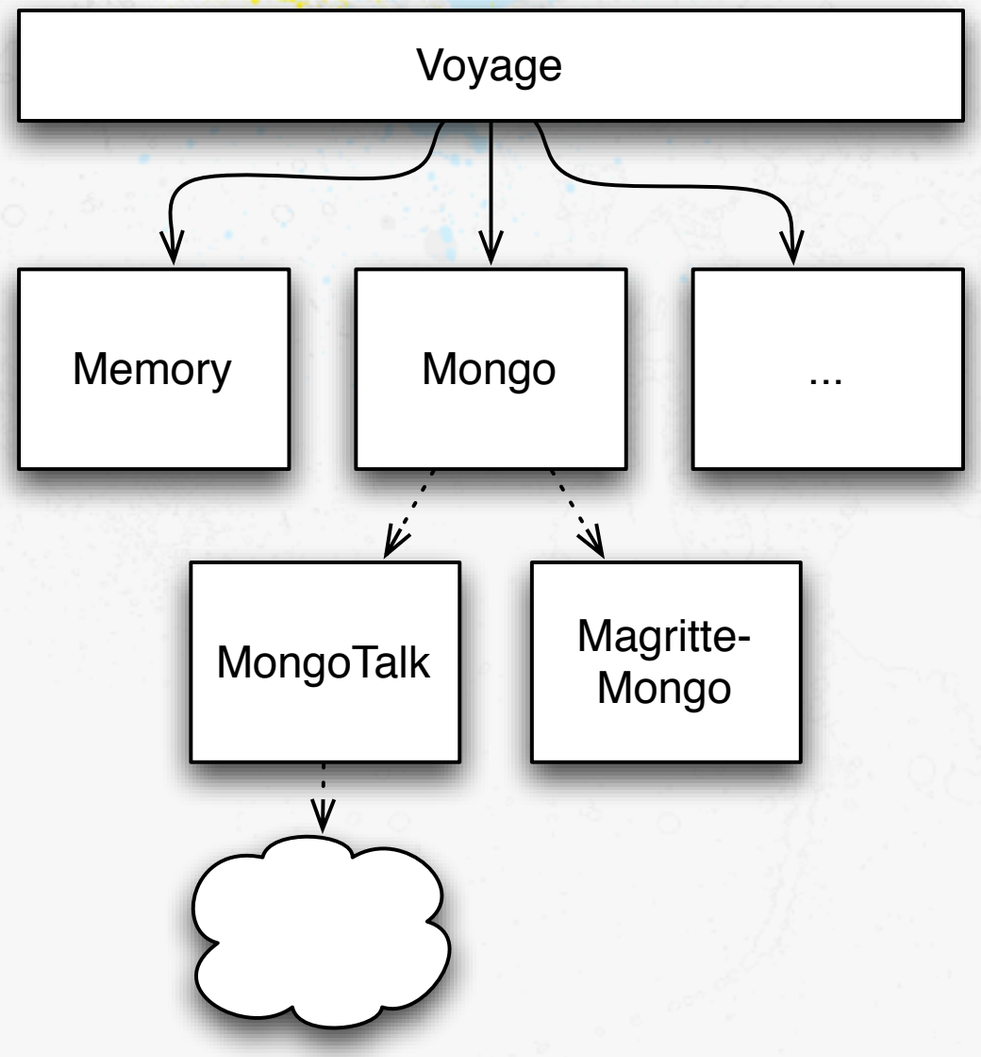
*And I mean **really** complex: take a look to all those commands, decorations and dictionaries :)

Show time!

Voyage

- A “non-abstract” persistence layer
- An “implementation pattern” which provides some services as well

A simple layered approach



Use cases

Singleton mode:

```
SomeClass selectAll.
anObject save.
```

Regular mode:

```
repository selectAll: SomeClass.
repository save: anObject.
```

A simple API

- #save:
- #remove:
- #removeAll:
- #selectAll:
- #selectMany:where:
- #selectOne:where:

A simple API

- #save:
- #remove:
- #removeAll:
- #selectAll:
- #selectMany:where:
- #selectOne:where:

Where clauses are what the back-end needs, no something intermediate

A simple API

- #save:
- #remove:
- #removeAll:
- #selectAll:
- #selectMany:where:
- #selectOne:where:

Where clauses are what the back-end needs, no something intermediate

Memory:
[:each | each key = 42]

