Onboarding Document for New Developers

Table of Contents

Project Overview

The purpose of this project is to provide a robust solution for managing investments and personal data. The scope includes functionalities for creating, updating, and retrieving investment and personal information. The benefits of this project include improved data management, enhanced user experience, and streamlined operations for investment tracking.

Development Environment Requirements

To set up your local development environment, ensure you have the following installed:

- **Java**: Version 11 or higher
- **Maven**: Version 3.6 or higher
- **Docker**: For containerization (optional, but recommended)
- **Postman**: For API testing
- **IDE**: IntelliJ IDEA or Eclipse (recommended)

Step-by-Step Instructions to Clone, Build, and Run the Project Locally

1. **Clone the Repository**:

   ```
   git clone https://github.com/your-repo/project-name.git
   cd project-name
   ```

2. **Build the Project**:

   ```
   mvn clean install
   ```

3. **Run the Project**:

   ```
   mvn spring-boot:run
   ```

4. **Access the Application**: Open your browser and navigate to `http://localhost:8080`.

Project Structure Description

Controllers

Controllers handle incoming requests and return responses. They are responsible for routing requests to the appropriate services.

- **Examples**:
  - `InvestmentController.java`: Manages investment-related endpoints.
  - `PersonController.java`: Manages personal information endpoints.

Services

Services contain the business logic of the application. They interact with repositories to perform CRUD operations.

- **Examples**:
  - `InvestmentService.java` : Contains logic for managing investments.
  - `PersonService.java` : Contains logic for managing personal data.

Repositories

Repositories provide access to data storage. They are responsible for data retrieval and persistence.

- **Examples**:
  - `InvestmentRepository.java` : Interface for investment data operations.
  - `PersonRepository.java` : Interface for personal data operations.

Entities

Entities represent the data model of the application. They are mapped to database tables.

- **Examples**:
  - `Investment.java` : Represents an investment entity.
  - `Person.java` : Represents a person entity.

DTOs

Data Transfer Objects (DTOs) are used to transfer data between layers. They help in reducing the amount of data sent over the network.

- **Examples**:

- `InvestmentDTO.java`
  : DTO for investment data.
- `PersonDTO.java`
  : DTO for personal data.

Mappers

Mappers are responsible for converting between entities and DTOs. They simplify data transformation.

- **Examples**:
  - `InvestmentMapper.java`
    : Maps between `Investment` and `InvestmentDTO`.
  - `PersonMapper.java`
    : Maps between `Person` and `PersonDTO`.

Configuration

Configuration files manage application settings, including security, CORS, and API documentation.

- **Key Configurations**:
  - Security settings
  - CORS policies
  - Swagger configuration for API documentation

Code Conventions, Style, and Git Strategy

- **Code Style**: Follow Java naming conventions and use consistent indentation (4 spaces).
- **Git Strategy**:

    - `main` : Stable production-ready code.
    - `develop` : Latest development changes.
    - `feature/*` : New features being developed.
    - `bugfix/*` : Fixes for bugs in development.

Running Unit and Integration Tests

To run tests, use the following command:

```
mvn test
```

- **Testing Frameworks**: JUnit for unit tests, Mockito for mocking dependencies.

API Documentation Access

API documentation is available via Swagger. Access it at:

```
http://localhost:8080/swagger-ui.html
```

- **Main Endpoints**:
    - `/api/investments` : Manage investments.
    - `/api/persons` : Manage personal information.

Glossary of Common Technical and Business Terms

- **DTO**: Data Transfer Object
- **CRUD**: Create, Read, Update, Delete
- **CORS**: Cross-Origin Resource Sharing
- **API**: Application Programming Interface

Communication Channels and Useful Contacts

- **Slack**: Join the project channel for real-time communication.
- **Email**: Reach out to the technical lead at techlead@example.com.
- **QA Team**: Contact QA at qa@example.com for testing-related queries.

This onboarding document aims to provide you with the necessary tools and knowledge to get started quickly and effectively. Welcome to the team!