Students Handout

Introduction to Web API Workshop 2

Welcome to the "Building Web APIs with Repository Pattern and AutoMapper" workshop! In this workshop, we'll explore the implementation of a Web API for a job portal using the Repository Pattern and AutoMapper in C#.

Project Overview

We have a project structured into two controllers: UserController and JobSeekerController. These controllers handle user-related operations (registration and login) and job seeker-related operations (viewing job lists, applying for jobs, and retrieving applied jobs).

UserController

**Dependencies**

IUserService: Interface defining user-related operations.

IMapper: Interface for object-object mapping.

Endpoints

POST /account/register

Registers a new user.

Request Body: UserDto

Returns: User information after registration.

POST /account/login

Logs in a user.

Request Body: LoginDto

Returns: User information after successful login.

JobSeekerController

**Dependencies**

IJobService: Interface defining job-related operations.

IUserService: Interface defining user-related operations.

IUserRepository: Interface defining user repository operations.

IApplicationService: Interface defining job application-related operations.

IMapper: Interface for object-object mapping.

Endpoints

GET /job/GetJobList

Retrieves the list of all available jobs.

POST /JobSeeker/ApplyJob

Applies for a job.

Request Body: ApplicationDto

Returns: Success message upon successful application.

GET /JobSeeker/AllApplyJobs

Retrieves the list of all jobs applied for by a user.

Implementation Details

UserController

Utilizes IUserService and IMapper for user-related operations and object mapping.

Provides endpoints for user registration and login.

JobSeekerController

Depends on IJobService, IUserService, IUserRepository, and IApplicationService.

Implements endpoints for retrieving job lists, applying for jobs, and retrieving applied jobs.

Tips and Best Practices

Dependency Injection: Notice how dependencies are injected into controllers' constructors, promoting code modularity and testability.

Exception Handling: Check how exceptions are handled, and appropriate responses are provided.

Happy coding!