StrongLoop™

Working with LoopBack Models

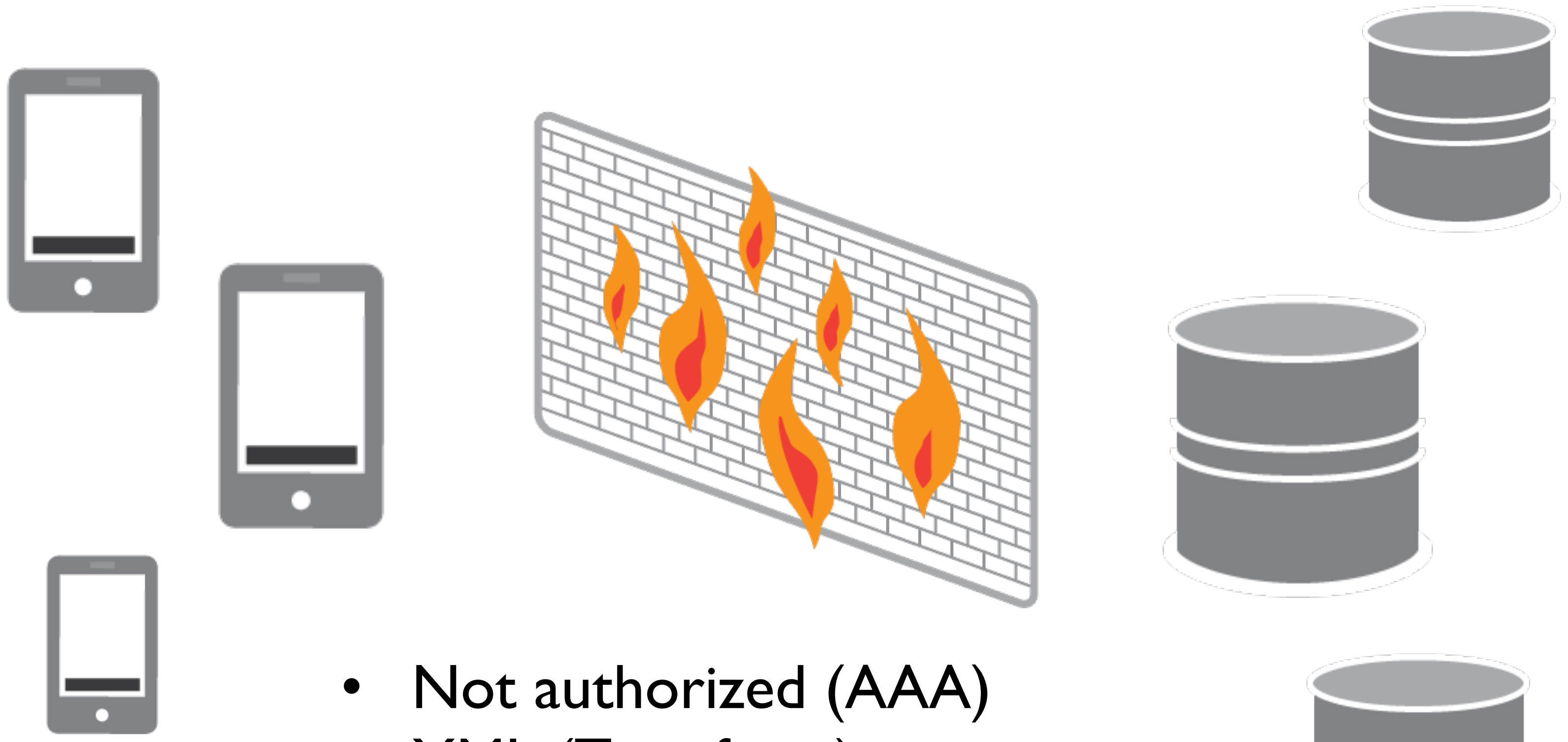
Raymond Feng
Co-Founder and Architect

About StrongLoop

- Founded 2012
- Develop and support…
  - LoopBack: Open Source Mobile Backend-as-a-Service
  - StrongOps (formally NodeFly): Real-time performance monitoring
  - StrongNode: Support for StrongLoop and public Node.js modules
- Also maintains and/or contributes to the npm ecosystem:
  - node-inspector, node-reggie plus over 30 more modules