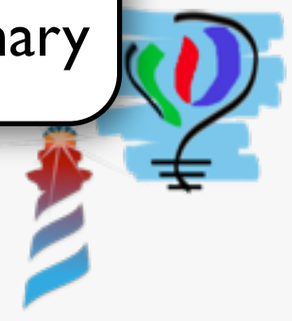
A simple API

- #save:
- #remove:
- #removeAll:
- #selectAll:
- #selectMany:where:
- #selectOne:where:

Where clauses are what the back-end needs, no something intermediate

Memory:
[:each | each key = 42]

Mongo:
{ #key->42 } asDictionary



What Voyage provides

- Common API
- Centralized management
  - Preserve identity (caching live objects)
  - Error handling
  - Reconnection
  - Pluggable connection pool

Voyage-Memory

- Is just a centralized dictionary of collections.
- Used for prototyping and early stages of development (no need to choose a persistent repository at first instance)
- Works for “stateless” applications which need readonly data (I made one once, yes)

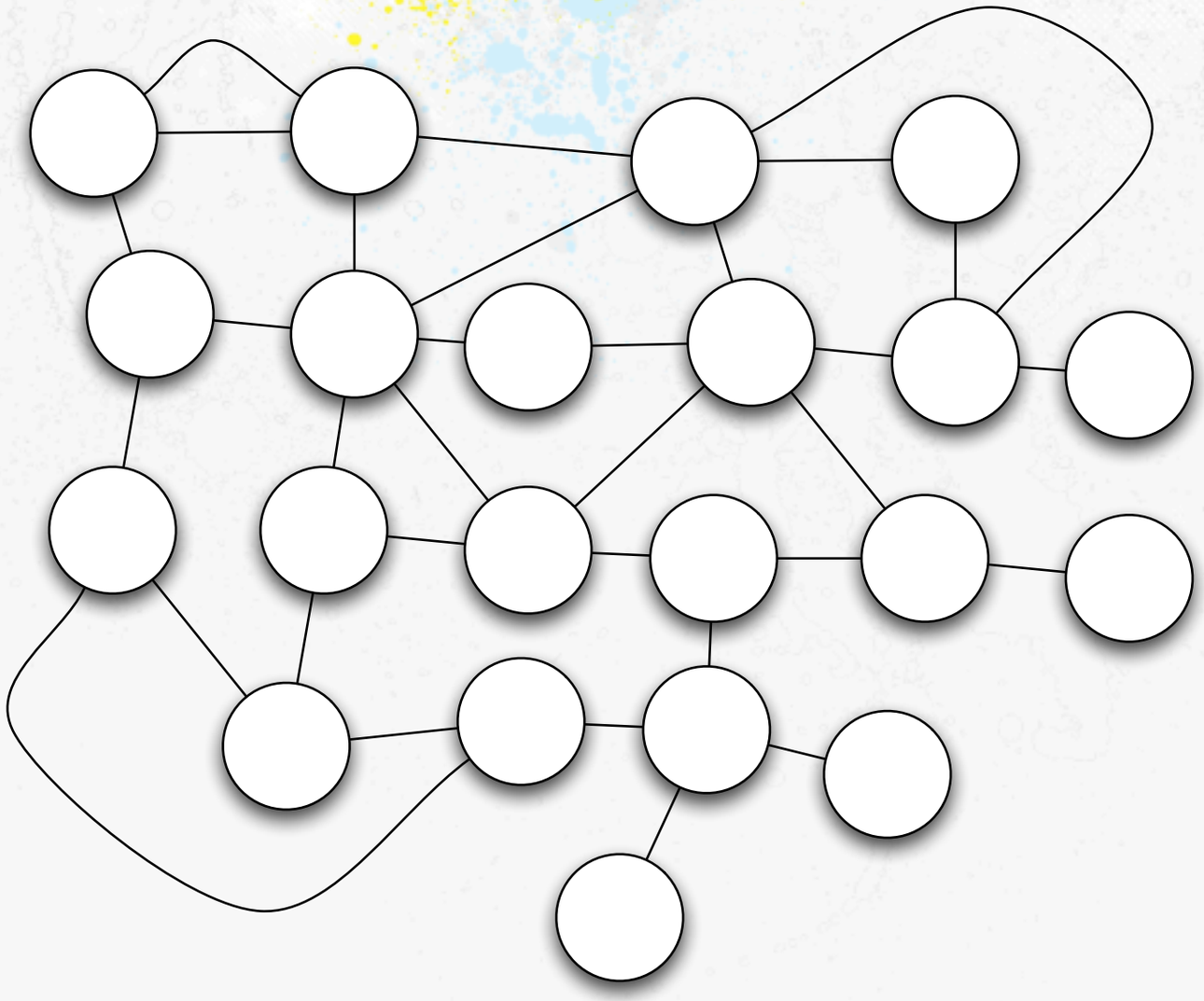
Voyage-Mongo

- Uses MongoTalk
- Adds
  - Object cache
  - Error handling/reconnection
  - Transparent mapping (magritte guided)

