Full Stack Development with Angular and Spring Boot

Build scalable, responsive, and dynamic enterprise-level web applications

Sangeeta Joshi

www.bpbonline.com

First Edition 2025

ISBN: 978-93-65890-778

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true to correct and the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

Dedicated to

My father, ***Anil R. Kulkarni*** *whose unconditional support from the very beginning has been priceless*

Foreword

Over the past five years, I've immersed myself in Angular development, yet I consider myself a full-stack developer. Unfortunately, full-stack development often faces criticism, with phrases like "Jack of all trades" or "one cannot be a specialist in all disciplines" frequently surfacing.

While it's true that software disciplines are becoming increasingly complex, leading to specialization, at our core, we are developers. The foundation of any software developer's skill set includes the ability to build basic applications, encompassing the classic trio: frontend, backend, and databases.

In the ever-evolving and challenging world of software development, full-stack remains a constant. Data needs a place to be stored (databases), transformation and logic must occur (backend), and end-users need to interact with the application (frontend).

As a trainer, I meet many developers, and I often find that the best Angular developers are also proficient in backend development and oversee the application holistically, performing critical tasks requiring diverse expertise. More developers are embracing full-stack development as the norm. This trend aligns with enterprise applications, where UIs are often simple: forms and grids, processing and sending back data. Companies increasingly favor developers with both frontend and backend skills, and Angular-only developers report difficulties finding jobs without backend proficiency.

Even in large applications with hundreds of developers, the need for full-stack skills persists. Teams are often vertically sliced, with specialists available to assist. The ability to implement both frontend communication and backend endpoints accelerates feature development, a valuable asset for any company.

I am grateful to Sangeeta for her dedication towards creating this book, addressing a growing skill in our industry. I first met her at the Angular Meetup during Ng-conf 2023 in SLC. Her professionalism and composed demeanor stood out, underpinned by thorough research and profound insights.

Full-stack development forms the foundational common ground for developers. Congratulations on your intent to discuss full-stack development, and a big shoutout to Sangeeta for making this possible.

-Rainer Hahnekamp

GDE, Speaker, Trainer

About the Author

Sangeeta Joshi has more than 16 years of experience in the software industry. She has a proven track record of working as a technical manager, technology training expert, and subject matter expert at various MNCs. Sangeeta is an international speaker, a passionate trainer and technology enthusiast. She is an Ng-Champion and approved blog writer for angular technology on medium for NgConf channel. She has been recognized as distinguished contributor for NgConf 2024, the angular original world conference held in Salt Lake City, Utah. She has conducted more than 90 training programs on various technologies like Full stack Java, Angular, React js, Micro-services, spring boot, REST, Data- JPA, Pivotal Cloud Foundry etc., for MNCs across the globe, primarily in India, the USA, the UK and Canada. Sangeeta has recently successfully completed the postgraduate certification program in Artificial Intelligence and Machine Learning from Purdue University, in collaboration with Simplilearn and IBM. Sangeeta strongly believes that "learning is the key to success."

About the Reviewer

Amit Krishnakumar Deshpande is a seasoned Full Stack Java Developer with over 20 years of extensive IT experience in both software development and technical mentorship. Since 2001, he has held key technical roles in prominent organizations such as Accenture, Infosys, Wipro, Persistent, Cap Gemini, and Wiley. His expertise spans a wide array of technologies including Data Structures and Algorithms, Core Java, Advanced Java, OOP, Servlet JSPs, Spring Core, Spring MVC, Spring Boot, Data JPA, REST APIs, RDBMS, SQL, as well as front-end technologies like HTML5, CSS3, Bootstrap, Advanced JavaScript, TypeScript and Angular, React.

In addition to his development skills, Amit has a strong focus on corporate training, having conducted numerous technical workshops and training sessions globally, in countries like India, Japan, Israel, and Australia. He has trained over 10,000 professionals, significantly contributing to their technical growth.

Beyond his professional life, Amit enjoys playing sports such as badminton and table tennis. He is also a poet and a keen participant in philosophical discussions, continuously seeking knowledge in the field of spirituality.

Acknowledgement

First and foremost, I would like to thank my husband, Dr. Satish Joshi, for being patient and making it possible for me to focus on my writing by freeing me from my other responsibilities. I would like to extend my thanks to my daughter, Rujuta Joshi, a voracious reader, who continuously encouraged me to write this book. I would like to thank my mother, Nandini Kulkarni, who is delighted by my success no matter how small it may be. I am grateful to my entire family including Nikhil, my two sisters, my brother, and my in laws. I express my love to the little bundle of joy, my granddaughter Saeshma, for being a delight throughout the ups and downs of my book-writing journey.

