The Cities Demonstration Application

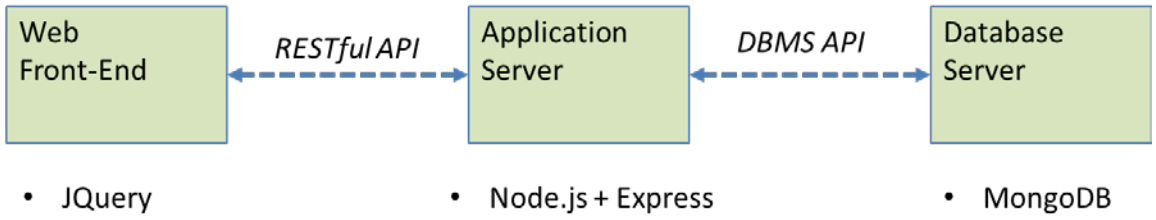
Overview

The Cities demonstration application provides a view of some key statistical data for major United States cities:

- Name of the city
- State
- Population
- Crime Index
- Cost Index

Note that the HTML5/CSS3 layout is intentionally very plain. The goal here is to demonstrate connectivity between technologies, not to demonstrate web design artistry. As such, the HTML structure is kept as simple as possible to make it easy to read and understand the source code.

Structure

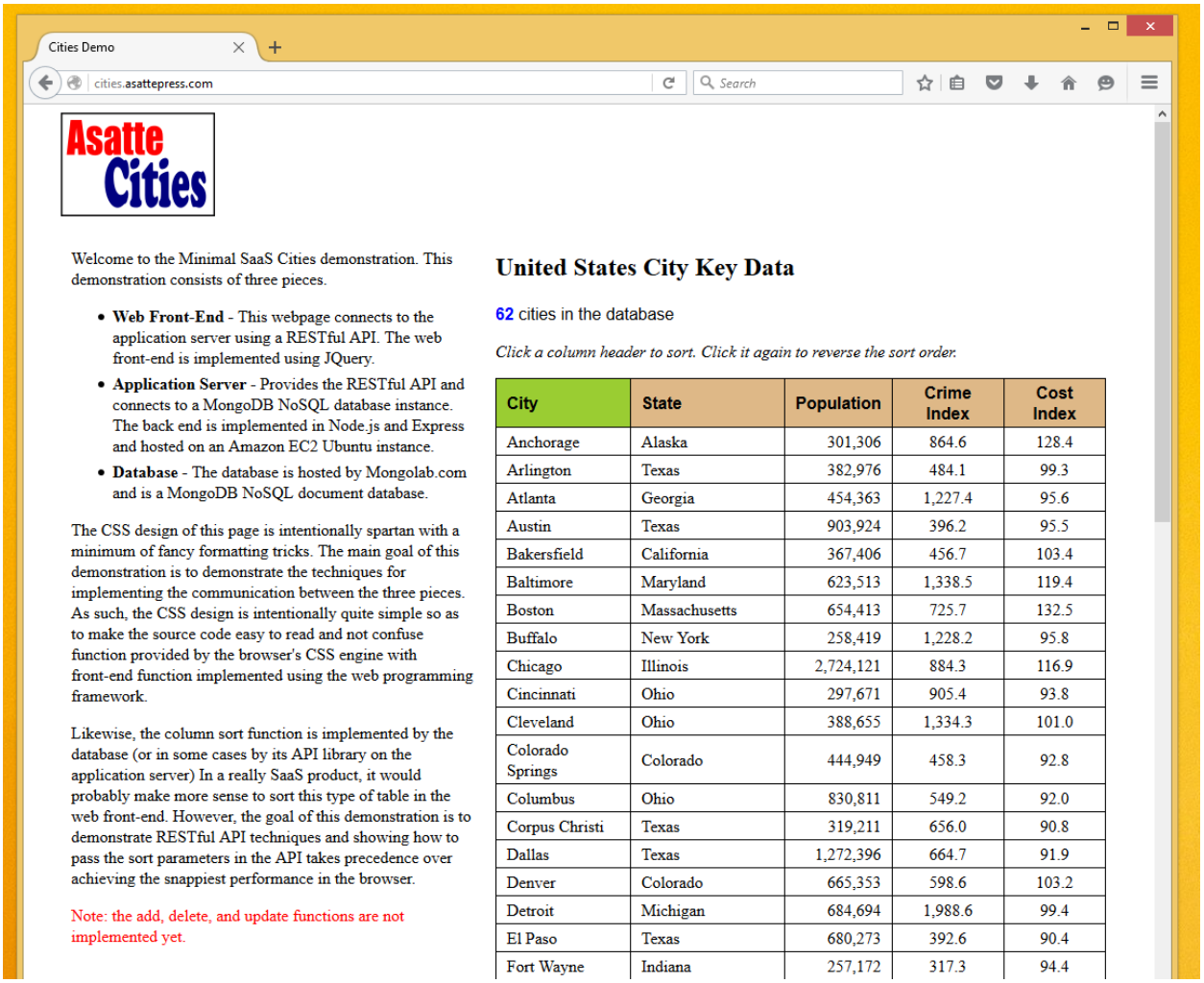


The application is structured as three layers:

1. **Web Front-End** – Currently implemented as HTML5/CSS3 and JQuery and hosted on CentOS at HostGator.com. *Future:* Angular.js, Ember.js, perhaps some older technologies like JSP for comparison.
2. **Application Server** – The main purpose of the application server is to provide a clean (and secure) RESTful API for the web front-end and interface to the different APIs provided by different database technologies. Currently this application is implemented in Node.js and Express running under Ubuntu Linux on an Amazon EC2 instance. *Future:* Java under Spring, perhaps a Native C++ application.
3. **Database Server** - The database server obviously hosts the data about the cities. Currently this database is a MongoDB instance – MongoDB being a leading "NoSQL" database – hosted by Mongolab.com which actually a reseller of Amazon storage services. *Future:* Amazon RDS, Cassandra (somewhere), Amazon DynamoDB, Google CloudSQL, Azure?

The Application

The live demonstration can be accessed at: http://cities.asattepress.com



When the application first loads, the table is empty. The browser status line will show waiting for a second or two after which the table will populate. Basically, the webpage loads with an empty table and then executes the script Cities.js which uses JQuery and Javascript to call the RESTful API to retrieve the data from the database. Once the data is returned, the table populates. After the table has loaded, the user can click on any column header to sort the data by that column. Continuing to click on the same column header will reverse the sort order.

GITHub

The source code for the project can be accessed at: https://github.com/DavidHetherington

