developerWorks®

Build your first Node.js website, Part 1

Ruth Willenborg

August 20, 2015
(First published July 29, 2014)

Do you want to build a dynamic website with Node.js but are unsure where to start? This three-part article introduces you to Node.js development step by step, with no software installation required. Using only a web browser, you create a Node.js application, write server-side JavaScript, display dynamic data on HTML pages, and store data in a Redis database.

To view the introductory video **Build your first Node.js website, Part 1** , please access the online version of the article. If this article is in the developerWorks archives, the video is no longer accessible.

When my 9-year-old son wanted help writing his first app, I figured we would build a few web pages displaying some information that we'd store in a database. I hadn't written code in many years, but how hard could it be to create this simple website?

Well, after we consulted various instructional websites and books — and I installed a lot of stuff that I really didn't want cluttering up my computer — I couldn't believe how hard it was.

So I decided to give IBM Bluemix™ a try. It was easy to get a sample application running, but then I found myself wading through a bunch of code that I didn't understand and installing yet more stuff for developing applications — nothing I was ready to subject a 9-year-old to.

My love for Bluemix grew the day I discovered the web IDE on IBM Bluemix DevOps Services. You can edit an application in the IDE, push it to Bluemix, and run the app — all with no additional software to install. Of course, you can use Bluemix with local development environments such as Eclipse, and with the Bluemix command line. But for getting started and quickly trying out new things, the web IDE is a perfect fit.

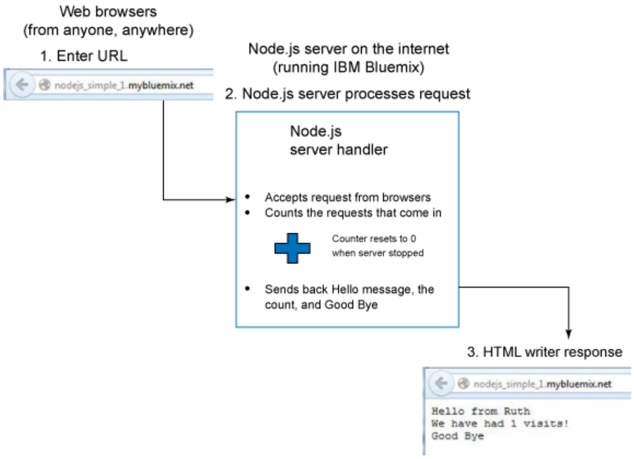
Still, all I wanted to do to start was put up an HTML page, yet the Bluemix Node.js web starter includes nothing that looks like good old-fashioned HTML. As for working with a database, the Bluemix sample ToDo application was more than I needed.

This article's examples are easy enough for even primary-school students to follow — not young first-timers (who'll do better with no-typing languages such as Scratch), but kids who are ready for their next step into programming. For each major concept in this article, the worksheets folder in the Bluemix DevOps Services project contains corresponding printable cards that follow CoderDojo Sushi Card guidelines.

That's what inspired this three-part article, which starts from the bare minimum and takes you step-by-step through building up a dynamic website with Node.js— all with no installation required. This first part covers bringing up a Node.js server for simple request handling. I started with a classic Hello World sample and added a server-side counter. You'll create a copy of the application for your own use, deploy the app on Bluemix, and then make some code changes and see their effect.

> *“ The Node.js application serves requests from multiple browsers and keeps a count of each request. ”*

The Node.js application serves requests from multiple browsers and keeps a count of each request:


Web browsers
(from anyone, anywhere)
1. Enter URL
nodejs_simple_1.mybluemix.net
Node.js server on the internet
(running IBM Bluemix)
2. Node.js server processes request
Node.js
server handler
Accepts request from browsers
Counts the requests that come in
Counter resets to 0
when server stopped
Sends back Hello message, the count, and Good Bye
3. HTML writer response
nodejs_simple_1.mybluemix.net

```
Hello from Ruth
We have had 1 visits!
Good Bye
```

Part 2 adds the Express framework, formatting of HTML pages, and passing data from the Node.js server for display in an HTML page. Part 3 shows how to use a Redis database to persist the data that the website displays.

What you'll need

As promised, you don't need to install anything. You do need a Bluemix account.

Get the code

Step 1: Copy the Node.js application

1. Log in to Bluemix DevOps Services.
2. In Bluemix DevOps Services, click the **FORK PROJECT** button on the menu to create your own copy of the code. Provide a unique name, and click the **CREATE** button.
3. Click the **EDIT CODE** button to begin working with the application.