The Problem: Apps Need Data

StrongLoop

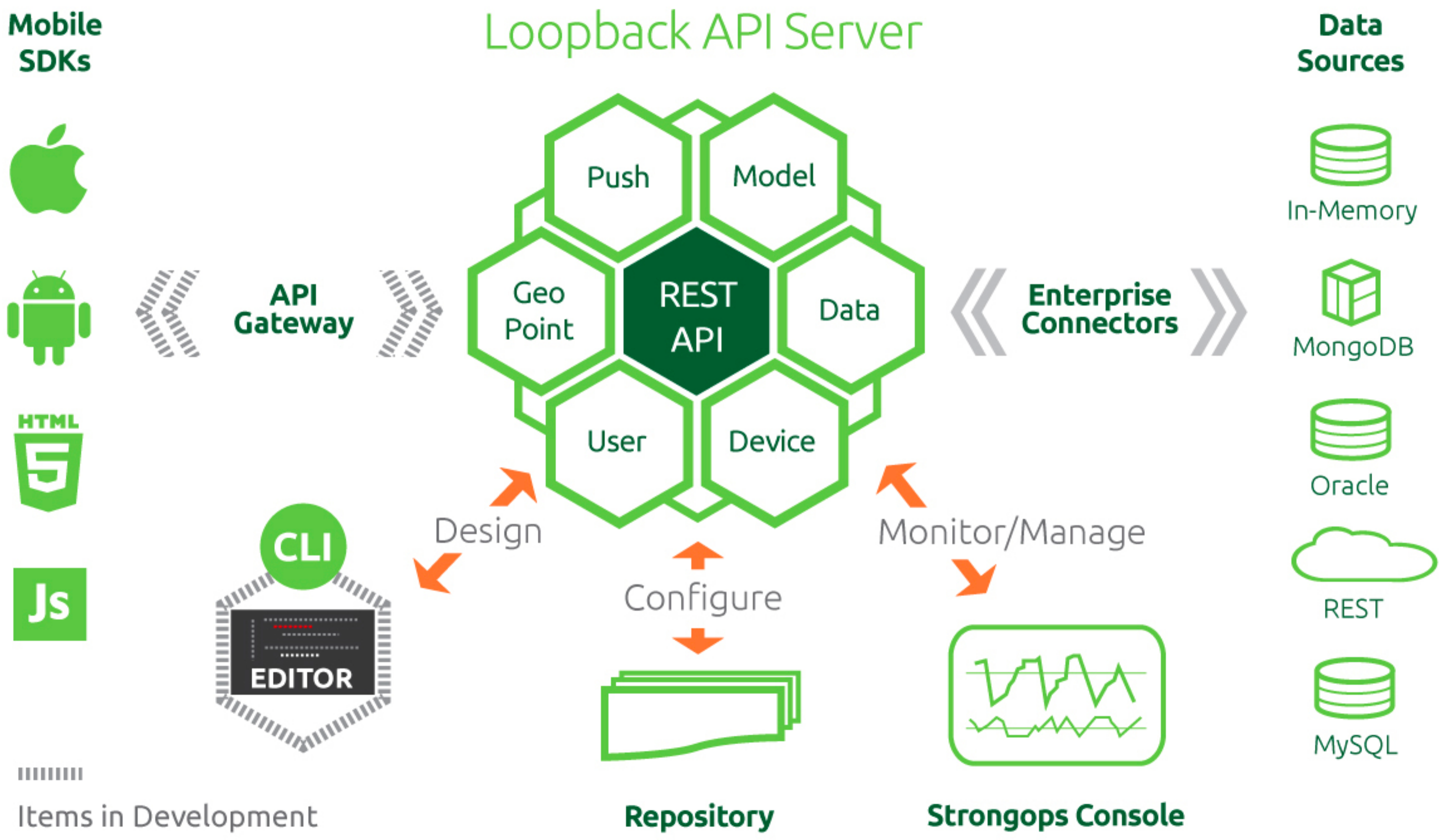


- Not authorized (AAA)
- XML (Transform)
- Too much data (Filter)
- Combine multiple DBs (Join)
- 50k phones kill DB (Cache)