I am grateful to Rainer Hahnekamp for accepting my request, taking time out from his busy schedule and writing a foreword.

Last but not least, I would like to extend my gratitude to all my friends who have always stood by me.

Preface

Distributed applications have been around for many years, taking a central role in software development. A distributed application comprises a variety of software components running on multiple computers or devices connected over a network. These components perform different tasks to achieve the application's targeted functionality. To ensure distributed systems are scalable, resilient, flexible, and performant, modern architectures and technologies are emerging. Consequently, developing various components of distributed applications involves different technologies. In other words, a stack of technologies is now essential for building such applications.

In the recent past, Full Stack Development has become an important segment of the developer community worldwide. A software developer with a variety of skill sets can replace two or more developers working on individual technologies within the required stack. Full Stack Developers are especially valuable as they need little or no outside help in their work. The unique advantage of Full Stack development is that it combines two essential parts of software application development: front-end and back-end technologies, into a single complete stack. Full Stack developers have the required skill set to work across the entire technology stack. They are proficient in developing front-end and back-end code and integrating them. They also possess data management skills using RDBMS or NoSQL databases. Software companies seek developers with Full Stack skills.

Angular is an ideal front-end framework for developing the web applications that modern businesses aspire to build. Characteristics of modern web applications, such as user experience, performance, flexibility, scalability, and rapid development are met by Angular.

Angular is one of the top web application development platforms, helping create efficient and sophisticated Single Page Applications. The framework has gained popularity due to its compelling features like templating, modularization, dependency injection, data binding, component libraries, and more.

For many years, Java has been a popular choice for back-end development. Spring is one of the most popular Java frameworks for developing enterprise-level distributed applications. Full stack developers proficient in technology stack like Angular, Java, and Spring Boot are in high demand.

This book will help learners master Full Stack development skills with Angular as the front end and Spring Boot as the back end. It will guide learners through the entire process of building scalable, enterprise-level, dynamic web applications, from scratch to end-to-end testing.

Chapter 1: Single-page Application Architecture - This chapter provides you an overview of Single Page Applications. Further, it explains the Component Architecture and its importance. It also introduces TypeScript. The last section of the chapter contains clear instructions for setting up the Angular environment and building the first Angular "Hello World" application.

Chapter 2: Angular Building Blocks - It introduces main building blocks of Angular, such as components, directives. This chapter introduces data binding in angular and, template syntax. It covers different types of angular directives, built-in pipes etc.

Chapter 3: Components In-Depth - This chapter takes a deep dive into angular components. It provides in-depth knowledge of component's life cycle hooks. It also covers inter-component communication and data sharing among components. It explains Angular Change Detection Mechanism and runtime optimization.

Chapter 4: Services and Dependency Injection - Components and Services are two different entities in Angular and they serve different purposes. This chapter explains services in depth. It talks about certain application tasks and how those are delegated to services. The chapter also provides knowledge of Dependency Injection -a design pattern. The chapter explains the significance of DI and how it can provide flexibility and modularity to applications. The chapter provides a detailed explanation of the working of Angular Dependency Injection System.

Chapter 5: RxJs Observables - This chapter focuses on synchronous vs. reactive (asynchronous) programming. It explains the Reactive programming paradigm and event handlers. The chapter also explains Observable Design Pattern, terminology and its usage. It then takes a deep dive into creating and working with Observables using RxJs library.

Chapter 6: Routing and Navigation - Routing is the backbone of single page applications. Angular facilitates Single Page Application development by providing built-in router service. This chapter explains how to use Angular Routes to determine a user's navigation from one view to another.

Chapter 7: Forms in Angular - Handling user input has been one of the important functions of front end application development. Angular provides two different strategies for creating forms namely Template Driven Forms and Reactive Forms. This chapter covers both of these approaches in depth, and also summarizes the key differences in two approaches.

Chapter 8: HTTP-client Service - It provides a detailed insight into Http-Client service provided by angular. Communication with back-end services is one of the major tasks to be carried out by front end applications and Angular provides " HttpClient "- API for the same. Chapter explains about carrying out all http related tasks like sending requests to server, requesting typed responses, handling errors,intercepting requests and responses.

Chapter 9: Angular Modules and Standalone Components - This chapter explains the concept of angular modules and its meta-data. Angular introduced the 'StandAlone components' (also pipes, directives) in version 16. Standalone components enhance the application development process reducing the need for NgModules. The chapter delves into the importance of standalone components, and their usage. The chapter also covers bootstrapping the application and lazy loading with standalone components.

