Web Development: Module 2, Lesson 6  
Moving to Azure Table Storage and Developing React front-end Hands-On Lab

## Overview

Building on Module 4 Lesson 4 Lab, we will implement a persistent storage solution for a blog. Without a persistent storage solution, all your data will be lost every time your server crashes or you need to restart it. Obviously, people won't tolerate such loss, so here we will implement persistent storage and a front-end UI.

## Objectives

In this hands-on lab you will learn how to:

* Implement CRUD using Azure Table Storage NoSQL remote database (index.js)
* Implement React front-end application which displays blog posts and allows for creation of new posts and deletion of old posts

## Prerequisites

The following are required to complete this hands-on lab:

* A text editor
* Windows PowerShell, Mac Terminal, or some other shell with node.js and npm installed
* You should have completed Module 2 Lessons 1, 2, 3, and 4 as well as the Module 2 Lessons 1 and 4 Labs.

## Exercises

This hands-on lab includes the following exercises:

* Exercise 1: Moving to Azure Table Storage
* Exercise 2: Developing React Front-end

## Exercise 1: Moving to Azure Table Storage

In this exercise, you will move locally stored data into Azure table storage.

NOTE: The following resource should prove helpful:

* <https://github.com/github/fetch>

1. Enter the Azure portal (<https://portal.azure.com>). You should have a subscription from the Module 1 Lesson 1 Lab.
2. Create a new project folder, and copy your lab2/index.js code into it
3. Create start.sh, add it to .gitignore and copy your key and storage name into start.sh
   1. start.sh should look like this:

AZURE\_STORAGE\_ACCOUNT=name AZURE\_STORAGE\_ACCESS\_KEY=key node index.js

1. Use code/lab3/package.json to configure babel
2. Use code/lab3/package.json and npm i to install azure-storage, react, react-dom, babel-cli, babel-preset-es2015, and babel-preset-react
3. Modify index.js (Node/Express server code) to work with Azure storage so that each route such as GET, POST, PUT and POST work with the database and not with the in-memory array
4. Use npm test to verify that your server is working
5. Compare your solution with lab3/express.js
6. Add static middleware to serve content from public.

## Exercise 2: Implementing a React Front-end

In this exercise, you will implement the front-end UI for a small blog using React.

1. Implement React app (components/app.js) which fetches existing posts from the RESTful API using fetch, and shows them in a list
2. Implement in your React app functionality to send POST requests to create a new blog post
3. Implement in your React app functionality to send DELETE requests to remove blog posts
4. Use npm run build or npm run build-watch to compile JSX
5. Compare your solution with lab3/components/app.js.

## Summary

In this hands-on lab, you learned how to:

* Transfer data into Azure Table Storage
* Use React.js to implement the front-end for a simple blog