ArangoDB Foxx: Building APIs on top of ArangoDB
ArangoDB Foxx: Building APIs on top of ArangoDB
NoSQL is not one thing
NoSQL does not mean you can't do joins.
NoSQL does not mean you can’t do transactions.
Aggregate Oriented Databases
Graph Databases

Data

Servers

Structure

Graph

Databases

Aggregate

Oriented

Databases

Oriented

Time

Series

Databases
Why combine aggregate and graph?

Identified by...

- Entity
  - Add an edge to me!

- Value Object
  - Embed me!

State is...

- Mutable
- Immutable
Hi.

I’m Lucas.
I work for ArangoDB GmbH
We build ArangoDB
ArangoDB is an Open Source NoSQL Database
ArangoDB

- multimodel
- open source
- free
- C++
- JavaScript
- space efficient
- sharding
- transactions
- joins
- master-slave replication
AQL
FOR user IN users
    FILTER user.active == true
RETURN {
    name: user.name
}

FOR u IN users
  FILTER u.active == true
  LIMIT 0, 4
FOR f IN relations
  FILTER f.type == "friend" && f.from == u.id
RETURN {
  "user" : u.name,
  "friendId" : f.to
}
FOR u IN users
    LET userRelations = (FOR p IN PATHS(
        users,
        relations,
        "OUTBOUND"
    )
        FILTER p._from == u._id
        RETURN p
    )
    RETURN {
        "user" : u,
        "relations" : userRelations
    }
JS extends ArangoDB

Multi Collection Transactions
Individual Graph Traversals
Extending the Web API
App 1
Structured
Controller
Foxx = require("org/arangodb/foxx");
Foxx = require("org/arangodb/foxx");

controller = new Foxx.Controller(appContext);
Foxx = require("org/arangodb/foxx");
controller = new Foxx.Controller(appContext);
controller.get("/users", function(req, res) { });
Foxx = require("org/arangodb/foxx");

controller = new Foxx.Controller(appContext);

controller.get("/users", function(req, res) {
    res.json({
        hello: "world"
    });
});
Parameterize your routes
Foxx = require("org/arangodb/foxx");

controller = new Foxx.Controller(appContext);

controller.get("/users", function(req, res) {
    res.json({
        hello: "world"
    });
});
Foxx = require("org/arangodb/foxx");

controller = new Foxx.Controller(appContext);

controller.get("/users/:name", function(req, res) {
  res.json({
    hello: "world"
  });
});
Foxx = require("org/arangodb/foxx");

controller = new Foxx.Controller(appContext);

controller.get("/users/:name", function (req, res) {
  res.json({
    hello: "world"
  });
});

req.params("name");
Annotate your routes
controller.get("/users/:name", function(req, res) {
    res.json({
        hello: req.params("name");
    });
});
controller.get("/users/:name", function(req, res) {
    res.json(
        {
            hello: req.params("name");
        }
    );
patternParam("name", {
    type: joi.string().description("Name of the User")
});
/** What's my name? */

* This route knows it. */

controller.get("/users/:name", function(req, res) {
    res.json({
        hello: req.params("name");
    });
}).pathParam("name", {
    type: joi.string().description("Name of the User")
});
Model Layer
Domain Models

Foxx.Model

Persistence

Foxx.Repository
Foxx.Model
• Representation of the data
• Convenience Methods
• Validation

Foxx.Repository
• Save and Retrieve Data
• Simple Queries
• Define your own queries
Full Access To Simple Queries
Full Access To AQL
Full Access To Traversals
Background Worker
Authentication Module
### Active Applications

<table>
<thead>
<tr>
<th>Application</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>aardvark (system)</td>
<td>active</td>
</tr>
<tr>
<td>aye-aye (dev)</td>
<td></td>
</tr>
<tr>
<td>cerberus (system)</td>
<td>active</td>
</tr>
<tr>
<td>gharial (system)</td>
<td>active</td>
</tr>
<tr>
<td>sessions (system)</td>
<td>active</td>
</tr>
<tr>
<td>simple-auth (system)</td>
<td>active</td>
</tr>
<tr>
<td>users (system)</td>
<td>active</td>
</tr>
</tbody>
</table>

### Available Applications

<table>
<thead>
<tr>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>aardvark (system)</td>
</tr>
<tr>
<td>aye_aye</td>
</tr>
<tr>
<td>cerberus (system)</td>
</tr>
<tr>
<td>gharial (system)</td>
</tr>
<tr>
<td>oauth2</td>
</tr>
<tr>
<td>sessions (system)</td>
</tr>
<tr>
<td>sessions-example-app</td>
</tr>
<tr>
<td>simple-auth (system)</td>
</tr>
<tr>
<td>users (system)</td>
</tr>
</tbody>
</table>
FoxxGenerator uses statemachines to create REST APIs
REST
by Fielding (not DHH)
Representational state transfer
Let’s talk about state
How do we get from state to state?
JSON doesn’t have links.

Siren & HAL have.
Why choose one of those standards?

To use existing libraries
How do we get from state to state?
showIdeas

ideas

showIdea

idea

relatedIdea
A single idea entity parameterized

showIdeas -> ideas -> showIdea

relatedIdea

A single idea entity parameterized
A repository of entities
generator.addStartState({
    transitions: [
        { to: 'ideas', via: 'showIdeas' },
    ]
});

generator.addState('ideas', { 
    type: 'repository',
    contains: 'idea',

    transitions: [
        { to: 'idea', via: 'showIdea' } 
    ]
});

generator.addState('idea', { 
    type: 'entity',
    parameterized: true,
    containedIn: 'ideas',

    attributes: {
        title: Joi.string().required(),
        description: Joi.string().required()
    },

    transitions: [
        { to: 'idea', via: 'relatedIdea' } 
    ]
});
Every state is a... repository, entity or service
An entity has attributes

A service has an action
/** Get a list of all ideas

* This will show you a list of all ideas that are currently available.
*/
generator.defineTransition('showIdeas');
A transition can be a follow transition
A transition can be a
connect
transition
A transition can be a disconnect transition
A transition can be a modify transition.
A transition can have a precondition
A transition can have parameters.
You:
1. Choose Media Type
2. Define Transitions
3. Document Transitions
4. Describe Statemachine
5. Generate
FoxxGenerator:

- Generates collections
- Generates underlying graph
- Generates models and repositories
- Generates all routes
- Uses correct status codes
- Gives error responses
- Validates input parameters
- Follows selected standard
- Generates interactive documentation
How?

- The description is in the file system
- Use Git!
- Either only generate the initial version and then use Foxx for Rapid Prototyping
- or evolve it
- Can be combined with a Foxx app
When?

Christmas Release
Thanks for listening

https://www.arangodb.com/download

I have stickers!

lucas@arangodb.com

@moonbeamlabs on Twitter

moonglum on Github