Chapter 10: Signals NgRx Introduction and Testing - It introduces Signals, a new feature of angular. Further, the chapter provides insight into state management in angular using NgRx library. The last section of the chapter shows how angular facilitates testing and explains how to test angular components, services etc.

Chapter 11: Enterprise Application Architecture - It provides insight into enterprise level application architecture which is usually distributed in nature. It covers important architectural patterns including multi-tier architecture and MVC architecture for Web application development.

Chapter 12: Spring Core/DI-IOC - Spring framework has become a de-facto standard in java enterprise application development. The chapter provides knowledge on an important feature of spring framework i.e. DI / IOC container. It explains how dependency injection approach helps in development of enterprise applications with greater flexibility, enhanced testability. It explains how DI approach helps in eliminating the shortfalls of previous (non-DI) application development approaches.

Chapter 13: Spring MVC - It teaches web application development using spring web-MVC module. It explains different components involved in spring MVC flow like dispatcher servlet, handler mappings etc.

Chapter 14: Spring Boot - This chapter provides deep knowledge about spring boot project and its significance in building enterprise level distributed application.

Chapter 15: Spring REST - The chapter takes deep dive into REST principles and the significance of this modern approach in achievieing application to application communication and decoupling between client -server applications. The Spring REST module greatly helps in simplified and faster development of RESTful web applications.

Chapter 16: Spring Data JPA - This chapter first explains Object Relational Mapping (ORM) concepts, different ORM tools and significance of Java Persistence API (JPA) for achieving loose coupling between different ORM tools and RDBMS. Later in this chapter, readers learn the spring Data JPA module.This module makes writing the DAO layer of applications extremely simple.

Chapter 17: Testing, Best Practices and Project - This last chapter explains how spring facilitates unit testing of web applications in a simplified way with spring boot. This chapter also delves into best coding practices and java coding conventions. Lastly, the chapter describes an End-To-End Application to be developed using Angular as frontend, and java, spring boot as backend technologies.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/qra2ztr

The code bundle for the book is also hosted on GitHub at
https://github.com/bpbpublications/Full-Stack-Development-with-Angular-and-Spring-Boot.
In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

Chapter 1
Single-page Application Architecture

Introduction

In this chapter of the book, we will cover single-page applications. We will see the characteristics of a single-page application. We will go through its architectural details. Then we will have an introduction to JavaScript, ECMAScript 2015 features and TypeScript. We will also get introduced to Node.js.

Structure

The chapter covers the following topics:

- Single-page applications
- Single-page application architecture
- Introduction to TypeScript
- Introduction to Angular
- Setting up Angular environment in Visual Studio Code
- Building your first Angular app: "Hello World"

Objectives

Upon completion of this chapter, you will be able to understand the single-page application architecture. You will appreciate the significance of SPA architecture in web application development and will know how to leverage the SPA architecture to gain various benefits. At the end of the chapter, you will set up the Angular environment and build your first Angular application.

Single-page applications

Observe *Figure 1.1*. It is a screenshot of an online shopping website:

