REST API Worksheet: Blogging API

In this worksheet you will use Restify and CouchDB to create a REST API for a blogging platform.

Links:

- Restify documentation. - http://mcavage.me/node-restify/
- Nano documentation. - https://github.com/dscape/nano

Design

In order to design a RESTful API, you need to analyze the types of data you will use. Then you can define the most appropriate resources / URL endpoints. In a Blogging API for example, you may need:

- Articles
- Comments
- Users

Our example will implement the `Articles` resource, and you will add `Comments` and `Users` as you go.

HTTP methods are used to perform *operations* on the API resources:

HTTP Method	URL	Operation
GET	/articles	List all articles
POST	/articles	Add a new article
PUT	/articles/123	Update article 123
GET	/articles/123	View article 123
DELETE	/articles/123	Delete article 123

That gives us the design for the `Articles` resource.

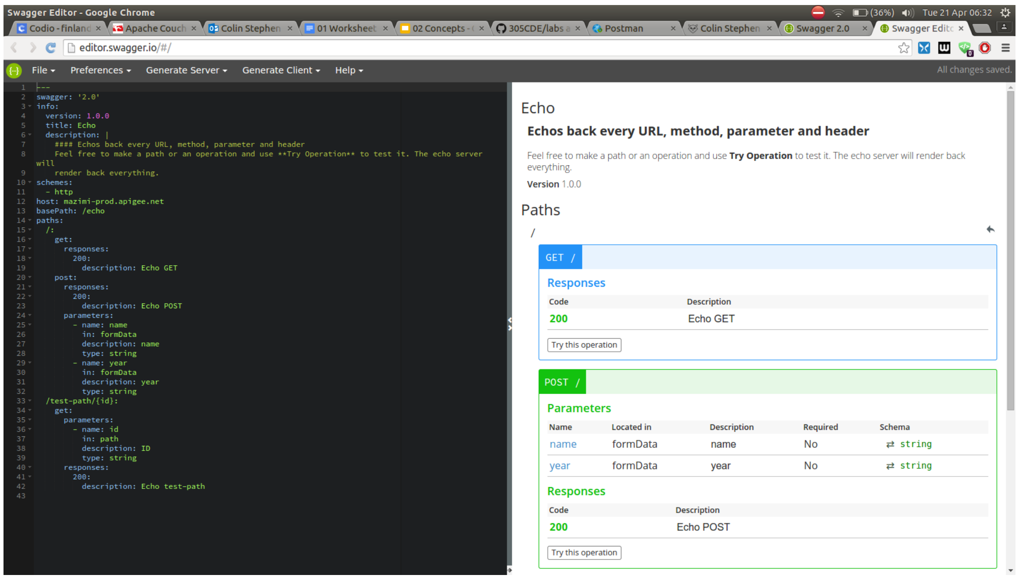
Design Task

- Design the *methods, URLs and operations* you will use for the `Comments` and `Users` resources.

Swagger

The Swagger API documentation system helps present your API designs:

- http://editor.swagger.io.


```
---
swagger: '2.0'
info:
  version: 1.0.0
  title: Echo
  description: |
    #### Echos back every URL, method, parameter and header
    Feel free to make a path or an operation and use **Try Operation** to test it. The echo server will
    render back everything.
schemes:
  - http
host: mazimi-prod.apigee.net
basePath: /echo
paths:
  /:
    get:
      responses:
        200:
          description: Echo GET
    post:
      responses:
        200:
          description: Echo POST
      parameters:
        - name: name
          in: formData
          description: name
          type: string
        - name: year
          in: formData
          description: year
          type: string
  /test-path/{id}:
    get:
      parameters:
        - name: id
          in: path
          description: ID
          type: string
      responses:
        200:
          description: Echo test-path
```

Echo

Echos back every URL, method, parameter and header

Feel free to make a path or an operation and use **Try Operation** to test it. The echo server will render back everything.