Introducing LoopBack

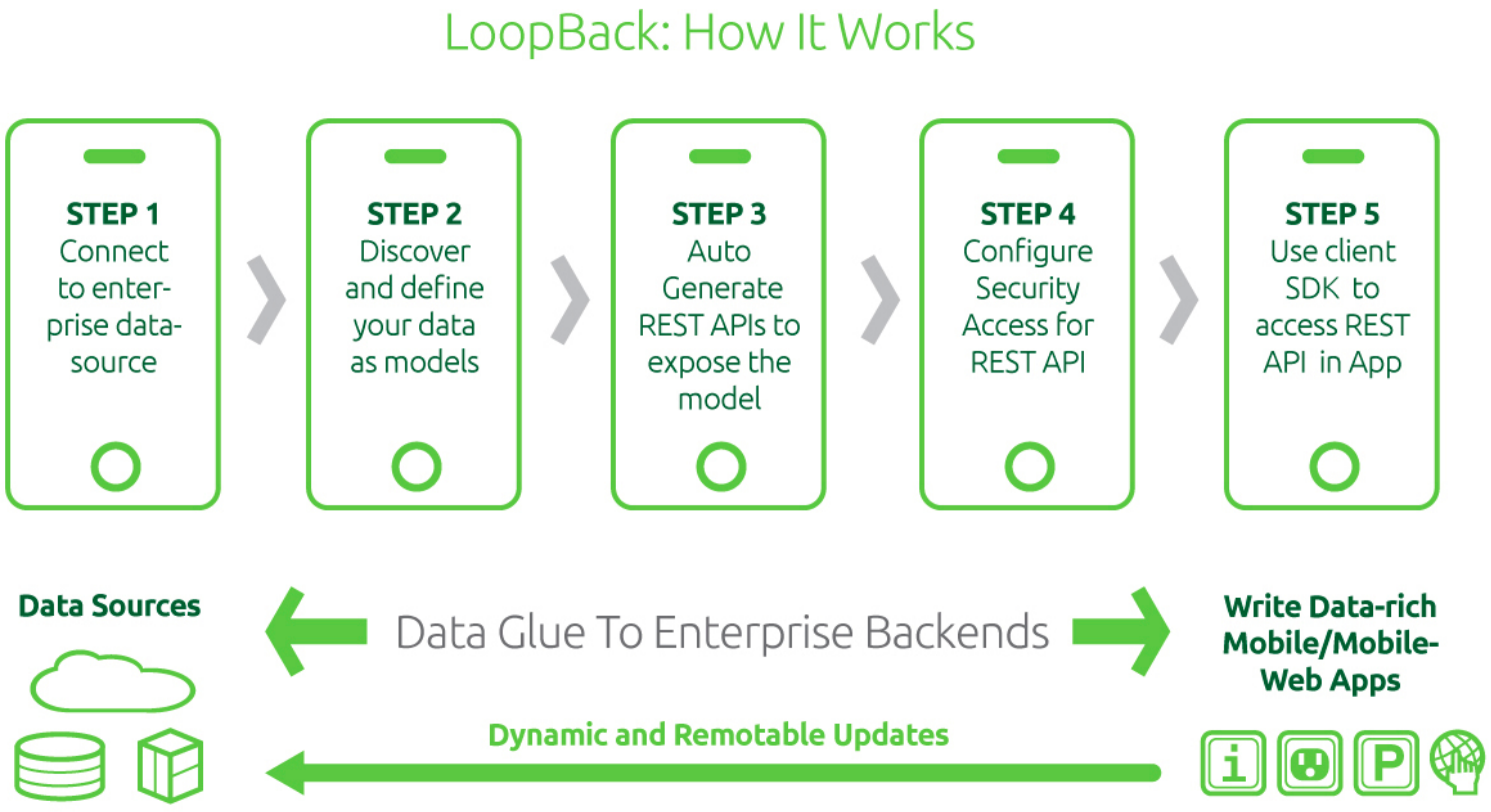
- How can we build scalable Enterprise mobile apps?
- Mobile Backend-as-a-Service (e.g. a private Parse you control)
- Connects devices and browsers to Enterprise data
- Written in Node.js – proven language for mobile backends
- Open source – extensible by design
- On-premise or on your favorite cloud
- Android and iOS SDKs

LoopBack Architecture



How it Works

StrongLoop


LoopBack: How It Works
STEP 1 Connect to enter-prise data-source
STEP 2 Discover and define your data as models
STEP 3 Auto Generate REST APIs to expose the model
STEP 4 Configure Security Access for REST API
STEP 5 Use client SDK to access REST API in App
Data Sources
Data Glue To Enterprise Backends
Write Data-rich Mobile/Mobile-Web Apps
Dynamic and Remotable Updates

