🎓 Complete Beginner's Guide to FlowTrack (Spring Boot)

☑ What I Fixed

I've cleaned up your repository by removing:

- ❌ `.idea/` folder (IntelliJ IDE config)
- ❌ `.vscode/` folder (VS Code config)
- ❌ `target/` folder (compiled `.class` files and build output)

These files are now **ignored by git** (they still exist locally but won't be committed).

📚 Understanding Spring Boot Project Structure

What is Spring Boot?

Spring Boot is a Java framework that makes it easy to create web applications. Think of it as a pre-configured toolkit for building websites and APIs.

Your Project Structure Explained:

```
FlowTrack/
├── src/
│   ├── main/
│   │   ├── java/                    ← Your Java code lives here
│   │   │   └── com/example/...      ← Package structure
│   │   │       ├── controller/      ← Web pages/API endpoints
│   │   │       ├── model/           ← Database tables (entities)
│   │   │       ├── repository/      ← Database access layer
│   │   │       ├── service/         ← Business logic
│   │   │       ├── dto/             ← Data transfer objects
│   │   │       └── config/          ← App configuration
│   │   └── resources/
│   │       ├── application.properties  ← App settings (DB, port, etc.)
│   │       ├── static/              ← CSS, JS, images
│   │       └── templates/           ← HTML pages (Thymeleaf)
│   └── test/                        ← Test files
├── pom.xml                          ← Dependencies & build config
├── .gitignore                       ← Files to ignore in git
└── target/                          ← Generated files (gitignored)
```

🏗 Spring Boot Architecture Layers

1. Model Layer (`model/`)

- **What:** Defines database tables
- **Example:** `Item.java`, `User.java`, `Vendor.java`
- **Think of it as:** Blueprint for data

```
@Entity  // This is a database table
public class Item {
    @Id  // Primary key
    private Long id;
    private String name;
    private int quantity;
}
```

2. Repository Layer (`repository/`)

- **What:** Talks to the database
- **Example:** `IItemRepository.java`
- **Think of it as:** Database messenger

```
public interface IItemRepository extends JpaRepository<Item, Long> {
    // Spring automatically creates: save(), findAll(), findById(), etc.
}
```

3. Service Layer (`service/`)

- **What:** Business logic (calculations, rules)
- **Example:** `ItemService.java`
- **Think of it as:** Brain of the app

```
@Service
public class ItemService {
    public void addItem(Item item) {
        // Validation logic
        // Save to database
    }
}
```

4. Controller Layer (`controller/`)

- **What:** Handles web requests (URLs)
- **Example:** `ItemController.java`
- **Think of it as:** Receptionist directing traffic

```
@Controller
public class ItemController {
```

```
    @GetMapping("/items")  // URL: localhost:8080/items
    public String viewItems(Model model) {
        return "item_list";  // Returns item_list.html
    }
}
```

🚀 How to Run This Project

Prerequisites:

1. **Java 11** installed
2. **Maven** installed (or use `mvnw` wrapper)
3. **MySQL** database running

Step 1: Set Up Database

```
# Run this script to create database
.\setup-mysql-database.bat
```

Step 2: Configure Database Connection

Edit `src/main/resources/application.properties`:

```
spring.datasource.url=jdbc:mysql://localhost:3306/flowtrack
spring.datasource.username=root
spring.datasource.password=yourpassword
```

Step 3: Build the Project

```
# Clean and compile
mvn clean install

# Or use the wrapper (no Maven installation needed)
.\mvnw clean install
```

Step 4: Run the Application

```
# Option 1: Using Maven
mvn spring-boot:run

# Option 2: Using the batch file
.\start.bat
```

```
# Option 3: Run the JAR
java -jar target/IMS-AP-0.0.1-SNAPSHOT.jar
```

Step 5: Access the Application

Open browser: `http://localhost:8080`

🔧 Common Development Tasks

1. Adding a New Feature (e.g., "Category")

Step A: Create Model

`src/main/java/com/example/IMS/model/Category.java`

```
@Entity
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    // Getters and setters
}
```

Step B: Create Repository

`src/main/java/com/example/IMS/repository/ICategoryRepository.java`

```
public interface ICategoryRepository extends JpaRepository<Category, Long> {
}
```

