Movie App Tutorial

# General Introduction

This tutorial walks through the creation of a complete web application, [Neo4j Movies](http://neo4jmovies.herokuapp.com/#/movies), a Neo4j-Swagger-AngularJS version of Cineasts.net, a social movie database where users can connect with friends, rate movies, share scores, and generate recommendations for new friends and movies.

This tutorial takes the reader through the steps necessary to create the application, explaining each step of the stack on the way. The complete source code for the app is available on [GitHub](https://github.com/kbastani/neo4j-movies-template).

# The Stack: An Overview

## Neo4j

Written in Java since YEAR, [Neo4j](http://neo4j.org/" \t "_top) is a scalable, a fully transactional database (ACID) that stores data structured as graphs. Designed to be intuitive, high performance and scalable, it has a disk-based, native storage manager optimized for storing graph structures with maximum performance and scalability. Neo4j can handle graphs with many billions of nodes/relationships/properties on a single machine, but can also be scaled out across multiple machines for high availability.

## Swagger

Developed by Wordnik, Swagger™ defines a standard, language-agnostic interface to REST APIs which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection. When properly defined via Swagger, a consumer can understand and interact with the remote service with a minimal amount of implementation logic.

## AngularJS

“What HTML should have been”, AngularJS is an open-source web application framework, assists in the creation of web applications that only require HTML, CSS, and JavaScript on the client side. Its goal is to augment web applications with model–view–controller (MVC) capability, in an effort to make both development and testing easier. AngularJS' two-way data binding is its most notable feature and reduces the amount of code written by relieving the server backend of templating responsibilities. Instead, templates are rendered in plain HTML according to data contained in a scope defined in the model.

# The Domain Model

The Neo4j data model consists of nodes and relationships, both of which can have key/value-style properties. What does that mean, exactly? Nodes are the graph database name for records, with property keys instead of column names. That's normal enough. Relationships are the special part. In Neo4j, relationships are first-class citizens. More than a simple foreign-key reference to another record, relationships carry information. So we can link together nodes into semantically rich networks.

Make some UML type diagrams

Hint at <https://github.com/kbastani/neo4j-movies-template/tree/master/api/models/neo4j> talk more about this topic in the Swagger section.

# Neo4j: Setting up the Database

## Neo4j: Getting it Running

* If you haven’t done so already, download Neo4j [here](http://www.neo4j.org/download)
* Extract Neo4j to a convenient location and rename the folder to something less cumbersome, like ‘Neo4j’, if you want
* Navigate to the extracted folder and run ./bin/neo4j start
* If all goes well, you should see the Neo4j web application running at <http://localhost:7474/>

To try out Neo4j on your local machine, an empty database is not much fun.

* Navigate to your Neo4j directory
* If you have Neo4j running, stop it with ./bin/neo4j stop in the Neo4j directory
* If you want to make sure you killed it good, check by running launchctl list | grep neo and launchctl remove any processes that might be listed
* If you ls data, you’ll see a file called graph.db.
* Delete the existing graph.db.
* Grab the zipped movies graph database file from the databases folder in the [web app repository](https://github.com/kbastani/neo4j-movies-template)
* Unzip it into the data folder
* Run Neo4j! You should be able to see some nodes at <http://localhost:7474/>

## Cypher: An Introduction

Get started with Cypher on the Neo4j [Learn Cypher](http://www.neo4j.org/learn/cypher) page.

Don’t know how much to put in this section.

## Building the Database

TODO:

Create CSV for use in this tutorial from the Kenny database.

### Using LOAD CSV

Don’t know how much to put in this section.

#### Resources

## Using batch-import

I think this is the best method by far to get data in. Either way, the user has to create CSVs. It’s a lot less work and faster than using LOAD CSV imo.

## Testing in the Console

## Testing in the Web Dashboard

# Swagger: Querying the Database

## Node-Neo4j-Swagger API: An Introduction

The Node-Neo4j-Swagger API was written to make it as easy as possible to create an API using Node.js and Neo4j that can be consumed by some other app. Swagger provides interactive documentation so that it is easy to interact with the API. The goal is merge Swagger with Neo4j queries and visualizations so developers can see how Neo4j and the API results relate to each other.

## Building the Models

### Filling the Models with Cypher

## Building the Routes

## Building the Views

# AngularJS: Building the Website

## Why AngularJS

# References

## Neo4j

## Swagger

[The Swagger Spec](https://github.com/wordnik/swagger-spec)

## NodeJS

## AngularJS