LoopBack

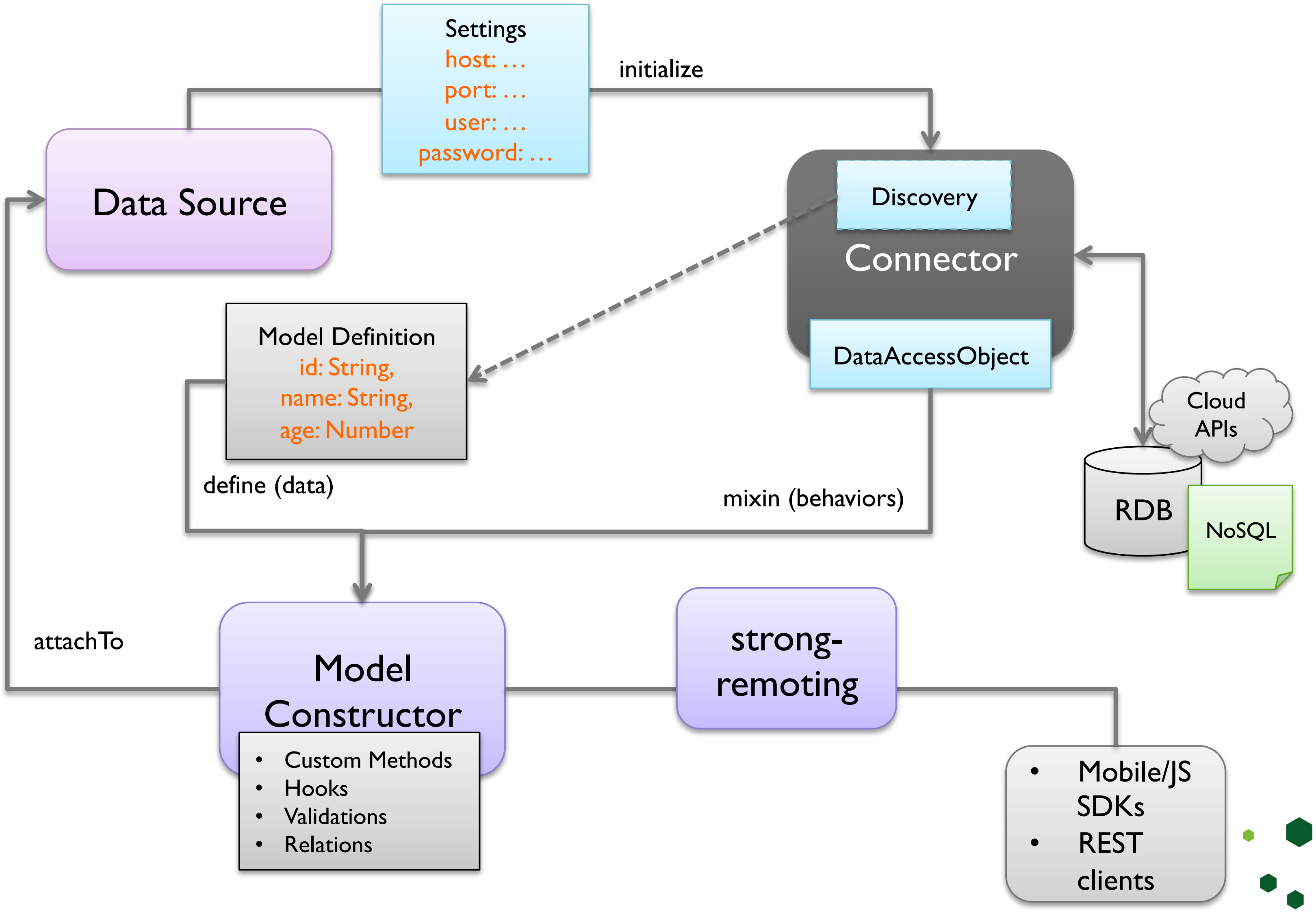
StrongLoop

- Backend for mobile applications (native, web, and hybrid)
- Frontend for traditional enterprise systems
- Model = data + behavior.
- Isomorphic models: LoopBack, backend DBs, frontend

Model = Data + Behavior

- Rich mobile applications are driven by data.
- Data is created and consumed by mobile devices, browsers, cloud services, legacy apps, databases, and other backend systems.
- Mobilizes data through *models* that represent business data and behavior.
- Exposes models to mobile apps through REST APIs and client SDKs.
- You need to interact with the model differently, depending on the location and type of data.

The big picture



Choose Your Camp and Recipes

1. Open Models
   - "I don't know my data model yet. Let's start free form and show me the CRUD APIs!"
2. Models with schema
   - "Now I can tell you more information about my data model. Let's add properties!"
3. Discover models
   - "Hey, I already have data in relational databases such as Oracle or MySQL. Can the table schema be my data model?"
4. Models by instance introspection
   - "Sorry, I'm a NoSQL guy and I have JSON documents for my data. Reverse engineering?"
5. Model synchronization with relational databases
   - "Now I have the data model, should I beg the DBA to create/update the tables/columns for me?"

Recipe 1

I'm mobile developer. Can LoopBack help me store and load data transparently? I don't need to worry about the backend or define the model up front, because my data are free-form.