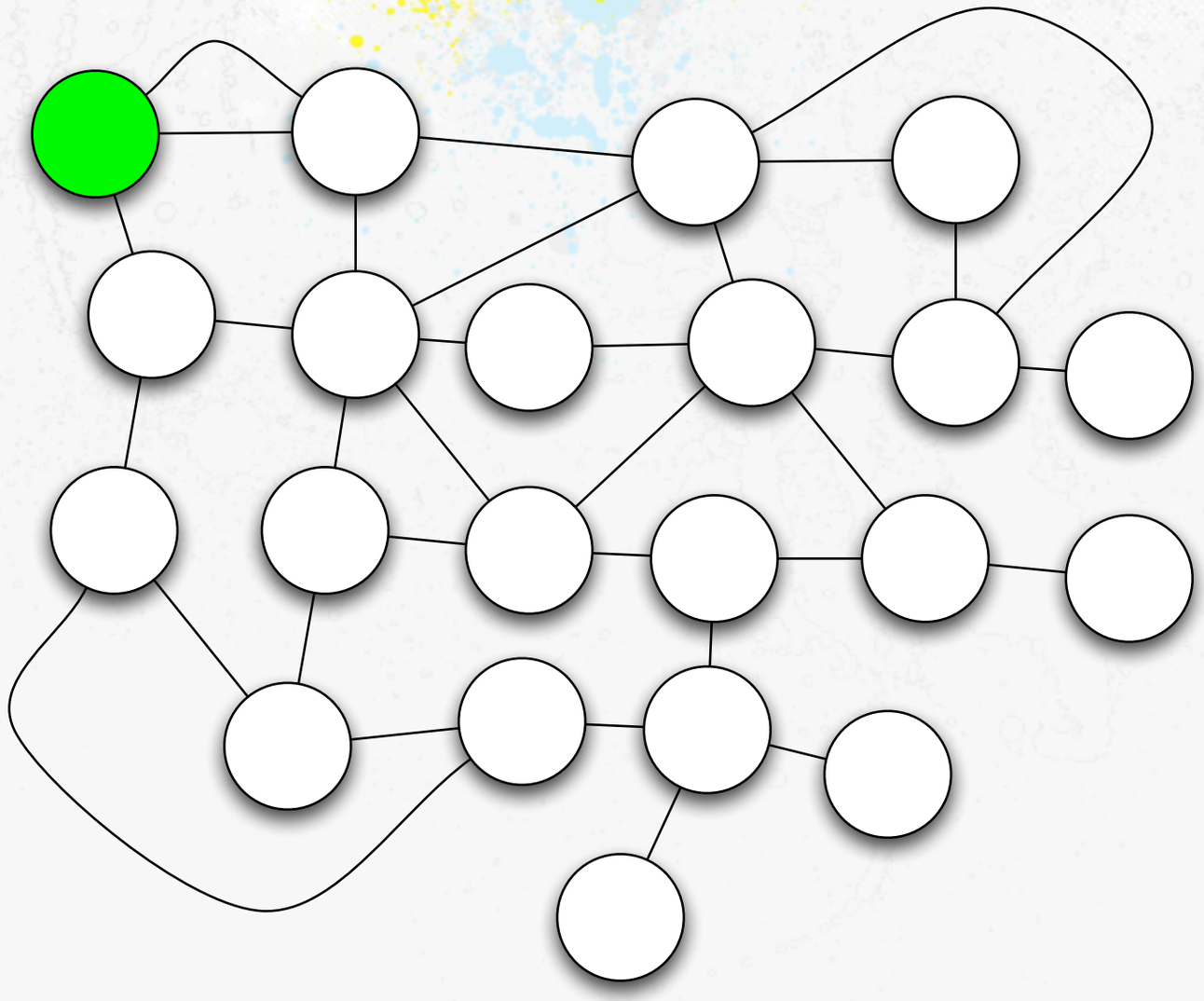
Magritte-Mongo

- "Object-Document mapper"
- Magritte base implementation is intended to UI and there is some difference with mongo needs.
  - Explicit (static) declaration vs. dynamic type inference. (i.e. Pier persistence is nearly impossible with an explicit declaration approach)
- I changed it a lot since Norbert saw him (more and more inference... and still needs more)