Step C: Create Service

`src/main/java/com/example/IMS/service/CategoryService.java`

```
@Service
public class CategoryService {
    @Autowired
    private ICategoryRepository categoryRepository;

    public List<Category> getAllCategories() {
        return categoryRepository.findAll();
    }
}
```

Step D: Create Controller

`src/main/java/com/example/IMS/controller/CategoryController.java`

```
@Controller
@RequestMapping("/categories")
public class CategoryController {
    @Autowired
    private CategoryService categoryService;

    @GetMapping
    public String listCategories(Model model) {
        model.addAttribute("categories", categoryService.getAllCategories());
        return "category_list";
    }
}
```

Step E: Create HTML Template

`src/main/resources/templates/category_list.html`

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head><title>Categories</title></head>
<body>
    <table>
        <tr th:each="category : ${categories}">
            <td th:text="${category.name}"></td>
        </tr>
    </table>
</body>
</html>
```

🛠 IDE Setup

For IntelliJ IDEA:

1. Open IntelliJ → `File` → `Open` → Select `FlowTrack` folder
2. Wait for Maven to download dependencies
3. Right-click `ImsApApplication.java` → `Run`

For VS Code:

1. Install extensions:
   - Extension Pack for Java
   - Spring Boot Extension Pack
2. Open folder in VS Code

3. Press `F5` to run

Understanding `pom.xml`

This file manages your project's dependencies (libraries):

```
<dependencies>
    <!-- Web framework -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!-- Database access -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <!-- MySQL driver -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>
</dependencies>
```

To add a new library:

1. Find it on mvnrepository.com
2. Copy the XML snippet
3. Paste into `<dependencies>` section
4. Run `mvn clean install`

Common Issues & Solutions

Issue: Port 8080 already in use

Solution: Change port in `application.properties`:

```
server.port=8081
```

Issue: Database connection failed

Solution: Check MySQL is running:

```

```

```
# Check MySQL status
mysql -u root -p
```

Issue: Changes not reflected

Solution:

```
# Clean build and restart
mvn clean install
mvn spring-boot:run
```

Issue: Build fails

Solution:

```
# Delete target folder and rebuild
Remove-Item -Recurse -Force target
mvn clean install
```

📖 Key Spring Boot Annotations

Annotation	Purpose	Example
`@SpringBootApplication`	Main app entry point	`ImsApApplication.java`
`@Entity`	Database table	`Item.java`
`@Repository`	Database access	`IItemRepository.java`
`@Service`	Business logic	`ItemService.java`
`@Controller`	Web controller	`ItemController.java`
`@RestController`	REST API controller	For JSON responses
`@Autowired`	Inject dependency	Connects layers
`@GetMapping`	HTTP GET request	View pages
`@PostMapping`	HTTP POST request	Form submissions

🎯 Next Steps for Learning

1. **Read the docs:**
   - Spring Boot Official Docs
   - Thymeleaf Template Engine

2. **Explore existing code:**
    - Start with `ItemController.java`
    - Trace how data flows: Controller → Service → Repository → Database
3. **Make small changes:**
    - Add a new field to `Item` model
    - Create a simple "Hello World" controller
    - Modify an HTML template
4. **Watch tutorials:**
    - Search YouTube: "Spring Boot Beginner Tutorial"
    - Follow along with your project

Pro Tips

- ☑ **Always rebuild after code changes:** `mvn clean install`
- ☑ **Check logs when errors occur:** Look in console output
- ☑ **Use Spring DevTools:** Auto-restart on file changes
- ☑ **Test in browser:** Use Chrome DevTools to debug
- ☑ **Ask for help:** Stack Overflow, Spring community

Files You Should NEVER Commit

Your `.gitignore` is now properly configured to exclude:

- `target/` - Build output
- `.idea/` - IntelliJ config
- `.vscode/` - VS Code config
- `*.class` - Compiled files
- `*.log` - Log files
- Database files

Always commit:

- `src/` - Source code
- `pom.xml` - Dependencies
- `application.properties` - Config (except passwords!)
- `.gitignore` - Ignore rules
- `README.md` - Documentation

Need Help?

If you get stuck:

1. Read error messages carefully

2. Check Stack Overflow
3. Use Spring Boot's excellent error pages (they explain issues)
4. Ask me specific questions!

Happy coding! 🚀