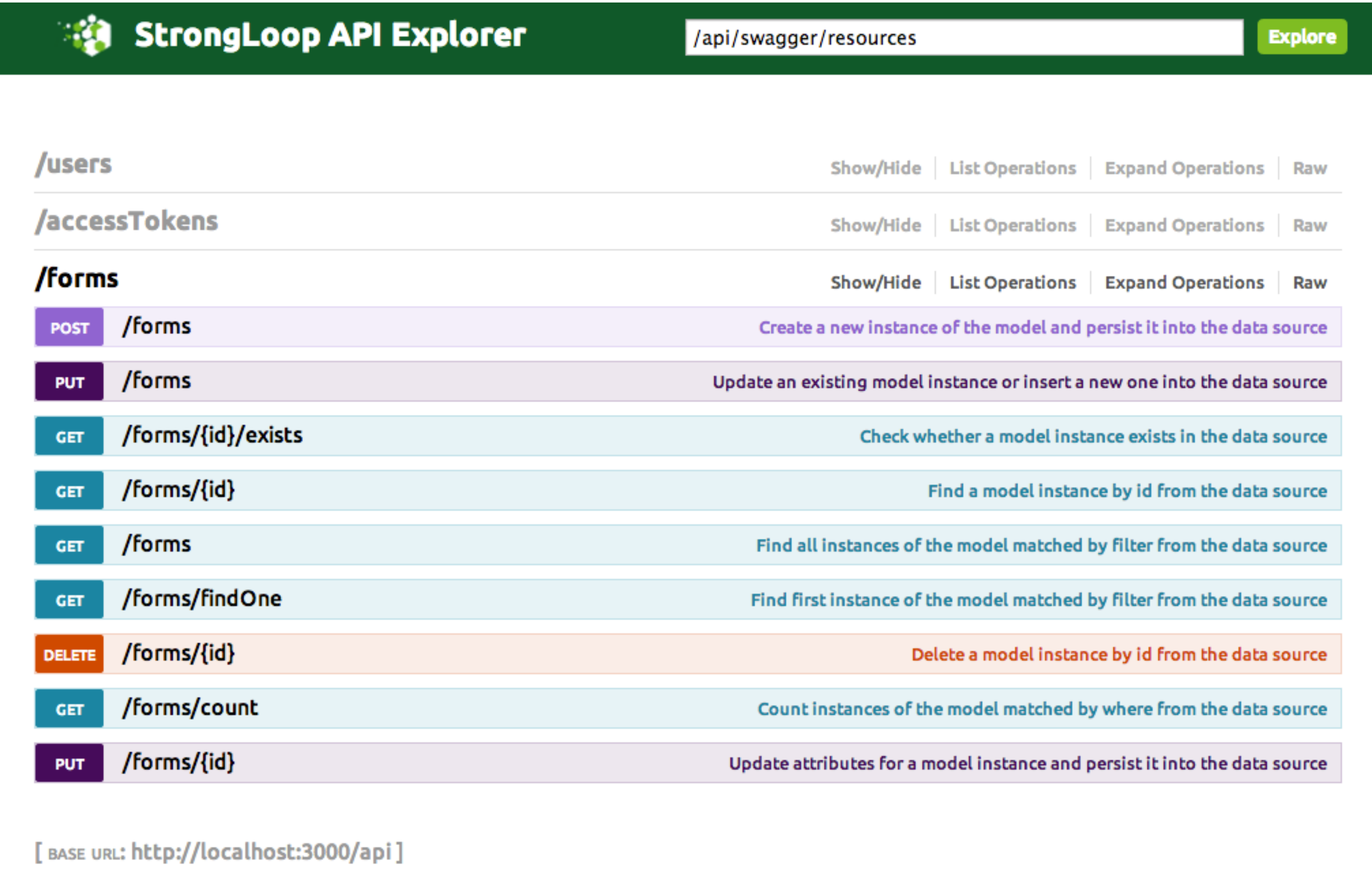
Open Models

- Open models are perfect for free-form data or API mockup

```
npm install -g strong-cli

slc lb project loopback-models
cd loopback-models
slc lb model form
slc run app
http://localhost:3000/explorer
```

Explore the APIs

StrongLoop



StrongLoop API Explorer

/api/swagger/resources

Explore

/users — Show/Hide | List Operations | Expand Operations | Raw

/accessTokens — Show/Hide | List Operations | Expand Operations | Raw

/forms — Show/Hide | List Operations | Expand Operations | Raw

POST	/forms	Create a new instance of the model and persist it into the data source
PUT	/forms	Update an existing model instance or insert a new one into the data source
GET	/forms/{id}/exists	Check whether a model instance exists in the data source
GET	/forms/{id}	Find a model instance by id from the data source
GET	/forms	Find all instances of the model matched by filter from the data source
GET	/forms/findOne	Find first instance of the model matched by filter from the data source
DELETE	/forms/{id}	Delete a model instance by id from the data source
GET	/forms/count	Count instances of the model matched by where from the data source
PUT	/forms/{id}	Update attributes for a model instance and persist it into the data source

[BASE URL: http://localhost:3000/api]

Recipe 2

StrongLoop

I want to build a mobile application that will interact with some backend data. I would love to see a working REST API and mobile SDK before I implement the server side logic.

