# SER422 Spring 2016 Lab 6 – REST Due: 11:59pm, Friday, April 29, 2016 at 11:59pm

## Activity 1 (20%):

For each of the REST APIs indicate which level of Richardson’s Maturity Model you think that API has achieved and why

1. <https://developer.github.com/v3/>
2. <http://www.scrumwise.com/api.html>

Next, go find your own (2) examples of a Level 1 API and a Level 2 or 3 API. Explain why the APIs you chose are at the given level.

Answers go in a Word doc.

## Activity 2 (80%):

You will implement your own REST API based on the Booktown DAO example of Lab 3.

Specifically, implement API behaviors to (note I am not telling you more than this!):

1. **Create an Author**
2. **Create a Book (note it must have at least one Author)**
3. **Update an Author’s first or last name**
4. **Update a Book’s publisher, title, or year**
5. **Add an Author to a Book**
6. **Delete a Book**
7. **Delete an Author (only if no Book with that Author)**
8. **Retrieve an Author by ID**
9. **Retrieve a Book by ISBN**
10. **Retrieve a collection of Books by Title substring**

Some very important things to note:

1. You may implement your REST API in Java or in Node.js. You may use whatever framework you like, or none at all. It is entirely possible, and even not that hard, to implement a REST service using the vanilla servlet APIs or basic Node.js http features. But you are free to use a framework. I suggest using Apache Jersey for Java implementations and Restify for Node implementations, but again it is up to you. Express is actually a decent starting point for REST implementations as well
2. I am not asking you to re-use your DAOs from Lab 3, although you may choose to do so if you wish. I do not care about ValueObjects or multiple persistent store implementations. I care that you have a proper REST API implementation.
3. You may use Apache Derby or MongoDB as a database. You may also simple use 1 or more files as the persistent store.
4. Return format should be some kind of JSON. Beyond that is up to you as a matter of design.
5. I expect you will achieve a Level 2 API. Level 0 APIs will receive a 20% deduction. Level 1 APIs will receive a 10% deduction. Level 3 APIs will receive a 20% bonus.
6. You will be graded as much on your design of the API as the correctness of the code. That is, I will evaluate your API design against the Top 10 best API design practices.
7. Note there is no web application here. I will test your API using a REST client like Postman or ARC.
8. Authentication and Authorization is a topic I would like to cover but we will not have time. For each API (News or Match) you may safely assume that the caller has full permissions to execute the specified action on the specified resource(s).
9. Your lab submission must DEPLOY AND RUN WITHOUT ERROR.
10. As it is the end of the semester I will not be entertaining appeals or corrections to the submissions, however minor, after the due date.

## Extra Credit (40 points):

For extra credit I give you 3 options. YOU CAN ONLY DO ONE OF THE OPTIONS – Usability study, AJAX app, or REST tool!

1. Usability study: You have to participate in one of the usability sessions Thursday the 21st for approximately 90 minutes.
2. Write a web application that uses methods 1, 2, 8, 9, and 10 of your REST API. This application should use AJAX to make calls to your REST API and dynamically update web page(s). In lab 3 there were several screens, you should start with those, although I really do not care what the screens look like as long as you make the AJAX call and DOM manipulate a page.
3. Use a tool such as Restlet Studio (<https://restlet.com/products/restlet-studio/>), Swagger (SwagEdit, <https://github.com/RepreZen/SwagEdit>), or API Blueprint’s Aton extensions (<https://apiblueprint.org/tools.html>) to design APIs 1, 2, 8, 9, and 10 and automatically generate the starting code for them (you can do the whole lab this way if you want).

**SUBMISSION**:

1. Your submission should be named lab1.<asurite1>[.<asurite2>].jar for upload to Blackboard. Only include your Word doc, source tree, and build files, not your compiled code and other runtime artifacts (you can include a lib directory).