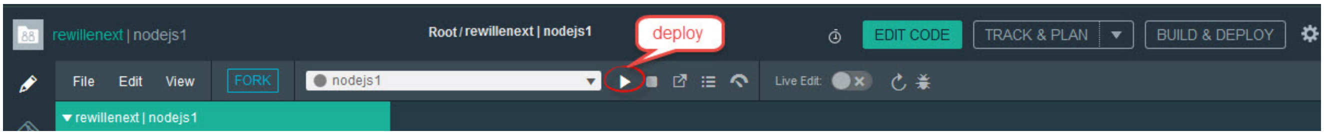
Step 2: Deploy the application manually

Bluemix DevOps Services deployment to Bluemix uses the manifest.yml file, which is all set up in your project and requires no changes. In manifest.yml, the `runtime: node08` line tells the deployment process to use the Bluemix Node.js runtime. The `host:` value, `${random-word}`, generates a unique URL for your application to ensure that your deployment doesn't conflict with others.

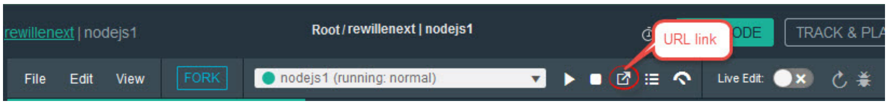
> To define your own easy to remember URL for your application, edit the Launch Configuration by clicking the upside down triangle in the toolbar, and selecting the pencil icon, and editing the configuration.

Bluemix DevOps Services provides both manual and automatic ways to deploy an application to Bluemix. You'll use the manual deploy:

1. In the Bluemix DevOps Services IDE, click the **DEPLOY** button:

2. Wait for the deployment to finish. A solid green dot and (running: normal) indicates that the deployment succeeded: If deployment problems occur, you can view the logs to troubleshoot.

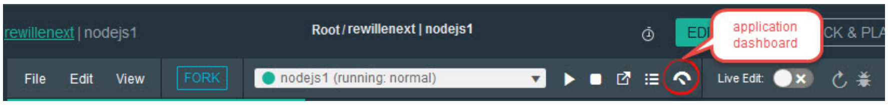
   

3. Click the **NodeJS_Simple_1** link to view your running application in a separate browser tab. Refresh the browser page and watch the counter increment with each refresh. (Depending on your browser, the counter may initially increment by more than 1.)

Step 3: Manage your application

You can manage your Bluemix deployments, including starting and stopping them, from Bluemix DevOps Services:

1. In Bluemix DevOps Services, follow the **application dashboard** link to manage the application in Bluemix:

2. In the Bluemix application dashboard, click the **Stop** button.
3. Click the **Start** button.
4. Follow the **Route** link to reopen the application in a separate tab. Notice that the counter is reset. It resets because the application is stateless and doesn't persist any data when it restarts. (Part 3 addresses persistence.)

Additional management capabilities are available from the Bluemix dashboard interface.

Step 4: Modify the application

> If you are going to be making a lot of editing changes, turn on Live Edit and use the Live Edit Restart to view your changes without redeploying.

1. Back in the Bluemix DevOps Services IDE, click **app.js** to open the Node.js code for the application.

2. In the line that says `res.write('Hello from Ruth\n');`, change `Ruth` to your name.
3. Deploy the app as you did in Step 2 and open the application in your browser. Observe that the page now reflects the changed text.
4. Return to app.js in Bluemix DevOps Services. Under `var userCount = 0`, add a new `userbytwo` variable and initialize it to `0`.
5. Under the line that increments `userCount`, add a line to increment `userbytwo` by 2:

   ```
   userbytwo = userbytwo + 2;
   ```

6. Copy and paste the write statement for `userCount` and modify it to print out `userbytwo`:

   ```
   res.write('We can also count by two. We have had '+userbytwo+ visits!\n');
   ```

   Your app.js code subsection should now look like this:

   ```
   var userCount = 0;
   var userbytwo = 0; /* added the var definition for new variable, userbytwo here */

   /**
    * This is the function that handles incoming requests
   **/
   var serverHandler = function(req,res) {
        userCount++;
        userbytwo = userbytwo + 2;
        /* added the incrementing by two for your new variable, userbytwo here */
        res.write('Hello from yourname\n');     /* edit this line to say your name */
        res.write('We have had '+userCount+' visits!\n');
        res.write('We can also count by two. We have had '+userbytwo+' visits!\n');
        res.end('Good Bye');
   };
   ```

7. Deploy the application and use the URL link to open it.
8. Confirm that the counter increments by 2.

Conclusion to Part 1

In this first part, you worked step by step through bringing up a Node.js server. You are now ready to move to Part 2, where you'll add a web front end with both static pages and dynamic content from the server. Part 3 covers storing the site's data in a database.

Related topics

- See IBM Bluemix in action
- Develop and deploy a Node.js app
- Getting started with Node.js Applications
- Bluemix Sample: ToDo Apps Built Using Your Favorite Language
- The SDK for Node.js runtime
- Node.js
- JavaScript