Define the model

```
// Load the MongoDB data source
var ds = require('../data-sources/db.js')('mongodb');

// Define a customer model
var Customer = ds.createModel('customer', {
   id: {type: Number, id: true},
   name: String,
   emails: [String],
   age: Number},
   {strcit: true});
```

CRUD

```
Customer.create({
  name: 'John1',
  emails: ['john@x.com', 'jhon@y.com'],
  age: 30
}, function (err, customer1) {
  console.log('Customer 1: ', customer1.toObject());
  Customer.create({
    name: 'John2',
    emails: ['john@x.com', 'jhon@y.com'],
    age: 30
  }, function (err, customer2) {
    console.log('Customer 2: ', customer2.toObject());
    Customer.findById(customer2.id, function(err, customer3) {
      console.log(customer3.toObject());
    });
    Customer.find({where: {name: 'John1'}, limit: 3}, function(err, customers) {
      customers.forEach(function(c) {
        console.log(c.toObject());
      });
    });
  });
```

Recipe 3

StrongLoop™

I have data in an Oracle or MySQL database. Can LoopBack figure out the models and expose them as APIs to my mobile applications?

Connect to Oracle

```
var loopback = require('loopback');

  var ds = loopback.createDataSource('oracle', {
           "host": "demo.strongloop.com",
           "port": 1521,
           "database": "XE",
           "username": "demo",
           "password": "L00pBack"
      });
```

Discover and run

```
var ds = require('../data-sources/db.js')('oracle');

/**
 * Discover and build models from INVENTORY table
 */
ds.discoverAndBuildModels('INVENTORY', {visited: {}, owner: 'LOOPBACK',
associations: true}, function (err, models) {

    models.Inventory.findOne({}, function (err, inv) {
        if (err) {
            console.error(err);
            return;
        }
        console.log("\nInventory: ", inv);
        inv.product(function (err, prod) {
            console.log(err);
            console.log("\nProduct: ", prod);
            console.log("\n ------------- ");
        });
    });
```

Recipe 4

I have JSON documents from REST services and NoSQL databases. Can LoopBack introspect my models from them?

Sample JSON document

```
// Instance JSON document
var user = {
   name: 'Joe',
   age: 30,
   birthday: new Date(),
   vip: true,
   address: {
      street: '1 Main St',
      city: 'San Jose',
      state: 'CA',
      zipcode: '95131',
      country: 'US'
   },
   friends: ['John', 'Mary'],
   emails: [
      {label: 'work', eid: 'x@sample.com'},
      {label: 'home', eid: 'x@home.com'}
   ],
   tags: []
};
```

Build a model from JSON

```
var ds = require('../data-sources/db.js')('memory');

// Create a model from the user instance
var User = ds.modelBuilder.buildModelFromInstance('MyUser',
user, {idInjection: true});
User.attachTo(ds);

// Use the model for CRUD

User.create(user, function (err, u1) {
  console.log('Created: ', u1.toObject());
  User.findById(u1.id, function (err, u2) {
    console.log('Found: ', u2.toObject());
  });
});
```

Recipe 5

Now I have defined a LoopBack model, can LoopBack create or update the relational database schemas for me?

Model synchronization

- LoopBack provides two ways to synchronize model definitions with table schemas:
- **Auto-migrate:** Automatically create or re-create the table schemas based on the model definitions. *WARNING: An existing table will be dropped if its name matches the model name.*
- **Auto-update:** Automatically alter the table schemas based on the model definitions.

Summary

Recipe	Use Case	Model Strict Mode	Database
Open Model	Taking care of free-form data	false	NoSQL
Plain Model	Defining a model to represent data	true or false	NoSQL or RDB
Model from discovery	Consuming existing data from RDB	true	RDB
Model from introspection	Consuming JSON data from NoSQL/REST	false	NoSQL
Model synchronization	Making sure models are in sync	true	RDB

What’s Next?

- Try LoopBack

  strongloop.com/get-started/

- RTFM

  docs.strongloop.com

- Questions?

