

Node, Restify and Fiddler Basics

Estimated time for completion: 45 minutes

Overview:

In this lab you will use Node.js and the Restify framework to expose simple movie data as JSON and XML over HTTP. You will then learn to use Fiddler to make HTTP requests to access the movie data.

Goals:

* Use the Restify framework to provide movie listings
* Use Fiddler to view and make HTTP requests

Lab Notes:

The lab uses Node.js.

* You can install Node.js from [http://nodejs.org](http://nodejs.org/)

Restify Basics

In this part of the lab, you will implement the GET methods for the Restify framework to return movie data.

Criteria:

* Modify the Server.js to provide movie listings and movie details (via GET)

Steps:

1. This lab is meant to be a very basic introduction to using the Restify framework to generate data.
2. Open the Server.js from ~/RestifyIntro/before. In this file you will add the basic REST server. The file Movies.js contains the move data to display.
3. Add Restify and Underscore to the project using NPM and require them in Server.js.
4. Add two routes to the path object. One for the complete set of movies at “/api/movies” and the other for a single movie at “/api/movies/:id”.
5. Now implement the two “get” methods: One for returning all the movie listings and another that returns a single movie based upon the “id” URL parameter.
6. Start the server.
   1. You might want to use a tool like [nodemon](https://github.com/remy/nodemon) during development to automatically restart the server for you after you saved changes.
7. To test, you can access the service from the browser at <http://localhost:8080/api/movies>. This should return a JSON formatted list of movies. Also try requesting individual movies.

**Helpful links:**

* + [Creating a Restify server](http://mcavage.me/node-restify/#Creating-a-Server)
  + [Find an element in an array using Underscore.js](http://underscorejs.org/#find)

Serving data in an XML format

In this part of the lab, you will expand the REST service so it can do content negotiation and return both JSON and XML formatted data.

Criteria:

* Add support for XML through Content-Negotiation
* Ensure a 406 error is returned when a not supported content type is requested.

Steps:

* Open the file Formatters.js and add the required code to convert the Movie objects to XML
* Update the Server.js code so it requires Formatters.js and specifies the formatters for the Restify server.
* Add the Accept header parsing plugin to return a 406 error status if an invalid request format is specifies.
* To test, you can access the service from the browser at <http://localhost:8080/api/movies>. As most browsers prefer XML over JSON this should return a XML formatted list of movies. Also try requesting individual movies.

**Helpful links:**

* + [Content Negotiation](http://mcavage.me/node-restify/#Content-Negotiation)
  + [The data2xml convertor](https://github.com/chilts/data2xml)
  + [Inflect custom inflections for nodejs](http://pksunkara.github.io/inflect/)
  + [Restify Bundled Plugins](http://mcavage.me/node-restify/#Bundled-Plugins)

Using Fiddler

In this part of the lab, you will install and use Fiddler to view and make HTTP requests.

Criteria:

* Use Fiddler to view HTTP requests
* Use Fiddler to make HTTP requests

Steps:

1. Fiddler is an excellent tool for debugging and monitoring HTTP traffic. In this part, you will install Fiddler (if it’s not already) and then use it to view and make HTTP requests to the movie service.
   1. If it’s not already installed, download Fiddler from [here](https://www.fiddler2.com/Fiddler2/version.asp) and run the installer.

**Helpful Links:**

* If you have issues with Fiddler, [here](http://www.fiddler2.com/Fiddler/help/) is the help and documentation page.

1. Next use Fiddler to monitor browser traffic.
   1. Launch fiddler and then use your browser to access the movie listings.
   2. Inspect the details of the various requests in Fiddler.
2. Next use Fiddler to monitor HTTP traffic from the browser.
   1. Launch Fiddler and then use the browser to access the movie listings.
3. Finally, use Fiddler to make HTTP requests.
   1. Use the “Composer” feature of Fiddler to make HTTP requests to the movie service.
   2. In the composer set the Accept HTTP request header to use both application/xml and application/json to see the difference in the response.

The header should look like:

Accept: application/xml

Solutions:

The final solution for this lab is available in the ~/after directory.