Introduction to ASP.NET Core Development using React with Redux Eğitimi

Eğitim Hakkında

Introduction to ASP.NET Core Development using React with Redux training;

teaches attendees the skills necessary to build a modern web application powered by JavaScript on the client-side and ASP.NET Core on the backend. Students learn business application development skills, including securing an application with a login, building multi-page applications with routing, and building complex forms including validation. Participants also learn best practices of React, Redux, and ASP.NET Core and their usage in a Single Page Application (SPA).

Neler Öğreneceksiniz

- Understand the goals and benefits of the .NET Core platform
- Make good decisions about application architecture and the choice of data access technology
- Use ASP.NET Core's routing system to achieve a REST-style architecture
- Gain experience building a service that makes data available via a modern Web API
- Use a JavaScript package manager
- Understand the new JavaScript language features including classes, modules, and arrow functions
- Articulate what React is and why it is useful
- Explore the basic architecture of a React application
- Gain a deep understanding of JSX and the Virtual DOM
- Use React components to build interactive interfaces
- Create and validate forms using controlled components
- Make HTTP calls to read or change data
- Configure simple and complex routing
- Utilize Redux to manage the state of the application
- Use React and Redux together
- Implement React and Redux best practices
- Write unit tests for React using Jest and Enzyme

Eğitim İçeriği

Introduction

.NET Core SDK

- Installation
- Version Management
- Command-Line Interface (CLI)
- Hello World Application
- Using Visual Studio Code for C# Coding
- Optional: Using Visual Studio 2019 for C# Coding
- Optional: Using Visual Studio for Mac for C# Coding

ASP.NET Core Application Architecture

- NuGet Packages and Metapackages
- Application Startup
- Hosting Environments
- Middleware and the Request Processing Pipeline
- Services and Dependency Injection

Application Configuration

- Configure and ConfigureServices
- Configuration Providers and Sources
- Configuration API
- Options Pattern
- HTTPS, GDPR, and HTTP/2

Request Routing

- RESTful Services
- Endpoint Routing
- Attribute-Based Routing
- Route Templates
- Route Constraints

Models

- Persistence Ignorance
- Object-Relational Mapping
- Entity Framework (EF) Core
- Automapper

Controllers

- Responsibilities
- Requirements and Conventions
- Dependencies
- Action Results

Web APIs

- Introduction
- CRUD Operations
- Bad Requests
- Cross-Origin Resource Sharing (CORS)

Application State

- Client-Side vs. Server-Side
- HttpContext.Items
- Session State

Error Handling

- Best Practices
- HTTP Error Status Codes
- Status Code Pages
- Developer Exception Page
- Exception Filters

Logging

- Configuration
- ILogger
- Serilog and Seq

Testing

- Unit Testing
- xUnit
- Testing Controllers
- Integration Testing

Introduction to React and Redux

- What is React?
- What problem does React solve?
- Development Ecosystem
- React versus other frameworks

Development Tools

- Create React App project generator
- React Developer Tools
- Running and Debugging a React Application
- Role of Node.js
- Purpose of React and ReactDOM

Functional Components

- What are Components?
- Create Element and JSX
- Benefits of JSX
- Fragments
- JavaScript Arrow Functions
- ES2015 Modules
- JSX and Expressions
- Displaying Collections of Data
- JavaScript Array Maps and React Keys
- Passing Data with Props
- Validating Props with PropTypes
- Default Props
- Using Memo

Class-Based Components

- JavaScript Classes and Extends
- Configuring State
- Lifecycle Methods
- Google Performance Tool
- Context of Event Handlers
- Class Properties and Class Arrow Functions
- PropTypes and Default Props on Classes

Hooks

- Overview of Hooks
- State Hook
- Effect Hook
- Ref Hook
- Callback Hook

Advanced Components

- Composition vs. Inheritance
- Patterns: Specialization, Containment, and Higher Order Components
- Lifting State Up
- Forwarding Refs
- Context

Redux

- Managing Application State
- Three Principles of Redux
- Pure Functions
- Reducer Functions
- Composing Reducer Functions
- Dispatching Actions
- Action Creators

Connect React to Redux

- Connect React to Redux with React-Redux
- React-Redux Higher Order Components

- React-Redux Hooks
- Using State Selectors
- Optimizing State Selectors

Connect React to ASP.NET Core REST API

- JavaScript Review: Callbacks, Promises & Async/Await
- Using the Fetch API with ASP.NET Core REST API
- Asynchronous Operations and React/Redux using Saga
- JavaScript Generators
- Sagas Helpers
- Declarative Effects
- Error Handling
- Connecting to Redux

Unit Testing

- Using Jest
- Organizing Tests and Test Suites
- Setup and Teardown of Tests
- Performing Assertions with Expect
- Using Spies
- Snapshot Testing
- DOM Testing
- Shallow Testing
- Generating Code Coverage Reports
- React Unit Testing
- Redux Unit Testing

React Router

- What is routing?
- URL as State
- React Router Hooks
- Configuring Routes
- Page Pattern
- Error Page
- Redirects
- Animated Transitions
- Nested Routes

Advanced Forms

- What is Formik?
- Challenges with React and Forms
- Formik Higher Order Components
- Formik Hooks
- Form-Level Validation
- Field-Level Validation
- Synchronous and Asynchronous Validation
- Form Submission
- Form Submission Phases

Authentication

- Introduction
- ASP.NET Core Identity
- Cookie Middleware
- Authorization
- Claims-Based Authorization
- React Login Form
- Login Error Handling
- Integrating Authorization with Routing