**Topics in Information Systems and Programming Languages - Mini Project – Yaniv Krol & Ron Rachev**

**System design**

* Programming Language chosen – Java Spring

**The Spring Framework (Spring)** is an open-source application framework that provides infrastructure support for developing Java applications.

Java programs are complex and feature many heavyweight components. Heavyweight means the components are dependent on the underlying operating system (OS) for their appearance and properties.

Spring is considered to be a secure, low-cost and flexible framework. Spring improves coding efficiency and reduces overall application development time because it is lightweight -- efficient at utilizing system resources -- and has a lot of support.

**Installation -**

1. Install java
2. Git clone <https://github.com/eNtrozx/todosys>
3. Run CMD
4. Execute 'C:\Users\{Computer Name}\.vscode\extensions\redhat.java-1.4.0\jre\17.0.2-win32-x86\_64\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '@C:\Users\{Computer name}\AppData\Local\Temp\cp\_ahaiqmd5yy0jz32s4vikbvp41.argfile' 'bgu.informationsystems.todosys.TodosysApplication'

**Language highlights**

**Pros -**

1. Spring is a lightweight framework because of its POJO implementation (POJO stands for Plain Old Java Object. It is an ordinary Java object, not bound by any special restriction other than those forced by the Java Language Specification and not requiring any classpath) . It does not force the programmer to inherit any class and implement any interface.
2. Spring’s core logic is **dependency injection.** Dependency injection is a programming pattern that allows developers to build more decoupled architectures. Dependency injection means that Spring understands the different Java annotations that a developer puts on top of classes. Spring knows that the developer wants to create an instance of a class and that Spring should manage it. Spring also understands the dependency and makes sure that all instances created have properly populated dependencies.
3. It provides flexible libraries trusted by developers all over the world. The developer can choose either XML or Java-based annotations for configuration options. The IoC and DI features provide the foundation for a wide-ranging set of features and functionality. It makes the job simpler.
4. Powerful Abstraction: It provides a powerful abstraction to JEE specifications such as JMS, JDBC, JPA ( This is what we actually used to handle the database) , and JTA.
5. Flexible Configurations – Developers have the option to choose either XML or Java-based annotations for configuration purposes. Having such an option makes the jobs of developers a lot simpler.

**Cons –**

1. **Complexity:** Working with Spring is more complex. It requires a lot of expertise. If you have not used Spring before, first you will have to learn. The learning curve is also difficult, so if you have not a lot of development experience, it is difficult to learn.

If you have not development experience in the field, it would be quite difficult to learn, It is difficult due to new programming methods.

1. One of the biggest advantages of Spring is that it gives developers **a wide array of options**, but this could also be a disadvantage because it causes confusion. Developers have to know which features will be useful, and making the wrong decisions could lead to significant delays.

**Impressions and notes from the working process**

1. **Layers –**

We have utilized the Spring Boot layers in our coding.

* Created a data access layer (Classes) that performs CRUD operation.
* The client makes the HTTP requests (PUT / GET / PATCH / DELETE).
* The request goes to the controller, and the controller maps that request and handles it. After that, it calls the service logic if required.
* In the service layer, all the business logic performs. It performs the logic on the data that is mapped to JPA with model classes.
* A response is returned to the use / error if there is an issue.



1. **DB abstraction –** We have used Java Persistence API (JPA) which is a specification provided by Java for APIs accessing various SQL databases. It allowed us to map entities from JSON to the data store and back (Via java classes/ objects)

**Pros -**

* Separates SQL from other logic. This lets you work with a higher level of abstraction as you can navigate through related classes, “reduces” the amount of SQL you should write, simplifying those operations with some generic methods, and JPA makes the translation of the objects you send into SQL.
* High level of abstraction that is useful for people who need to maintain your code after the fact. Changes to business logic would be easier for someone else to take on.
* Integrates with Java Bean Validation which allows you to have richer validation logic than what could be provided by SQL databases.
* Access is portable across databases by default. Useful when you have enterprise clients which can either be SQL Server, MySQL, DB2 or Oracle. So you can support more clients.

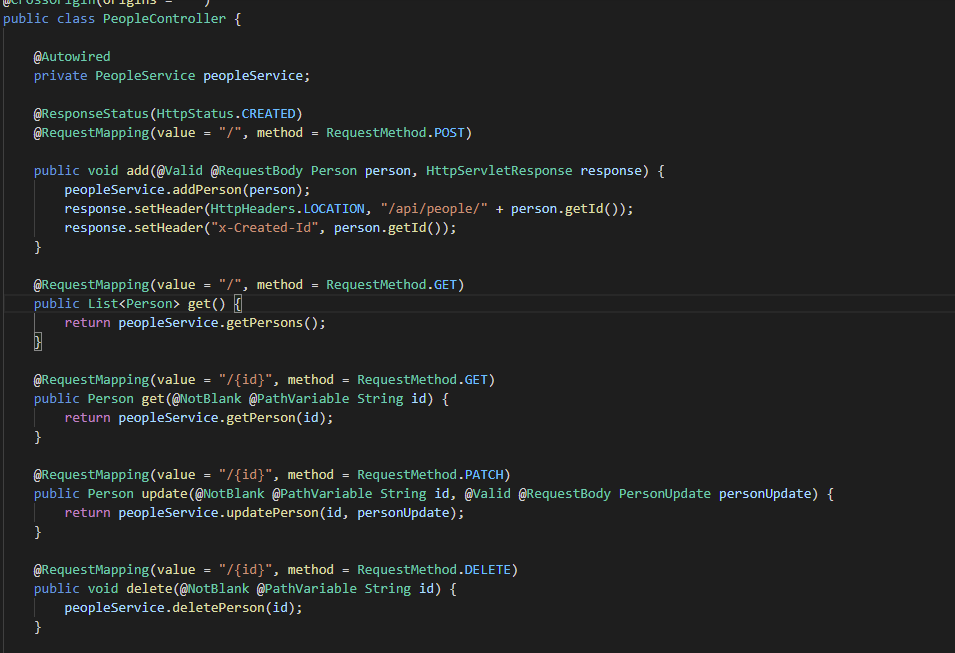
1. Does our system protects the data it stores from being corrupted?

Jpa has the functionality to backup data via a simple command

1. **Deserialization via Spring controller –**

In Spring MVC, a controller can handle the requests for all HTTP methods, which is a backbone of RESTful web services. For example, you can handle a GET method to perform read operations, POST methods to create resources, PUT methods to update resources, and DELETE methods to remove resources from the server.

In the case of REST, the representation of data is very important and that's why Spring MVC allows you to use the @ResponseBody annotation and various HttpMessgeConverter implementations. By using this, you can directly send a response to a client, e.g. the resource clients want and also in the format they want. See here to learn more about HttpMessageConvert and @ResponseBody annotation. Example :

****

1. Performing summation queries on the data store

**TODO**