Version 1.0.0

Paths

/

GET /

Responses

Code	Description
200	Echo GET

POST /

Parameters

Name	Located in	Description	Required	Schema
name	formData	name	No	⇄ string
year	formData	year	No	⇄ string

Responses

Code	Description
200	Echo POST

Figure 1: Swagger Interface

Design Task

- Load the Swagger editor and view some of the examples from the `File` menu.
- Code your API design in to Swagger (at least for the `Articles` resource)

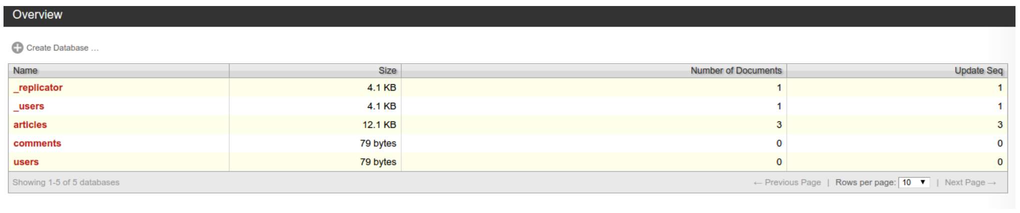
Database

An API needs somewhere to store its data. You will be using CouchDB.

Database Task

- Log in to your Codio account
- Launch a CouchDB instance
- Log in to the CouchDB GUI interface at `http://subdomain.codio.io:5984/_utils`
- Create databases for each of the resources you are going to use: `articles`, `comments`, `users`
- Populate your new databases with some sample documents using Postman REST Client in the Chrome browser
  - Choose appropriate JSON data structures for each resource

---



Name	Size	Number of Documents	Update Seq
_replicator	4.1 KB	1	1
_users	4.1 KB	1	1
articles	12.1 KB	3	3
comments	79 bytes	0	0
users	79 bytes	0	0

Showing 1-5 of 5 databases

Figure 2: Databases in CouchDB ready to store your API data

Code

Implementing a RESTful API is straightforward using NodeJS and CouchDB. You will use two NodeJS modules:

- `restify` is a simplified version of `express` and will serve your API
- `nano` is a simple interface to access your CouchDB database(s)

Code Tasks

1. Run the API server as follows:

- From the terminal in Codio, clone the git repository to get the example code for this lab:

  git clone https://gitlab.com/colinstephen/restify-api.git

- Open the `index.js` file and ensure you have completed the two `PREREQUISITE` lines to configure your environment.

  - `npm install restify`
  - `npm install nano`
  - your `articles` database should also be set up by now

- Read through the comments and code in `index.js` to understand what each part does

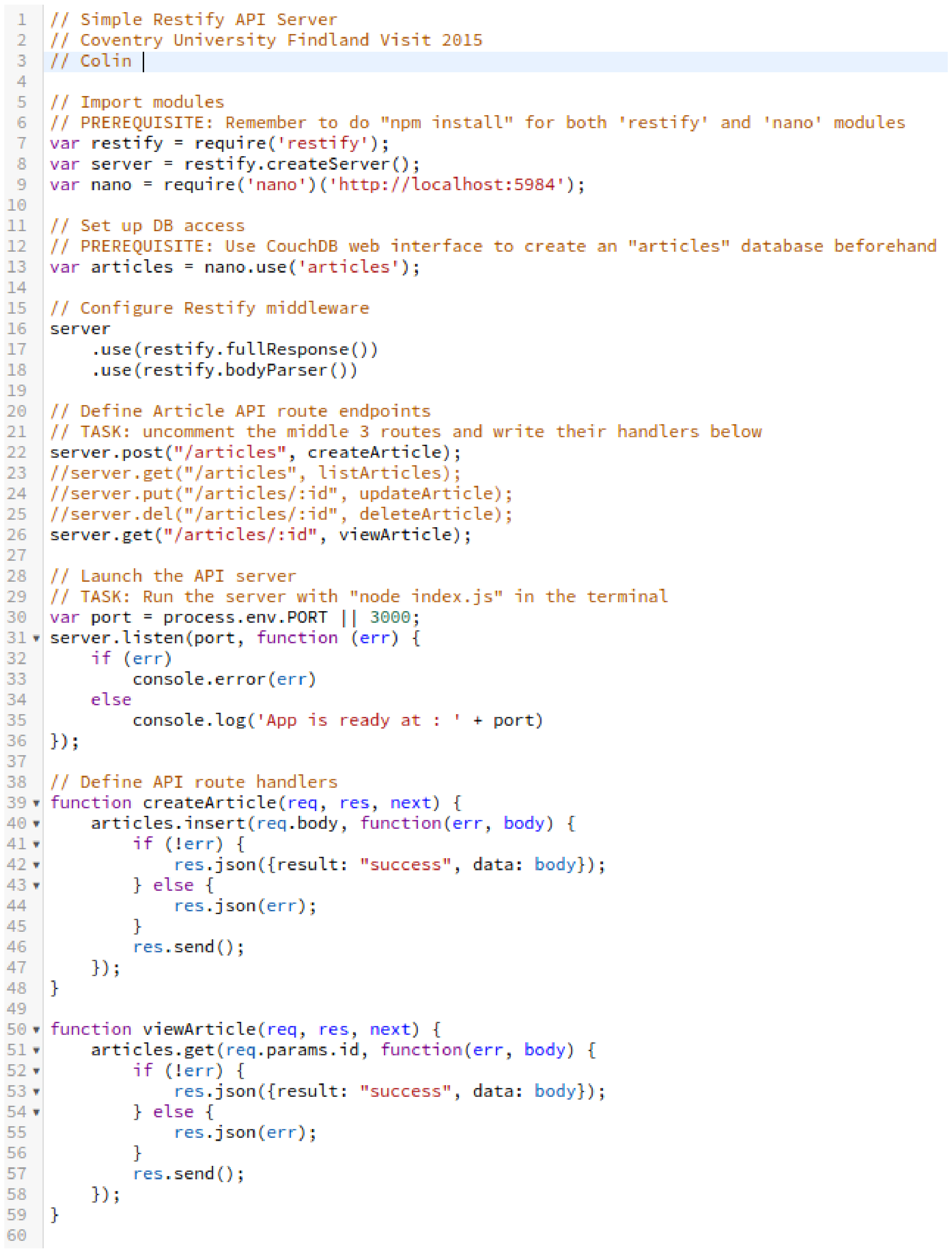
- From a terminal run the API server with `node index.js`