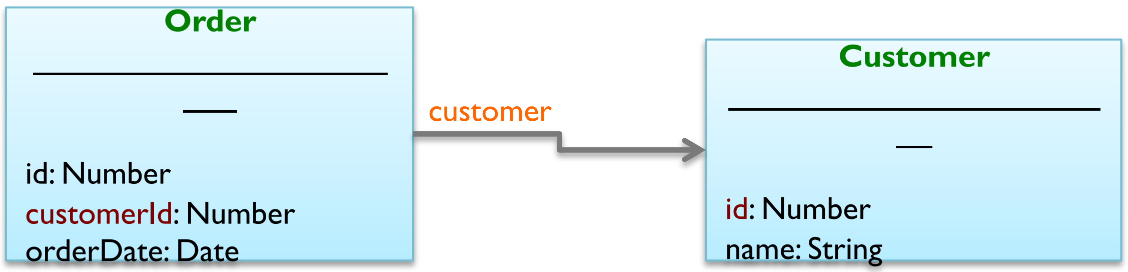
  groups.google.com/forum/#!forum/strongloop

  or callback@strongloop.com

Recipe 6: Relations

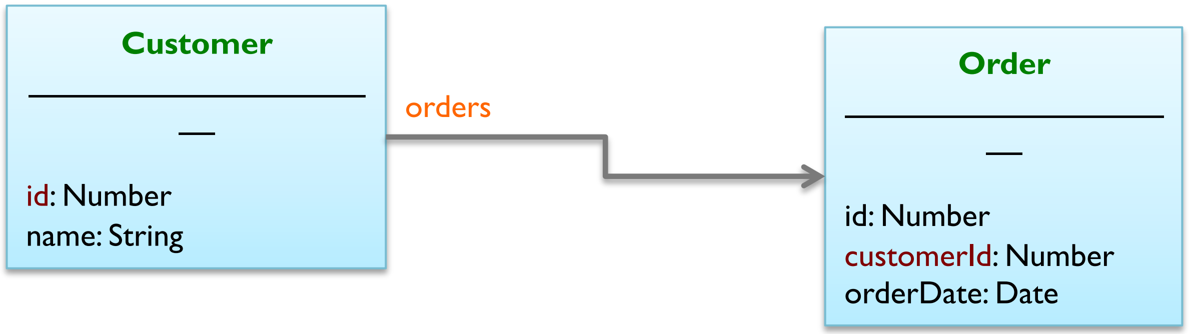
- Models are often connected/related. For example,
  - A customer has many orders and each order is owned by a customer.
  - A user can be assigned to one or more roles and a role can have zero or more users.
  - A physician takes care of many patients through appointments. A patient can see many physicians too.

belongsTo



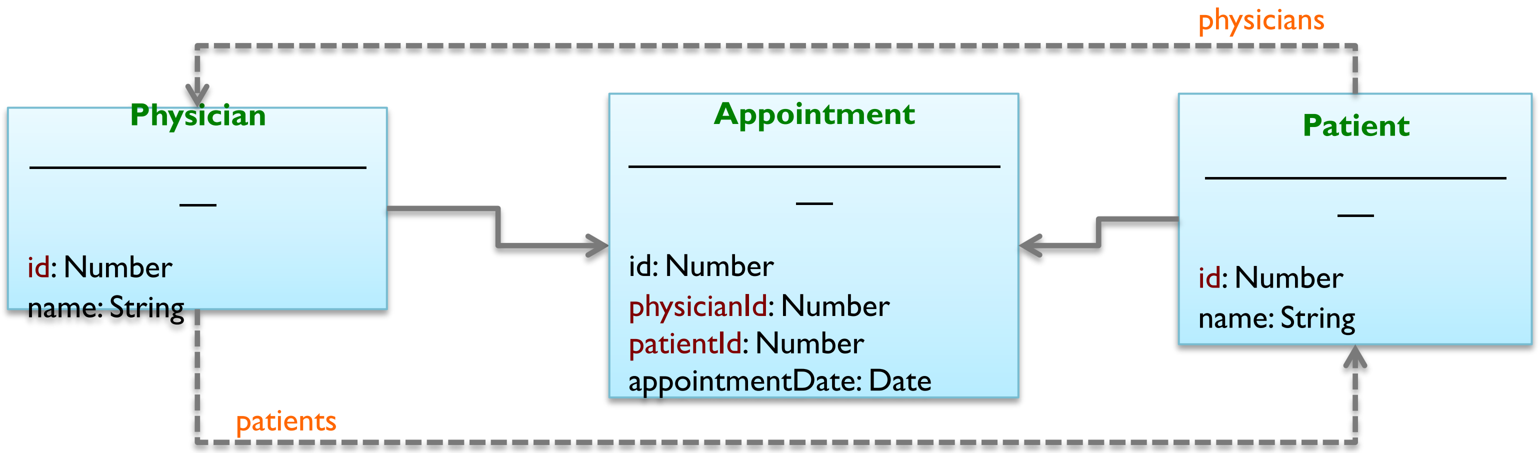
```
var Order = ds.createModel('Order', {
    customerId: Number,
    orderDate: Date
});

var Customer = ds.createModel('Customer', {
    name: String
});

Order.belongsTo(Customer);

...

order.customer(callback); // Get the customer for the order
order.customer(); // Get the customer for the order synchronously
order.customer(customer); // Set the customer for the order
```