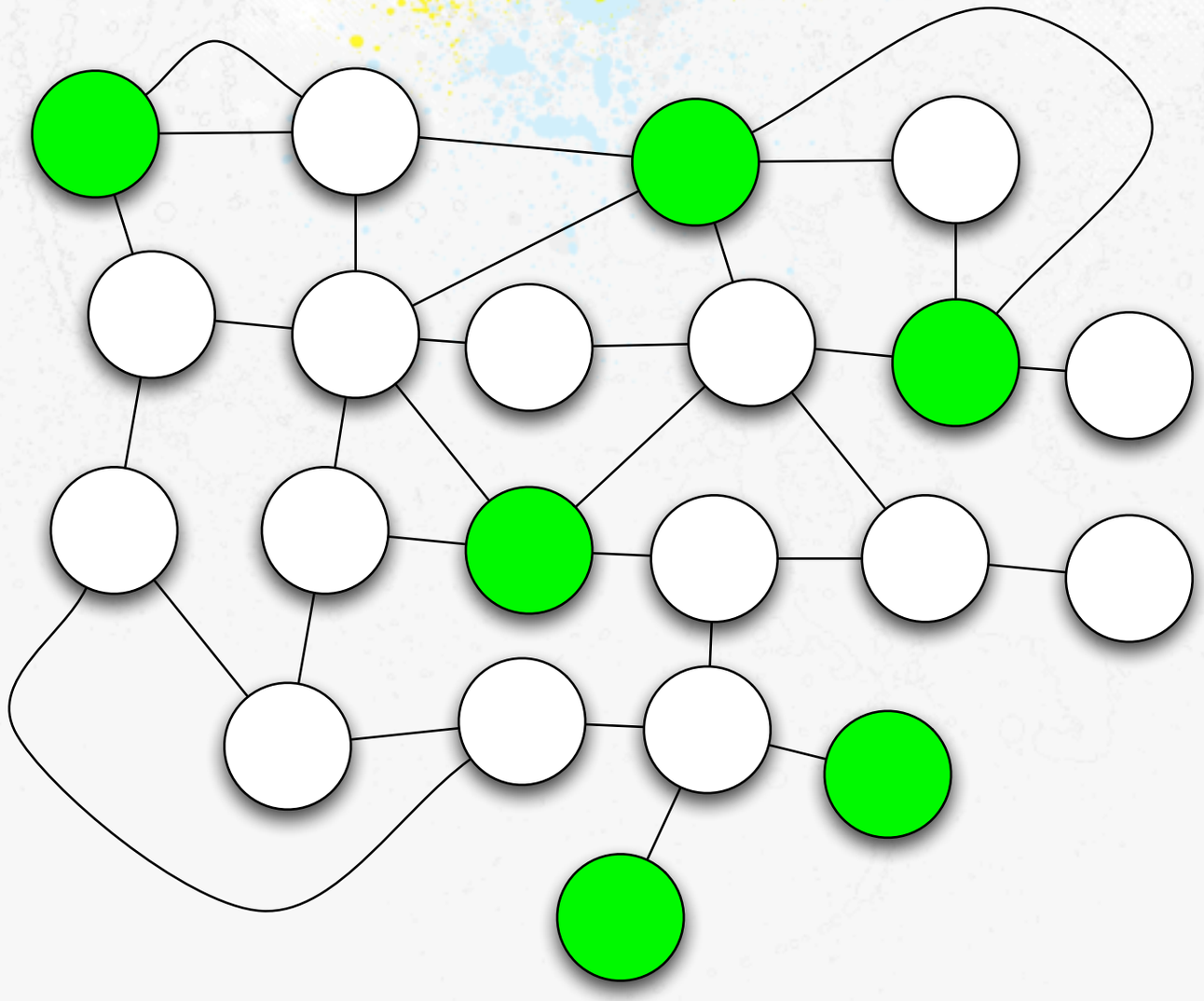
Serializer



Serializer



Serializer



Serializer

- Uses become, so yes, it is slow
  - So far, no need more speed.
- Other possible approaches: Two phases (mark and persist)

Status

✓ First version working (make it work)

- Need cleanup (make it right)
- Need some optimization (make it fast)

Future

- Finish it (collaborators would be fine :)
- Add/Update some backends (Just if/when needed)
  - Fuel
  - Phriak
  - Glorp?

Thanks!

Voyage and PierVoyage are available at:
http://smalltalkhub.com/#!/~estebanlm/Voyage

Already a tester who is blogging about!
http://articles.tulipemoutarde.be/