Introduction

This lesson will give you a basic introduction to MEAN stack and go over the database structure used in this chapter.

WE'LL COVER THE FOLLOWING

- Introducing MEAN Stack

In the previous chapter we explained how to use MongoDB in .NET world. However, this is probably not the most natural habitat for MongoDB; so, I decided to display how one can use MongoDB in other, debatably, more appropriate environment of MEAN stack. After all, this database got its reputation by being used in the so-called MEAN stack.

The other reason we made this chapter is that, when we tried to learn MEAN stack quickly, we didn't find a lot of sources out there that gave us a full preview of how it should be done. Information was scattered all over the Internet and, what was there, usually covered just one part of the topic. So, we'll try to merge all that information into one chapter, using a simple CRUD example, to help you. This is a lot of ground to cover but, hopefully, it won't be too overwhelming. We will not go too deep into the details to make it easier to digest

Introducing MEAN Stack

MEAN is an acronym for using *four* technologies to create web applications:

- MongoDB
- Express
- Angular
- Node.js

Node.js and *Express* handle the backend of such applications, while *Angular* is in charge of the frontend.

> Note: All of the setup required in order to run the project, in this chapter, has already been done for you. If you want to learn how to install node.js for this application, as well as set up the environment for the front-end and back-end on your local machine, check out the appendix.

The biggest benefit of this structure is that all of the technologies involved are based in the JavaScript language. This way we can write the whole web application using just one notation.

We will explain each feature from front-end to back-end, and then we'll talk about MongoDB. The database structure itself will stay the same as it was in the last chapter.

> **Note:** Towards the end of this chapter, we will run the front-end and the back-end together.

We will use the database called – `blog`, with a collection called – `user`. This collection will contain a document for each *user* in the structure:

```
{
  "_id" : ObjectId("59ce6b34f48f171624840b05"),
  "name" : "Nikola",
  "blog" : "rubikscode.net",
  "age" : 30,


  "location" : "Beograd"
}
```

Before we get into the CRUD implementation, there are some preparations we need to do on both the front-end and the back-end. So, let’s dive into the front-end infrastructure implementation in the next lesson.