Note that for security reasons, the GitHub versions have had the security credentials removed. If you would like to setup your own copy, you will need to spin up your own instances of each piece and edit the security credentials, URLs, etc… as needed.

- **MinimalSaaS -** Just the documentation. Each piece has a separate repository.
- **Cities-WebPage-JQuery –** The HTML, CSS, and JavaScript pieces for the front-end
- **Cities-BackEnd-Node.js –** The Node.js and Express implementation for the application server as well as some simple notes on deploying to an Amazon EC2 instance
- **MongoLoad –** A simple Java program for loading the initial cities data into the MonoDB instance

The API

As of Version 0.6 the API supports only two functions, both of them using HTTP GET. For the moment, the functions respond to both JSON and JSONP requests which gets around the cross-domain security problem. Future releases will add CORS support and PUT, POST, and DELETE functions.

Count

Returns the count of cities in the database.

Get count of cities (JSON)	http://localhost:3000/api/cities/count
Get count of cities (JSONP)	http:// localhost:3000/api/cities/count?callback=?

List

Returns a list of the cities in the database.

The basic list API supports two parameters:

- **sort =** Which column to sort by. The default (if the parameter is missing or unreadable) is to sort by the "city" column. Valid choices are:
  - **city** – sort on the city column
  - **state** – sort on the state column
  - **population** – sort on the population column
  - **crime** – sort on the crime index column
  - **cost-of-living** – sort on the cost index column

- **direction =** Direction of sort. The default (if the parameter is missing or unreadable) is to sort "up". Valid choices are:
  - **up** – sort from smallest to largest
  - **down** – sort from largest to smallest

Get list of cities (JSON)	http://localhost:3000/api/cities/list?sort=city&direction=up
Get list of cities (JSONP)	http://localhost:3000/api/cities/list?sort=city&direction=up&callback=?

Release Notes

Version 0.5 – 13 January 2016

This is the first deployed version. All that works currently is the populating of the table. *Next Steps:* implement sorting by clicking on column headers. Implement a second small table on the left that dynamically shows which technology is being used for each piece.

Version 0.6 – 4 February 2016

Add the ability to sort the table by columns. Involves changes to both the JQ