hasMany



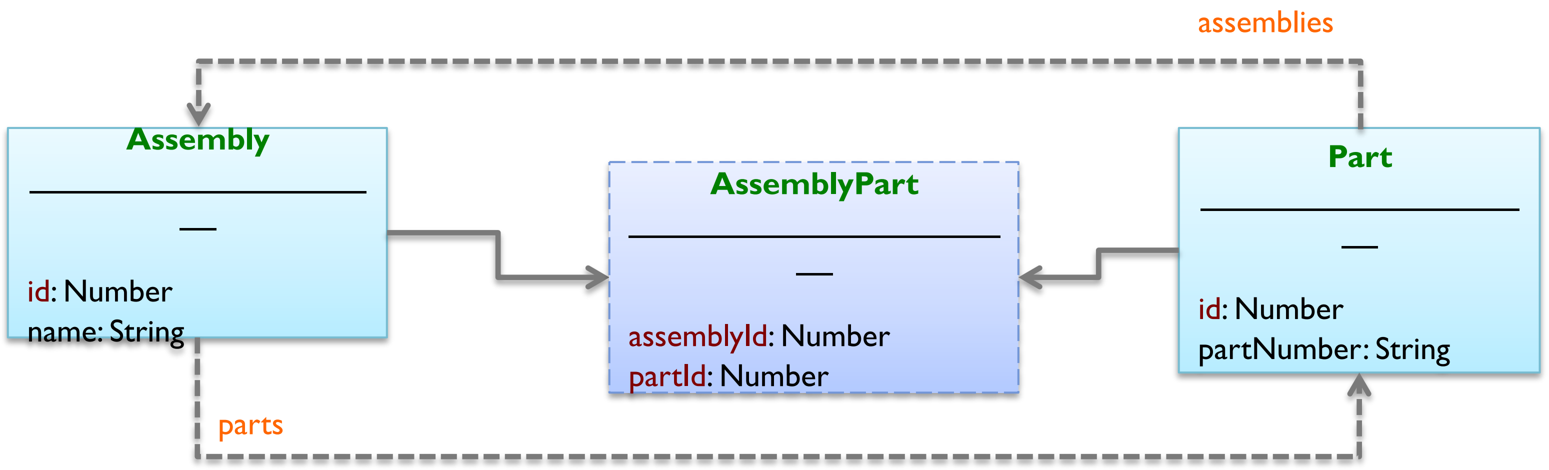
```
var Order = ds.createModel('Order', {
    customerId: Number,
    orderDate: Date
});

var Customer = ds.createModel('Customer', {
    name: String
});

Customer.hasMany(Order, {as: 'orders', foreignKey: 'customerId'});

...

customer.orders(filter, callback); // Find orders for the customer
customer.orders.build(data); // Build a new order
customer.orders.create(data, callback); // Create a new order for the customer
customer.orders.destroyAll(callback); // Remove all orders for the customer
customer.orders.findById(orderId, callback); // Find an order by id
customer.orders.destroy(orderId, callback); // Delete and order by id
```

hasMany through



```
var Physician = ds.createModel('Physician', {name: String});
var Patient = ds.createModel('Patient', {name: String});
var Appointment = ds.createModel('Appointment', {
    physicianId: Number,
    patientId: Number,
    appointmentDate: Date
});
Physician.hasMany(Patient, {through: Appointment});
Patient.hasMany(Physician, {through: Appointment});

physician.patients(filter, callback); // Find patients for the physician
physician.patients.build(data); // Build a new patient
physician.patients.create(data, callback); // Create a new patient for the physician
physician.patients.destroyAll(callback); // Remove all patients for the physician
physician.patients.add(patient, callback); // Add an patient to the physician
physician.patients.remove(patient, callback); // Remove an patient from the physician
physician.patients.findById(patientId, callback); // Find an patient by id
```

hasAndBelongsToMany



```
var Assembly = ds.createModel('Assembly', {name: String});
var Part = ds.createModel('Part', {partNumber: String});
Assembly.hasAndBelongsToMany(Part);
Part.hasAndBelongsToMany(Assembly);

...

assembly.parts(filter, callback); // Find parts for the assembly
assembly.parts.build(data); // Build a new part
assembly.parts.create(data, callback); // Create a new part for the assembly
assembly.parts.add(part, callback); // Add an part to the assembly
assembly.parts.remove(part, callback); // Remove an part from the assembly
assembly.parts.findById(partId, callback); // Find an part by id
assembly.parts.destroy(partId, callback); // Delete and part by id
```