

### The Cities Demonstration Application

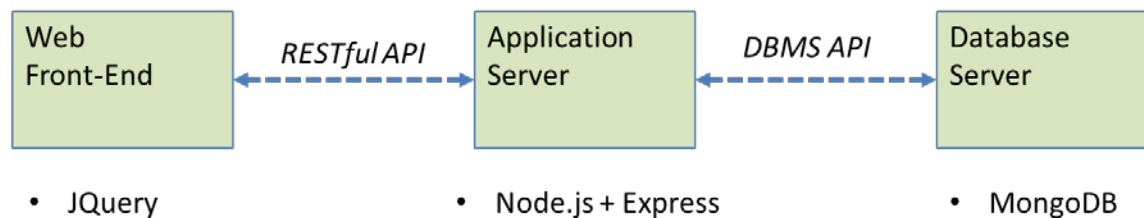
#### Overview

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>

The screenshot shows a web browser window titled "Cities Demo" with the URL "cities.asattepress.com". The page features the "Asatte Cities" logo in the top left. Below the logo, a welcome message states: "Welcome to the Minimal SaaS Cities demonstration. This demonstration consists of three pieces." This is followed by a bulleted list of the three components: Web Front-End, Application Server, and Database. To the right of this list, a section titled "United States City Key Data" indicates that 62 cities are in the database and provides instructions on how to sort the data by clicking on column headers. Below this text is a table with five columns: City, State, Population, Crime Index, and Cost Index. The table lists 20 cities, starting with Anchorage, Alaska, and ending with Fort Wayne, Indiana. A note at the bottom left of the page states: "Note: the add, delete, and update functions are not implemented yet."

Welcome to the Minimal SaaS Cities demonstration. This demonstration consists of three pieces.

- **Web Front-End** - This webpage connects to the application server using a RESTful API. The web front-end is implemented using JQuery.
- **Application Server** - Provides the RESTful API and connects to a MongoDB NoSQL database instance. The back end is implemented in Node.js and Express and hosted on an Amazon EC2 Ubuntu instance.
- **Database** - The database is hosted by Mongolab.com and is a MongoDB NoSQL document database.

The CSS design of this page is intentionally spartan with a minimum of fancy formatting tricks. The main goal of this demonstration is to demonstrate the techniques for implementing the communication between the three pieces. As such, the CSS design is intentionally quite simple so as to make the source code easy to read and not confuse function provided by the browser's CSS engine with front-end function implemented using the web programming framework.

Likewise, the column sort function is implemented by the database (or in some cases by its API library on the application server) In a really SaaS product, it would probably make more sense to sort this type of table in the web front-end. However, the goal of this demonstration is to demonstrate RESTful API techniques and showing how to pass the sort parameters in the API takes precedence over achieving the snappiest performance in the browser.

Note: the add, delete, and update functions are not implemented yet.

### United States City Key Data

62 cities in the database

Click a column header to sort. Click it again to reverse the sort order.

City	State	Population	Crime Index	Cost Index
Anchorage	Alaska	301,306	864.6	128.4
Arlington	Texas	382,976	484.1	99.3
Atlanta	Georgia	454,363	1,227.4	95.6
Austin	Texas	903,924	396.2	95.5
Bakersfield	California	367,406	456.7	103.4
Baltimore	Maryland	623,513	1,338.5	119.4
Boston	Massachusetts	654,413	725.7	132.5
Buffalo	New York	258,419	1,228.2	95.8
Chicago	Illinois	2,724,121	884.3	116.9
Cincinnati	Ohio	297,671	905.4	93.8
Cleveland	Ohio	388,655	1,334.3	101.0
Colorado Springs	Colorado	444,949	458.3	92.8
Columbus	Ohio	830,811	549.2	92.0
Corpus Christi	Texas	319,211	656.0	90.8
Dallas	Texas	1,272,396	664.7	91.9
Denver	Colorado	665,353	598.6	103.2
Detroit	Michigan	684,694	1,988.6	99.4
El Paso	Texas	680,273	392.6	90.4
Fort Wayne	Indiana	257,172	317.3	94.4

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.

## MinimalSaaS – Cities Demonstration Application

---

### 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 MongoDB 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