2. Test adding a new article and reading the data back

- Next use the Postman REST Client in another browser tab, to POST some JSON data to your `/articles` API endpoint

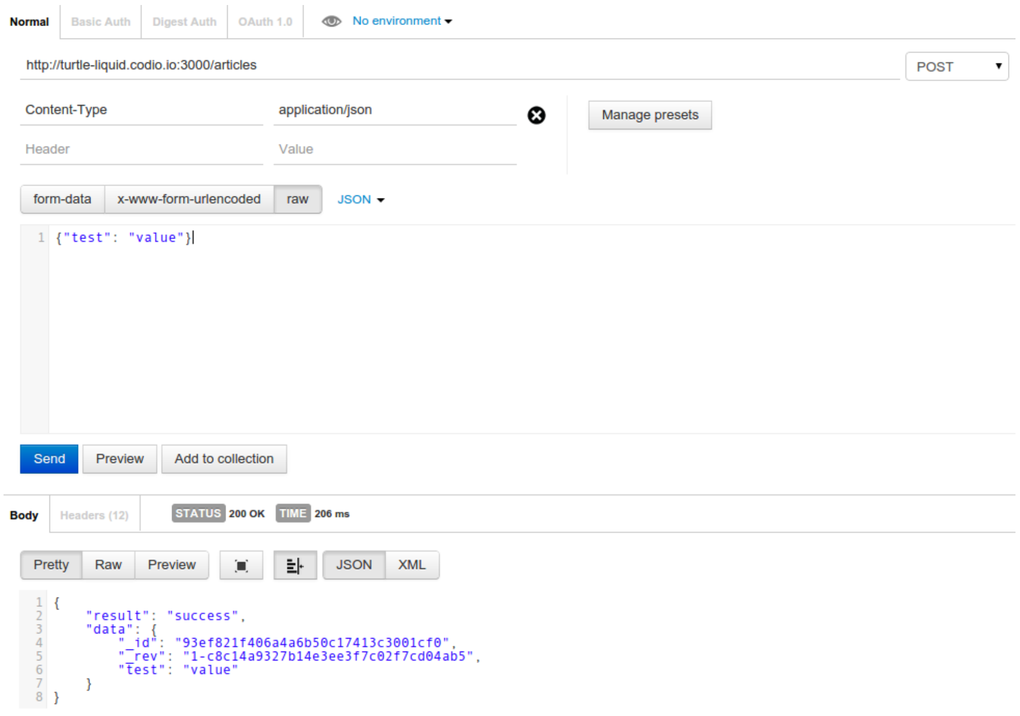
  - ensure you POST raw JSON data, not "form-data"
  - you also will need to the *header* for `Content-Type` is set to the value `application/json` for this to succeed