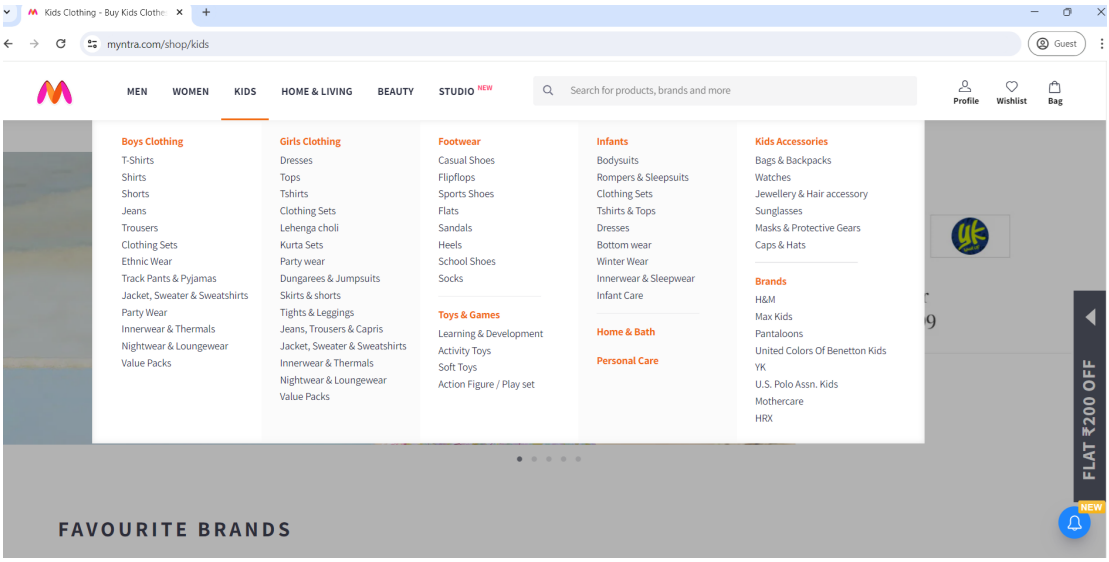

***Figure 1.1**: An online shopping website*

Suppose Bob wants to do some shopping for his daughter, Myra, who is four years old. He visits a website called myntra.com. On visiting the site URL, he gets a lot of options to select from. Various tabs are available, like **MEN, WOMEN, KIDS,** etc. As Myra is a 4-year-old kid, Bob clicks on the **KIDS** tab. We can see that many new options are available now, like **Boys Clothing, Girls Clothing, Footwear, Infants** and so on. As Bob hovers the mouse from one tab to another, respective contents will be changed immediately in the browser.

It is clear from the above example that, depending upon user (Bob) interaction, websites are required to load dynamic content instantly.

Usually, such websites show plenty of menus, menu items, submenus, submenu-items, etc., for the user to select from. Depending upon user interaction with any of those items, respective contents on the page change instantaneously, maybe within fractions of a second.

What if Bob hovers or clicks on the girls' footwear tab, and the website takes over 40 seconds to load the contents?

It means the website is unresponsive for a few seconds, and Bob must wait before seeing the new content. In such a scenario, there are chances that Bob will leave the current website myntra.com and will start looking for other websites to complete the shopping. This response (slow) from websites will result in losing potential clients.

The above example shows that modern websites (web applications) cannot afford to keep their clients waiting, and the webpage must be interactive throughout for the users. The above example discusses an important quality called fluid user experience that modern web applications must possess.

Fluid user experience

Fluid user experience for web applications can be characterized as following:

- Depending on the user interaction, however minute it may be, the web page's contents should be loaded or changed dynamically and quickly.
- A web page should be interactive with the user throughout.

This kind of user experience expected from today's modern web applications, is fluid user experience. It is one of the most essential quality requirements that today's modern websites must support.

Once we know the fluid user experience, we can see how to achieve it.

Web applications need to generate dynamic content/views based on user interaction. Loading another HTML page will trigger a browser refresh and require the creation of a new **Document Object Model** (**DOM**). The process is time-consuming, and hence we cannot afford it. It means dynamic view changes should happen on the same page. We can achieve it through DOM manipulation. It changes the contents dynamically when the user interacts with the page. That means no new page will get loaded in the browser, and there will not be a browser refresh.

Now we are almost there. Let us get all the pieces together.

Single-page application architecture

Single-page application (**SPA**) is a web application that loads the contents dynamically depending on user interaction without refreshing the browser throughout the application life cycle. The user will be navigated through different logical views on the same page.

Let us compare traditional web app architecture with SPA. *Figure 1.2* depicts the comparison clearly:

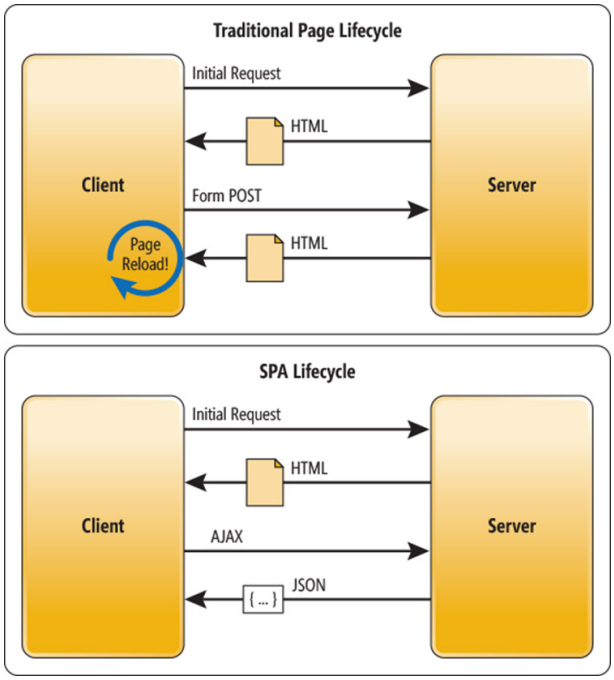

***Figure 1.2**: Traditional vs. SPA architecture*
***Source**: angular.io*

In the traditional approach, during initial and subsequent requests, the browser gets refreshed with new DOM creation because the new html page is sent back as a response from the server every time.