- Refresh the CouchDB management GUI to double check that the data was added to your `articles` database



```javascript
// Simple Restify API Server
// Coventry University Findland Visit 2015
// Colin

// Import modules
// PREREQUISITE: Remember to do "npm install" for both 'restify' and 'nano' modules
var restify = require('restify');
var server = restify.createServer();
var nano = require('nano')('http://localhost:5984');

// Set up DB access
// PREREQUISITE: Use CouchDB web interface to create an "articles" database beforehand
var articles = nano.use('articles');

// Configure Restify middleware
server
    .use(restify.fullResponse())
    .use(restify.bodyParser())

// Define Article API route endpoints
// TASK: uncomment the middle 3 routes and write their handlers below
server.post("/articles", createArticle);
//server.get("/articles", listArticles);
//server.put("/articles/:id", updateArticle);
//server.del("/articles/:id", deleteArticle);
server.get("/articles/:id", viewArticle);

// Launch the API server
// TASK: Run the server with "node index.js" in the terminal
var port = process.env.PORT || 3000;
server.listen(port, function (err) {
    if (err)
        console.error(err)
    else
        console.log('App is ready at : ' + port)
});

// Define API route handlers
function createArticle(req, res, next) {
    articles.insert(req.body, function(err, body) {
        if (!err) {
            res.json({result: "success", data: body});
        } else {
            res.json(err);
        }
        res.send();
    });
}

function viewArticle(req, res, next) {
    articles.get(req.params.id, function(err, body) {
        if (!err) {
            res.json({result: "success", data: body});
        } else {
            res.json(err);
        }
        res.send();
    });
}

```

Figure 3: Sample Restify Code (index.js)



http://turtle-liquid.codio.io:3000/articles POST

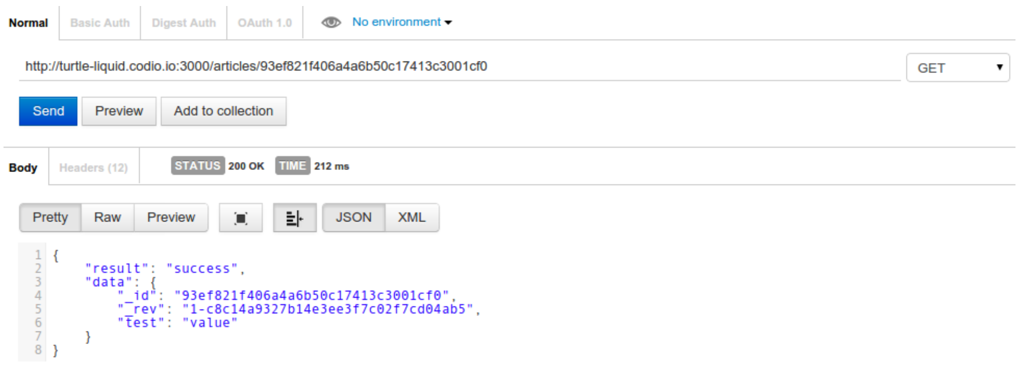
Content-Type: application/json

```
{"test": "value"}
```

STATUS 200 OK TIME 206 ms

```
{
    "result": "success",
    "data": {
        "_id": "93ef821f406a4a6b50c17413c3001cf0",
        "_rev": "1-c8c14a9327b14e3ee3f7c02f7cd04ab5",
        "test": "value"
    }
}
```

Figure 4: POSTing test article data to the API



http://turtle-liquid.codio.io:3000/articles/93ef821f406a4a6b50c17413c3001cf0 GET

STATUS 200 OK TIME 212 ms

```
{
    "result": "success",
    "data": {
        "_id": "93ef821f406a4a6b50c17413c3001cf0",
        "_rev": "1-c8c14a9327b14e3ee3f7c02f7cd04ab5",
        "test": "value"
    }
}
```

Figure 5: GETting an article from the API

- Copy the `_id` value of one of the records
- Use this in Postman to GET data from the `/articles/:id` API endpoint

3. Write your own article endpoints

- In `index.js` uncomment the three routes that are commented, between the two you just used
- Next, write function handlers for each of the routes:
  - `listArticles`
  - `updateArticle`
  - `deleteArticle`
  - Try to follow the model of the example code for `createArticle` and `viewArticle` to do this.

4. Write the API for Comments and Users

- If you have time, also implement the following:
  - endpoint URLs for the `comments` and `users` resources that you defined earlier
  - handlers for each of these