With a modern SPA approach, all the views/mark-ups will be sent to the client during the initial request only, and all dynamic view generation will take place on the client side. Here, dynamic view generation will happen through DOM manipulation, saving us from multiple round trips to the server for subsequent dynamic views. While we are considering client-side dynamic view generation, some questions will be arising like:

- When and for what subsequent requests will be made to the server (as view generation will happen on the client side)?
- What about application data? Where would the application data reside?
- What will happen when data embedded in the view needs to be changed?

Let us go deep to answer these questions.

While all logical views will be sent to the client during the initial request and navigation to different views will happen on the client side, application data will permanently reside on the server side. When data embedded in the view needs to be changed, a request to the server will be made. This request must be made in a non-blocking way, as the page must be interactive with the user throughout.

It will be an **Asynchronous JavaScript and XML** (**AJAX**) call. The request will be made asynchronously (AJAX call) only for data, and the server will respond with data in JSON format.

In the traditional web application development approach, there is one more issue. Supporting rich interactions with multiple components means, those components have many more intermediate states like:

- Menu opened.
- Menu item X selected.
- Menu item Y selected.
- Menu item Z clicked.

For all such intermediate states, server-side rendering is hard to implement. Small view states do not map well to URLs.

Now we can specify the quality requirements to be fulfilled from a modern web application as following:

- Ability to redraw any UI part without requiring a server roundtrip to retrieve HTML.
- No page refresh throughout the app life cycle.
- Page to be interactive throughout.

Considering all the above points, main traits of SPA are as follows:

- It loads all the resources required to navigate the site, on the first-page load.
- As the user clicks links and interacts with the page, subsequent content is loaded dynamically.
- It often updates the URL in the address bar to emulate traditional page navigation, but another full-page request is never made.

Now, we know that SPAs are modern web applications having the important quality of generating dynamic views on the client side. For developing such applications, we use JavaScript, the language that gets executed inside a browser on the client side.

JavaScript

JavaScript was originally developed at Netscape Communications by *Brendan Eich* in 1995. It was first developed as a scripting language for use in the web browser. JavaScript was not regarded as a serious programming language earlier. However, over 20 years after its inception, it is now one of the most used cross-platform languages. Though it started as a small scripting language for adding trivial interactivity to webpages, JavaScript has become a language of choice for front and back-end applications of every size. As it has come a long way and is widely used, the language has undergone many enhancements.

Let us now see what typescript is for.

Introduction to TypeScript

TypeScript is a superset of JavaScript. It offers all the features of JavaScript and a type system. In JavaScript, there are a few primitive types available, like a number, string, Boolean, etc., and it is a dynamically typed language. That means we do not have to specify the type of a variable while declaring it. Datatypes are automatically converted as needed during the execution of JavaScript. For example, we can declare a variable as follows:

```
let someVal = 13;     (here type of someVal is number)
And we can assign a different type to the variable someVal later as follows:
someVal = “Assigning a string value here”; (here type of someVal is string)
```

These kinds of assignments are inconsistent and may not be intended. JavaScript is dynamically typed, and these assignments do not generate any error messages. However, TypeScript will generate an error message here. The main advantage of TypeScript here is that it can highlight unexpected behavior in your code, lowering the chances of bugs.

Typescript can be regarded as an enhanced version of JavaScript with many new features.

Visual Studio Code

For developing Angular applications, we will need some **integrated development environment** (**IDE**). We will use *Visual Studio Code* which is a free editor and IDE. VS Code is a powerful code editor that comes with built in support for JavaScript, TypeScript, and Node.js.

Installing Visual Studio Code

Download Visual Studio Code from the link:

https://code.visualstudio.com/download

Run the downloaded executable and follow the steps to install VS Code on your machine. After installing VS Code editor, we will go ahead with installations of Node.js, type script and Angular CLI.

Node.js

We know JavaScript gets executed inside a browser, that is, on the client side. **Node.js** is an open-source and cross-platform runtime environment for JavaScript on the Server side. Node.js is built on top of the Google Chrome V8 JavaScript engine, mainly used to create web servers. It runs the V8 JavaScript engine, the core of Google Chrome, outside the browser. This allows Node.js to be very performant.

Installation of Node.js

Node.js installation steps are as follows: