FPT Education
FPT UNIVERSITY

Java

Spring Data

What is Spring Data?

- **Spring Data** is an umbrella project, not a single component. It's a collection of modules, each focusing on a different aspect of data access or a specific type of data store.
- **Features**
  - Powerful repository and custom object-mapping abstractions
  - Dynamic query derivation from repository method names
  - Implementation domain base classes providing basic properties
  - Support for transparent auditing (created, last changed)
  - Possibility to integrate custom repository code
  - Easy Spring integration via JavaConfig and custom XML namespaces
  - Advanced integration with Spring MVC controllers
  - Experimental support for cross-store persistence

Main modules

- Spring Data Commons - Core Spring concepts underpinning every Spring Data module.
- Spring Data JDBC - Spring Data repository support for JDBC.
- Spring Data R2DBC - Spring Data repository support for R2DBC.
- **Spring Data JPA** - Spring Data repository support for JPA.
- Spring Data KeyValue - Map based repositories and SPIs to easily build a Spring Data module for key-value stores.
- Spring Data LDAP - Spring Data repository support for Spring LDAP.
- Spring Data MongoDB - Spring based, object-document support and repositories for MongoDB.
- Spring Data Redis - Easy configuration and access to Redis from Spring applications.
- Spring Data REST - Exports Spring Data repositories as hypermedia-driven RESTful resources.
- Spring Data for Apache Cassandra - Easy configuration and access to Apache Cassandra or large scale, highly available, data oriented Spring applications.
- Spring Data for Apache Geode - Easy configuration and access to Apache Geode for highly consistent, low latency, data oriented Spring applications.

FPT Education
FPT UNIVERSITY

Java

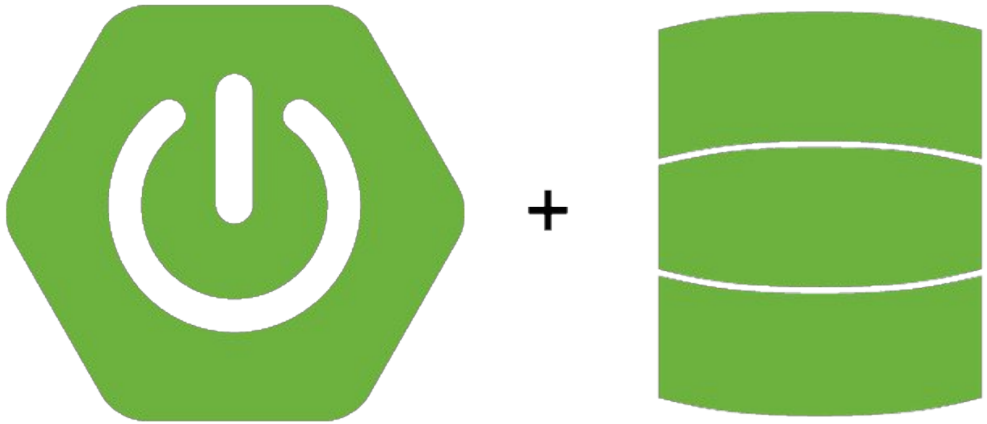
Spring Data JPA

Objectives

- What is Spring Data?
- What is Spring Data JPA?
- Core Concepts
- Repository Interface
- Query Methods
- JPQL and Native Queries
- Spring Data JPA with Spring Boot
- Benefits and Best Practices

What is Spring Data JPA ?

- **Spring Data JPA** is a part of the Spring Framework that simplifies the development of data access layers for Java applications, specifically those using JPA (Java Persistence API). It provides a higher-level abstraction over JPA, allowing developers to interact with databases using simpler and more expressive methods.



Key Features of Spring Data JPA

- **Repository Abstraction:** It introduces the Repository interface, which defines common data access methods like save, find, delete, etc. Spring Data JPA automatically generates implementations for these methods based on method names and domain class conventions.
- **Spring Data JPA Repositories:** You create custom repositories by extending the Repository interface and defining custom query methods. Spring Data JPA uses a naming convention to automatically generate query implementations.
- **Query Annotation:** You can use the @Query annotation to write custom JPQL or native SQL queries.
- **Specification API:** You can create dynamic query criteria using the Specification API, allowing for complex filtering and sorting.
- **Integration with Spring Framework:** It seamlessly integrates with other Spring components like Spring MVC, Spring Security, and Spring Boot.

Benefits of using Spring Data JPA

- **Reduced Boilerplate Code:** It significantly reduces the amount of code you need to write for data access operations.
- **Improved Productivity:** By simplifying data access, it increases developer productivity.
- **Enhanced Maintainability:** The code becomes more maintainable due to its simplicity and adherence to conventions.
- **Flexibility:** It supports various database systems through JPA and provides options for custom query methods and specifications.
- **Strong Community and Support:** It benefits from the large Spring community and has extensive documentation and support resources.

Core Concepts of Spring Data JPA

Spring Data JPA Annotations

- **@Entity:** Marks a class as a JPA entity, indicating that it represents a database table.
- **@Table:** Specifies the name of the database table to which the entity is mapped.
- **@Id:** Marks a field as the primary key of the entity.
- **@GeneratedValue:** Specifies the strategy for generating primary key values (e.g., GenerationType.IDENTITY, GenerationType.AUTO, GenerationType.SEQUENCE, GenerationType.TABLE).
- **@Column:** Configures column-specific properties like name, length, nullable, and unique constraints.

Relationship Mapping Annotations

- **@OneToOne:** Defines a one-to-one relationship between two entities.
- **@OneToMany:** Defines a one-to-many relationship between two entities.
- **@ManyToOne:** Defines a many-to-one relationship between two entities.
- **@ManyToMany:** Defines a many-to-many relationship between two entities.
- **@JoinColumn:** Specifies the foreign key column in the database table

Query Annotations

- **@Query:** Defines a custom JPQL or native SQL query.
- **@Modifying:** Indicates that a query modifies data.
- **@Param:** Binds a parameter to a query.
- **Other Annotations**
- **@Temporal:** Specifies the temporal type of a field.
  (e.g., TemporalType.DATE, TemporalType.TIME, TemporalType.TIMESTAMP)
- **@Enumerated:** Specifies the enumeration type for a field.
- **@Lob:** Maps large object types (e.g., CLOB, BLOB).
- **@Transient:** Excludes a field from persistence.
- **@Version:** Specifies a version field for optimistic locking.

JPA Query Methods

JPA Query

Keyword	Sample	JPQL snippet
`Distinct`	`findDistinctByLastnameAndFirstname`	`select distinct … where x.lastname = ?1 and x.firstname = ?2`
`And`	`findByLastnameAndFirstname`	`… where x.lastname = ?1 and x.firstname = ?2`
`Or`	`findByLastnameOrFirstname`	`… where x.lastname = ?1 or x.firstname = ?2`
`Is`, `Equals`	`findByFirstname`, `findByFirstnameIs`, `findByFirstnameEquals`	`… where x.firstname = ?1` (or `… where x.firstname IS NULL` if the argument is `null`)
`Between`	`findByStartDateBetween`	`… where x.startDate between ?1 and ?2`
`LessThan`	`findByAgeLessThan`	`… where x.age < ?1`
`LessThanEqual`	`findByAgeLessThanEqual`	`… where x.age <= ?1`
`GreaterThan`	`findByAgeGreaterThan`	`… where x.age > ?1`
`GreaterThanEqual`	`findByAgeGreaterThanEqual`	`… where x.age >= ?1`
`After`	`findByStartDateAfter`	`… where x.startDate > ?1`
`Before`	`findByStartDateBefore`	`… where x.startDate < ?1`
`IsNull`, `Null`	`findByAge(Is)Null`	`… where x.age is null`
`IsNotNull`, `NotNull`	`findByAge(Is)NotNull`	`… where x.age is not null`

https://docs.spring.io/spring-data/jpa/reference/jpa/query-methods.html

JPA Query

`Like`	`findByFirstnameLike`	`… where x.firstname like ?1`
`NotLike`	`findByFirstnameNotLike`	`… where x.firstname not like ?1`
`StartingWith`	`findByFirstnameStartingWith`	`… where x.firstname like ?1` (parameter bound with appended `%`)
`EndingWith`	`findByFirstnameEndingWith`	`… where x.firstname like ?1` (parameter bound with prepended `%`)
`Containing`	`findByFirstnameContaining`	`… where x.firstname like ?1` (parameter bound wrapped in `%`)
`OrderBy`	`findByAgeOrderByLastnameDesc`	`… where x.age = ?1 order by x.lastname desc`
`Not`	`findByLastnameNot`	`… where x.lastname <> ?1`
`In`	`findByAgeIn(Collection<Age> ages)`	`… where x.age in ?1`
`NotIn`	`findByAgeNotIn(Collection<Age> ages)`	`… where x.age not in ?1`
`True`	`findByActiveTrue()`	`… where x.active = true`
`False`	`findByActiveFalse()`	`… where x.active = false`
`IgnoreCase`	`findByFirstnameIgnoreCase`	`… where UPPER(x.firstname) = UPPER(?1)`

https://docs.spring.io/spring-data/jpa/reference/jpa/query-methods.html

Repository query keywords

Keyword	Description
`find…By`, `read…By`, `get…By`, `query…By`, `search…By`, `stream…By`	General query method returning typically the repository type, a `Collection` or `Streamable` subtype or a result wrapper such as `Page`, `GeoResults` or any other store-specific result wrapper. Can be used as `findBy…`, `findMyDomainTypeBy…` or in combination with additional keywords.
`exists…By`	Exists projection, returning typically a `boolean` result.
`count…By`	Count projection returning a numeric result.
`delete…By`, `remove…By`	Delete query method returning either no result (`void`) or the delete count.
`…First<number>…`, `…Top<number>…`	Limit the query results to the first `<number>` of results. This keyword can occur in any place of the subject between `find` (and the other keywords) and `by`.
`…Distinct…`	Use a distinct query to return only unique results. Consult the store-specific documentation whether that feature is supported. This keyword can occur in any place of the subject between `find` (and the other keywords) and `by`.

https://docs.spring.io/spring-data/jpa/reference/repositories/query-keywords-reference.html

Reserved methods


```
deleteAllById(Iterable<ID> identifiers)
```

```
deleteById(ID identifier)
```

```
existsById(ID identifier)
```

```
findAllById(Iterable<ID> identifiers)
```

```
findById(ID identifier)
```

https://docs.spring.io/spring-data/jpa/reference/repositories/query-keywords-reference.html

Supported query method predicate keywords and modifiers

Logical keyword	Keyword expressions
`AND`	`And`
`OR`	`Or`
`AFTER`	`After`, `IsAfter`
`BEFORE`	`Before`, `IsBefore`
`CONTAINING`	`Containing`, `IsContaining`, `Contains`
`BETWEEN`	`Between`, `IsBetween`
`ENDING_WITH`	`EndingWith`, `IsEndingWith`, `EndsWith`
`EXISTS`	`Exists`
`FALSE`	`False`, `IsFalse`
`GREATER_THAN`	`GreaterThan`, `IsGreaterThan`
`GREATER_THAN_EQUALS`	`GreaterThanEqual`, `IsGreaterThanEqual`
`IN`	`In`, `IsIn`

https://docs.spring.io/spring-data/jpa/reference/repositories/query-keywords-reference.html

Supported query method predicate keywords and modifiers

`IS`	`Is`, `Equals`, (or no keyword)
`IS_EMPTY`	`IsEmpty`, `Empty`
`IS_NOT_EMPTY`	`IsNotEmpty`, `NotEmpty`
`IS_NOT_NULL`	`NotNull`, `IsNotNull`
`IS_NULL`	`Null`, `IsNull`
`LESS_THAN`	`LessThan`, `IsLessThan`
`LESS_THAN_EQUAL`	`LessThanEqual`, `IsLessThanEqual`
`LIKE`	`Like`, `IsLike`
`NEAR`	`Near`, `IsNear`
`NOT`	`Not`, `IsNot`
`NOT_IN`	`NotIn`, `IsNotIn`
`NOT_LIKE`	`NotLike`, `IsNotLike`
`REGEX`	`Regex`, `MatchesRegex`, `Matches`
`STARTING_WITH`	`StartingWith`, `IsStartingWith`, `StartsWith`
`TRUE`	`True`, `IsTrue`
`WITHIN`	`Within`, `IsWithin`

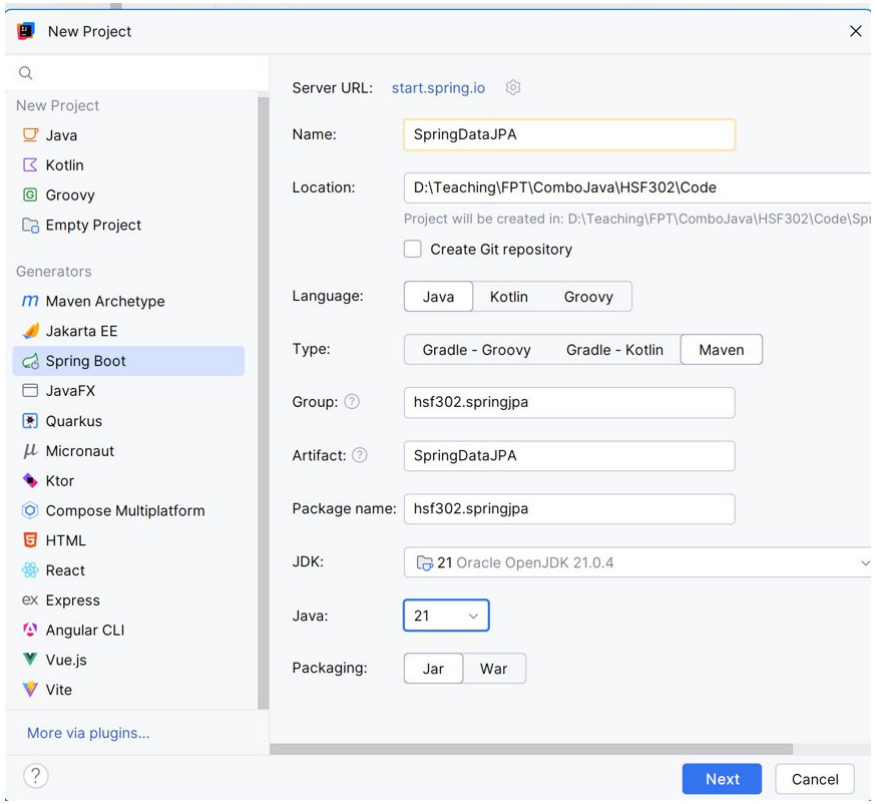
https://docs.spring.io/spring-data/jpa/reference/repositories/query-keywords-reference.html

FPT Education
FPT UNIVERSITY

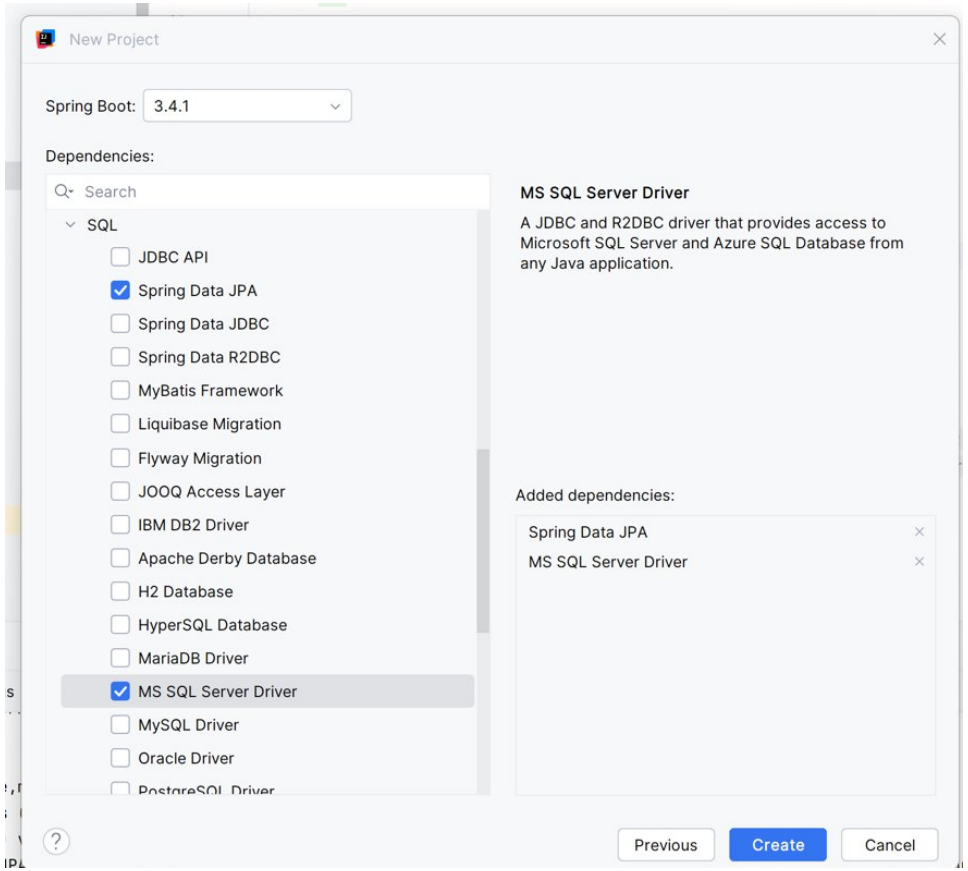
Java

Spring Data JPA Programming Demo (ManyToMany)

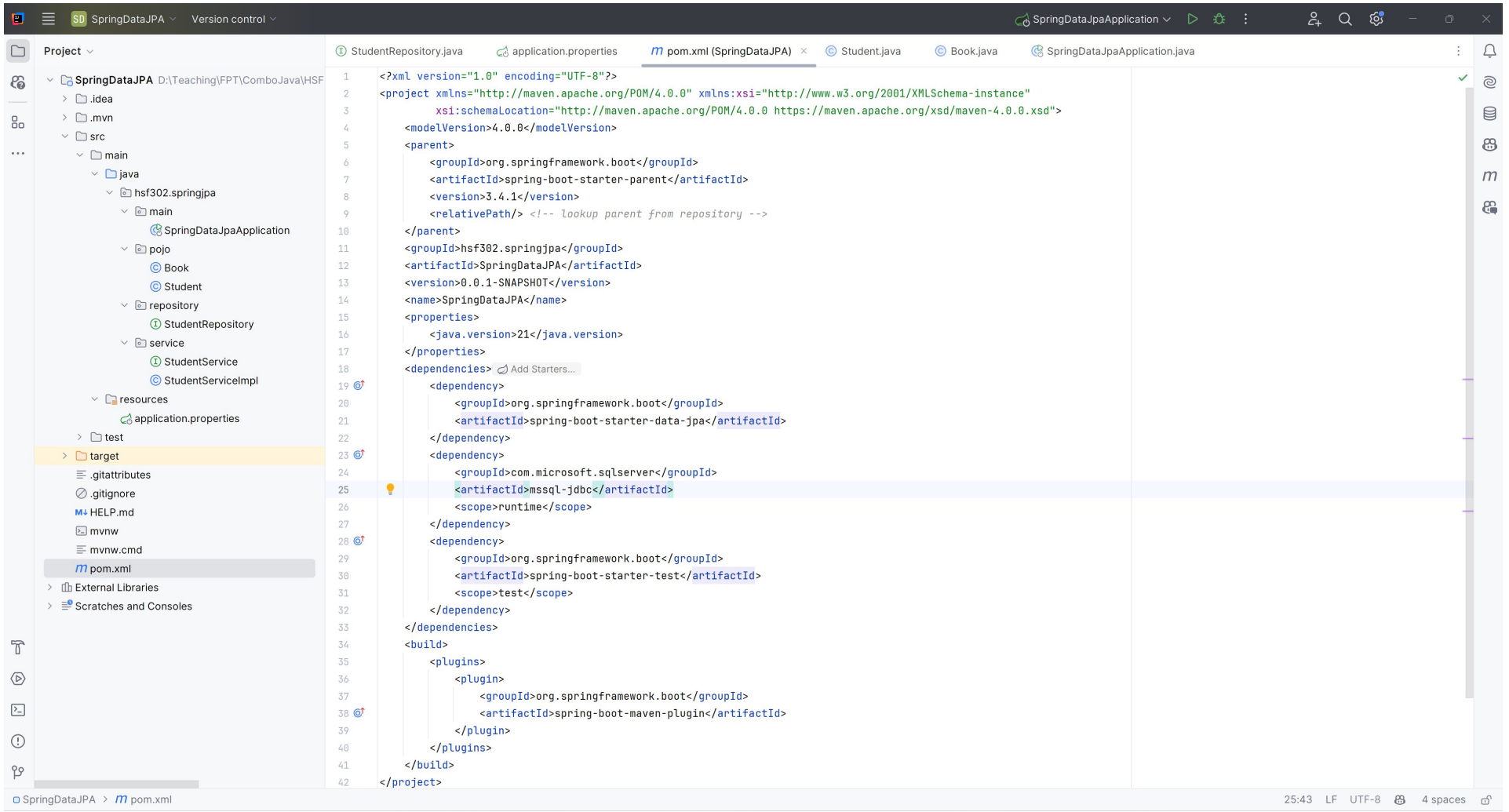
Open IntelliJ, File | New | Spring Boot -> Maven Project



Add Dependencies

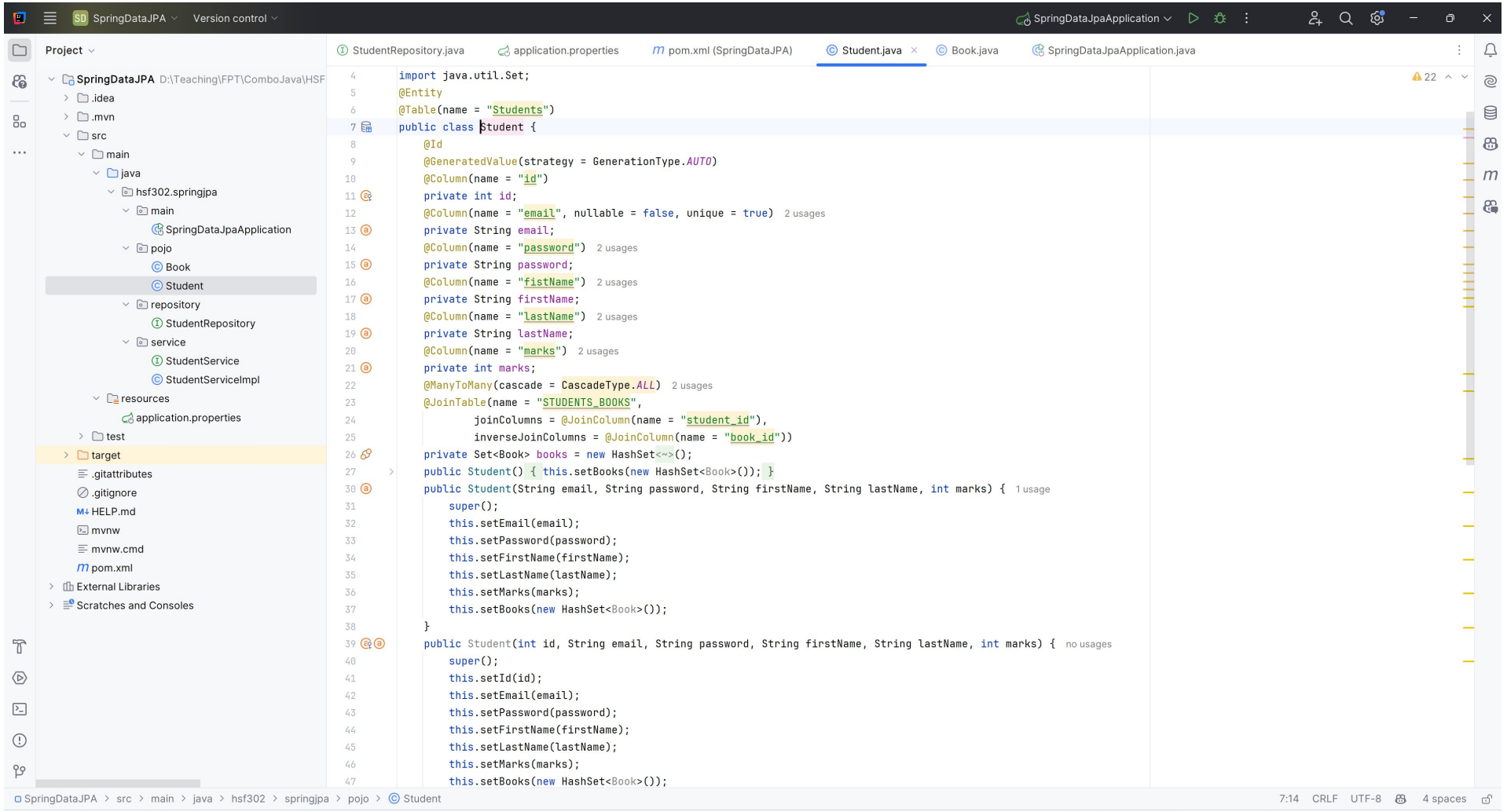


Create the Structure Project

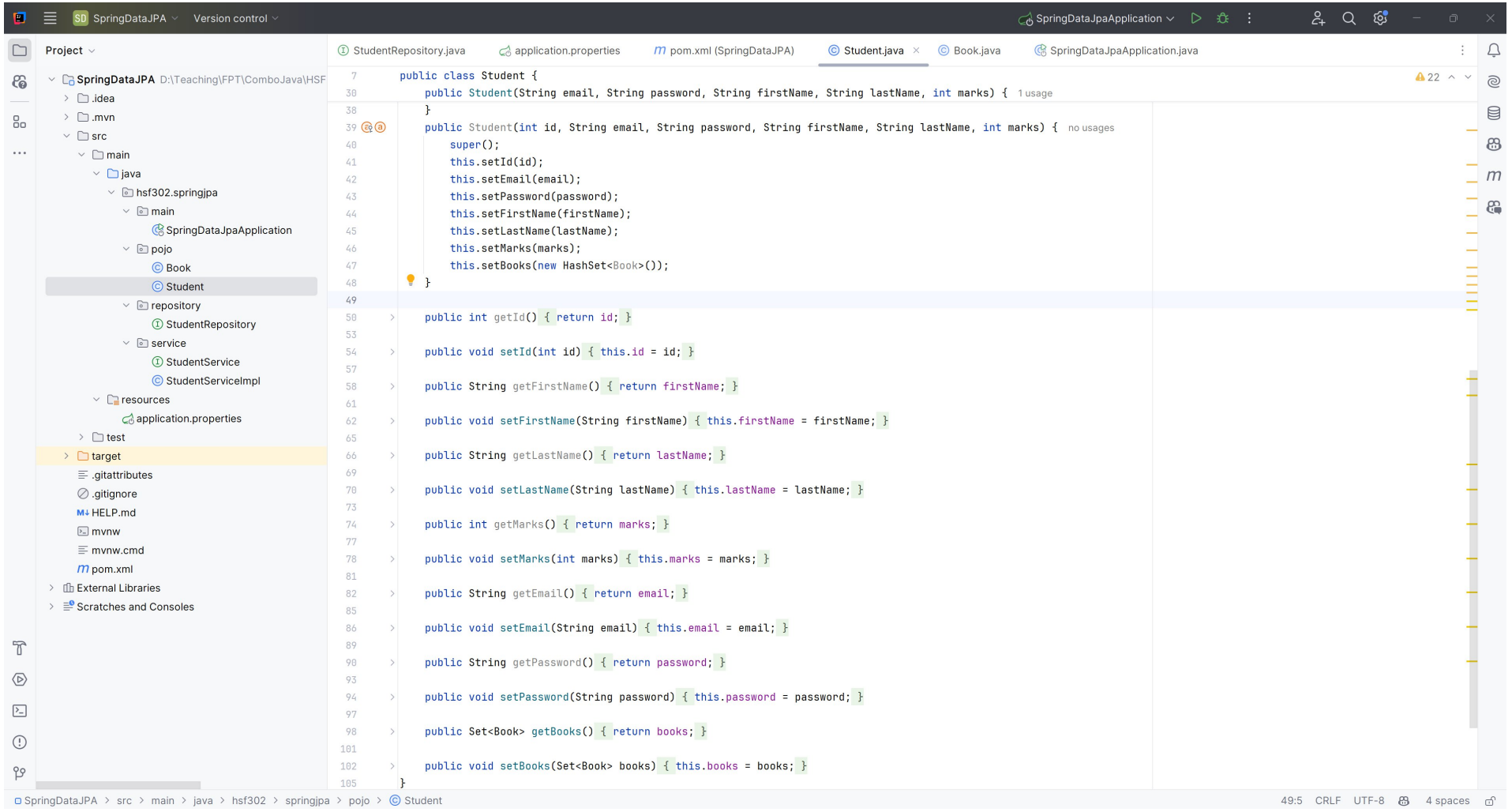


FPT Education
FPT UNIVERSITY

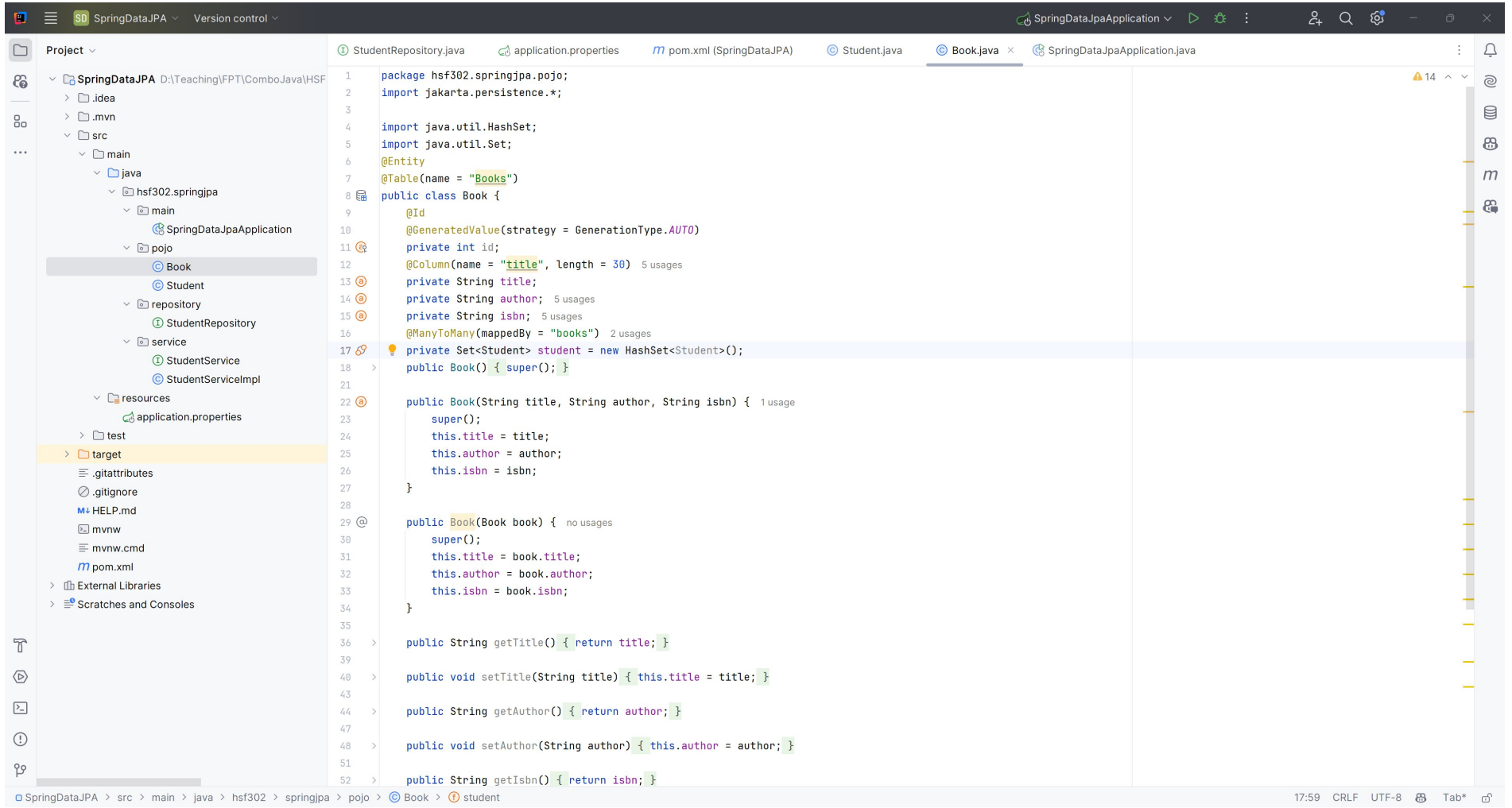
Create the Student.java in pojo



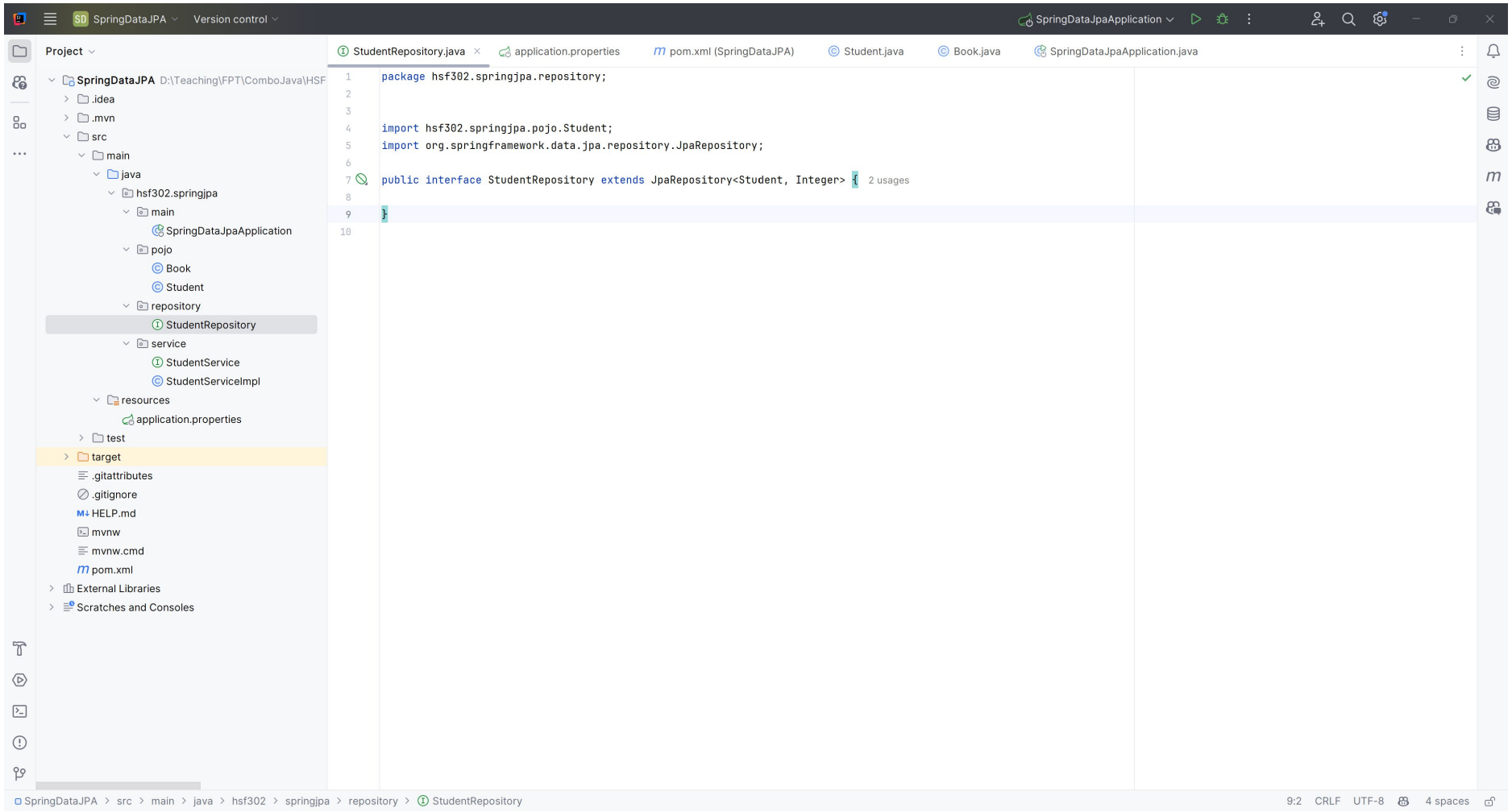
Create the Student.java in pojo



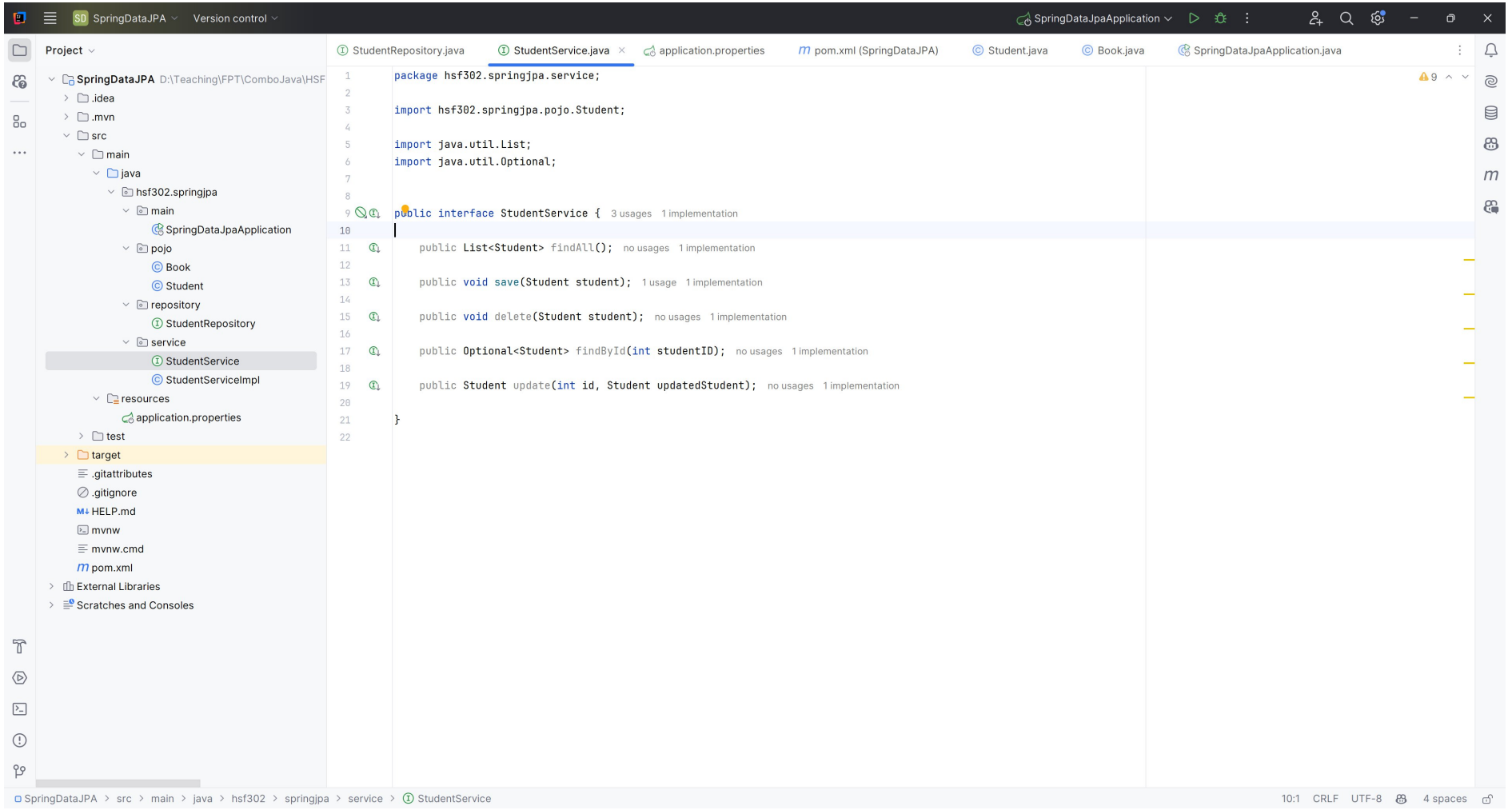
Create the Book.java in pojo



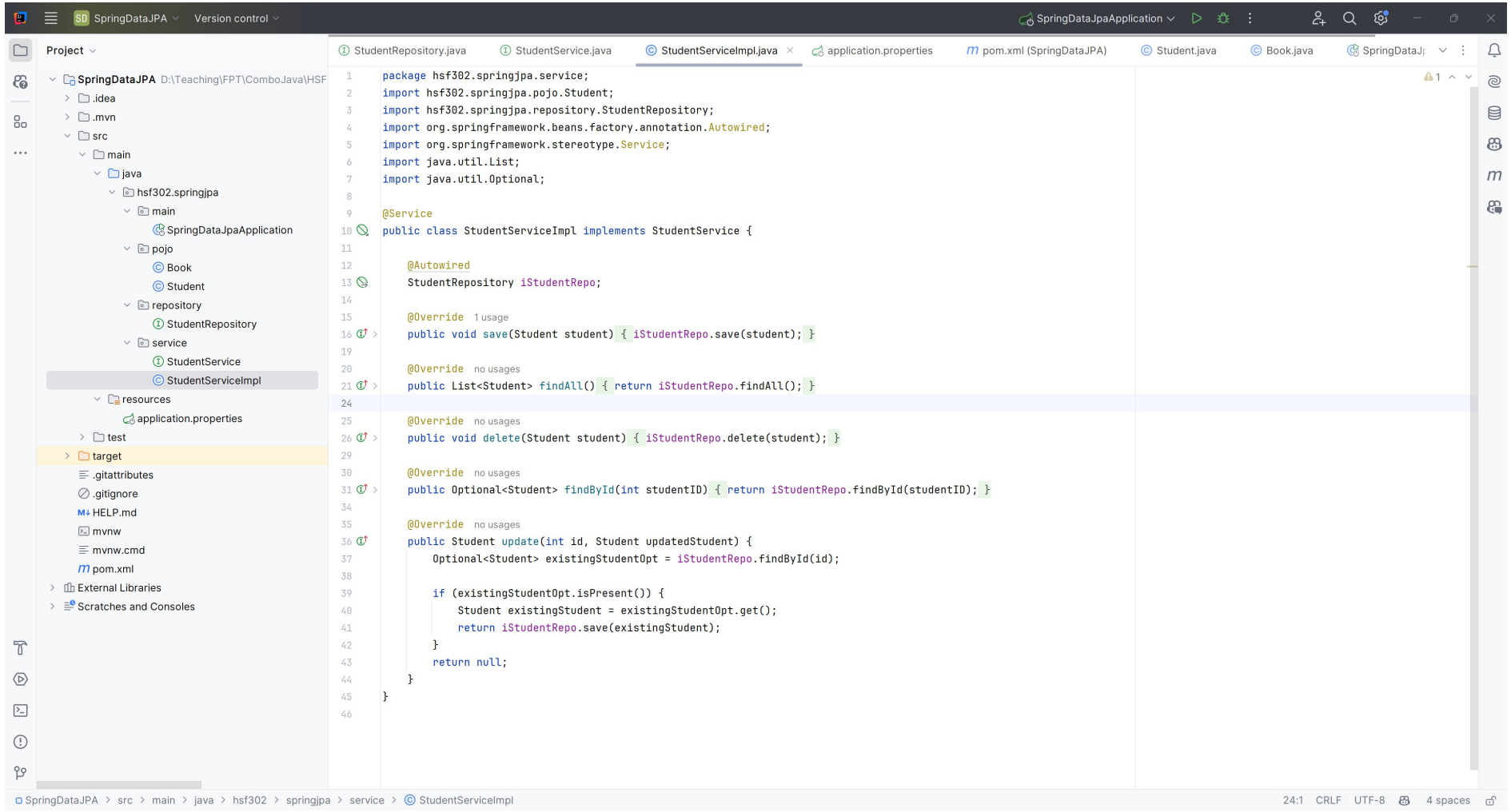
Create the StudentRepository.java in repository package



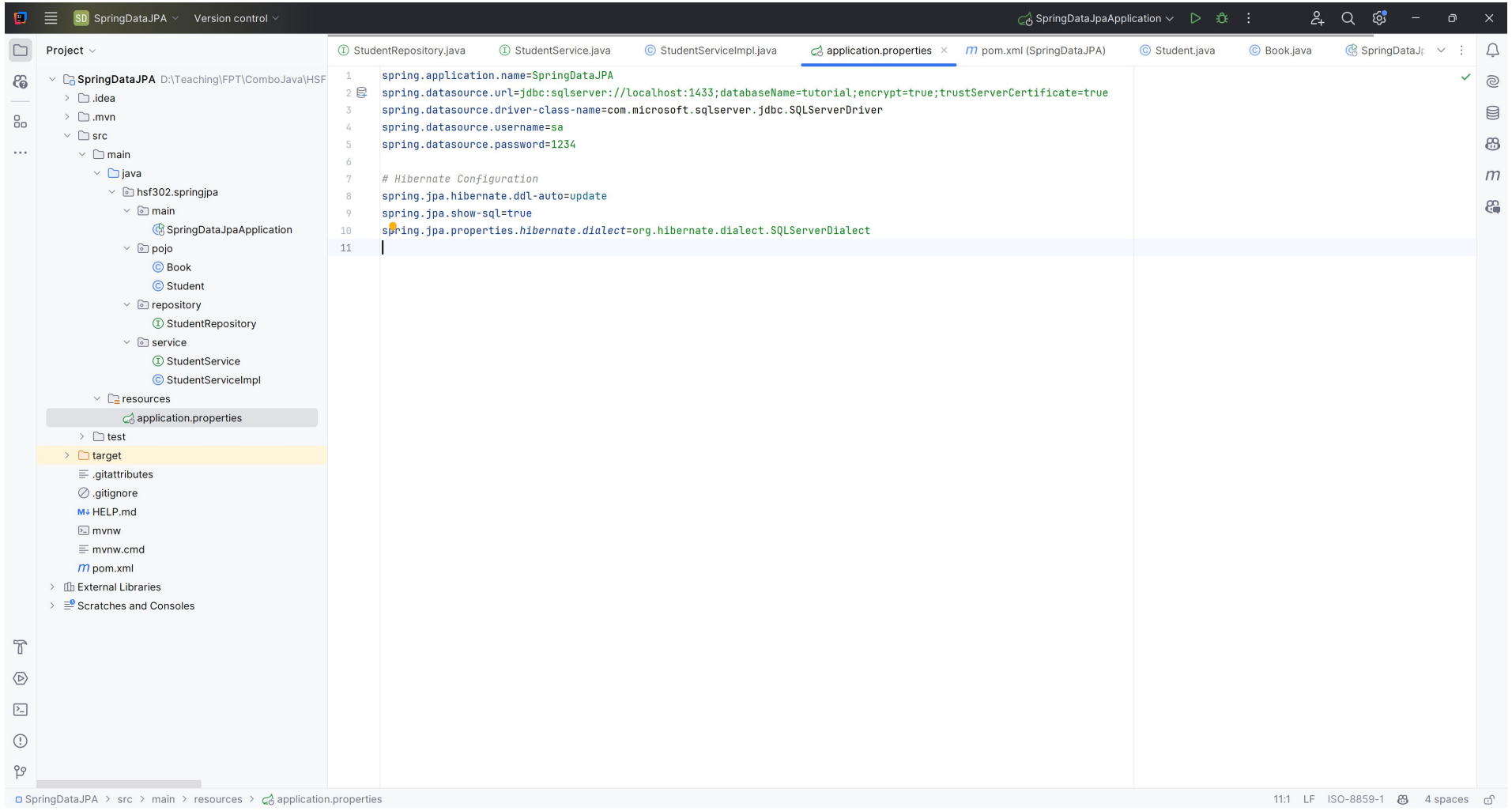
Create the StudentService.java in services package



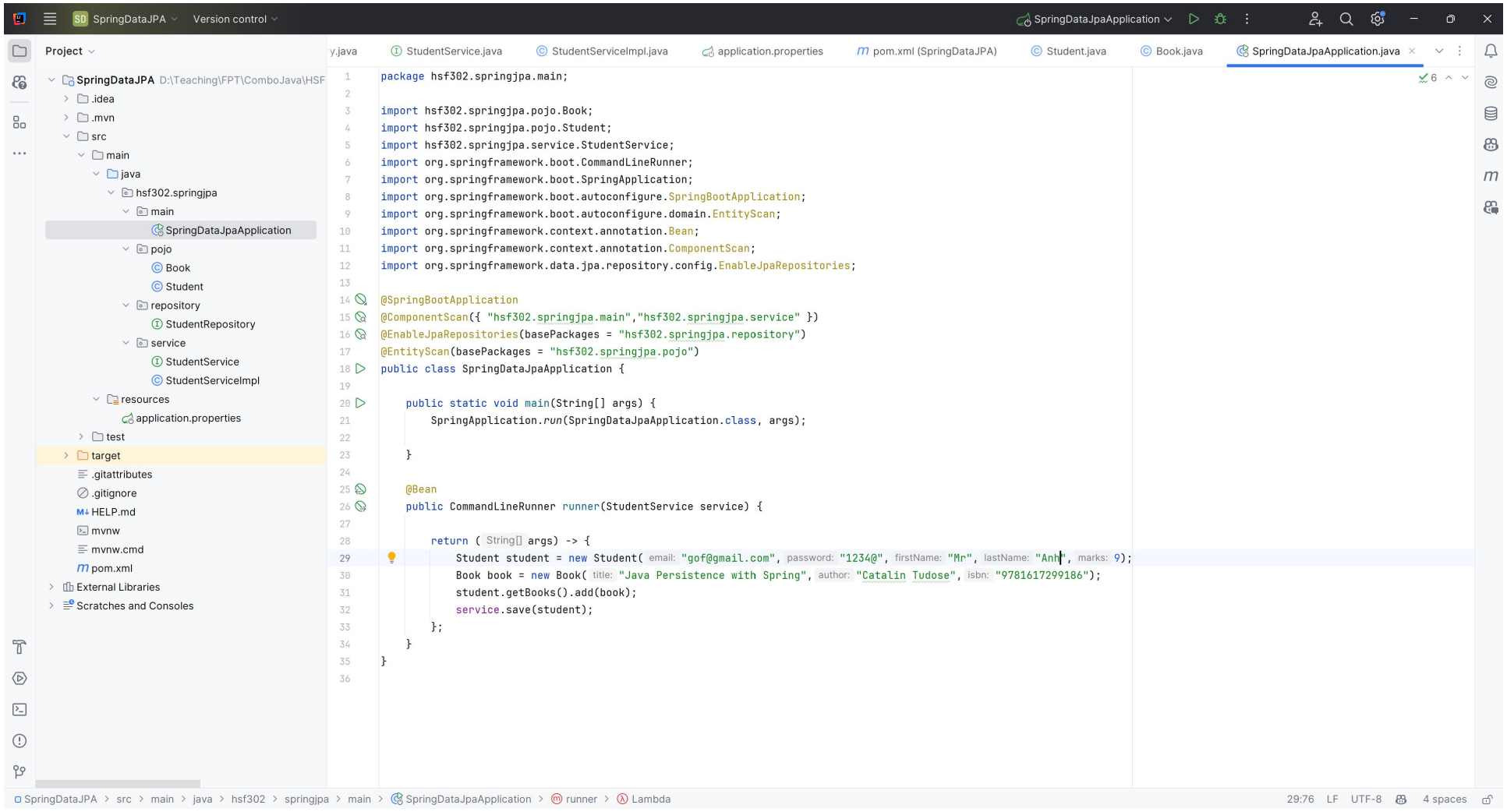
Create the StudentServiceImpl.java in services package



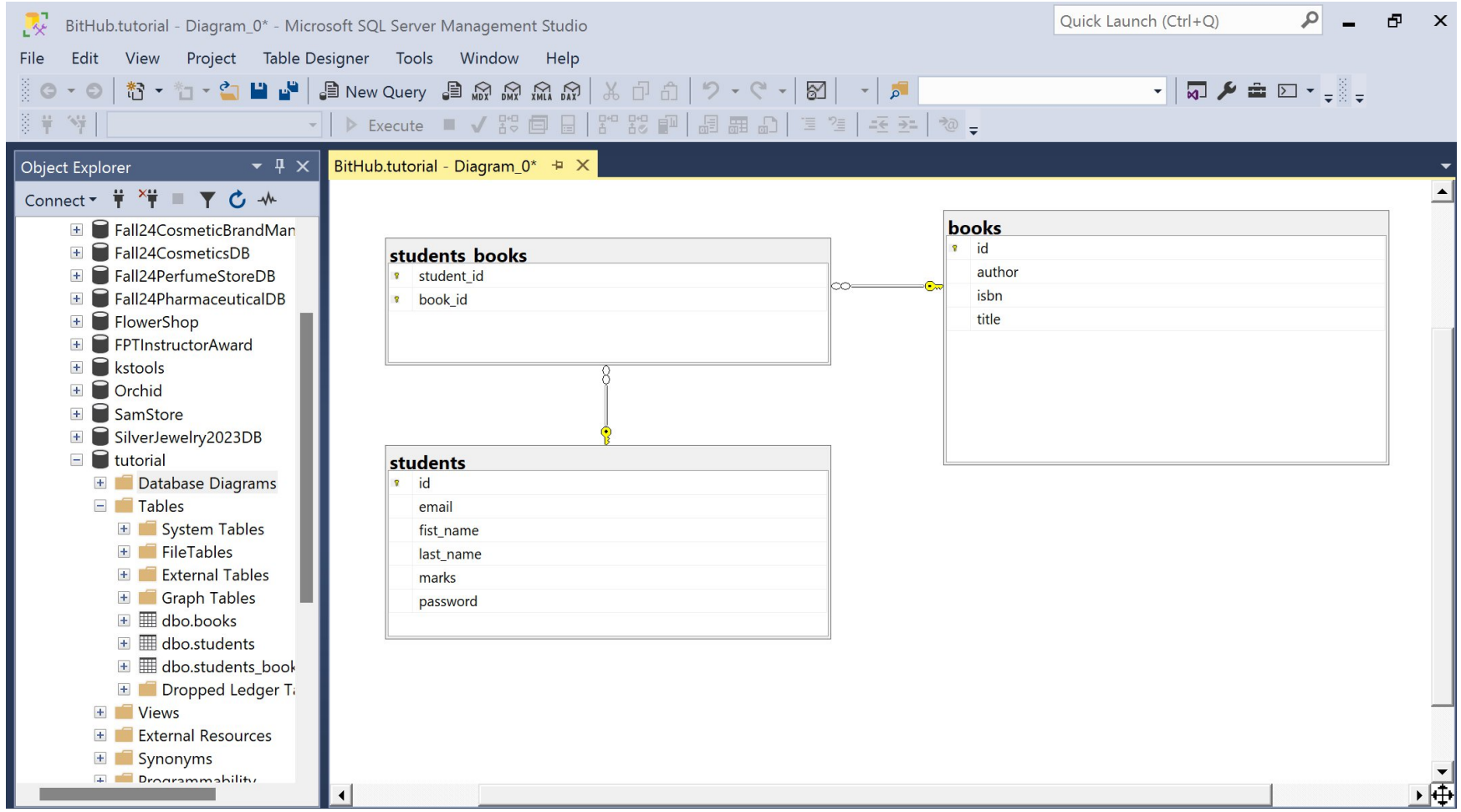
Create the application.properties in resources



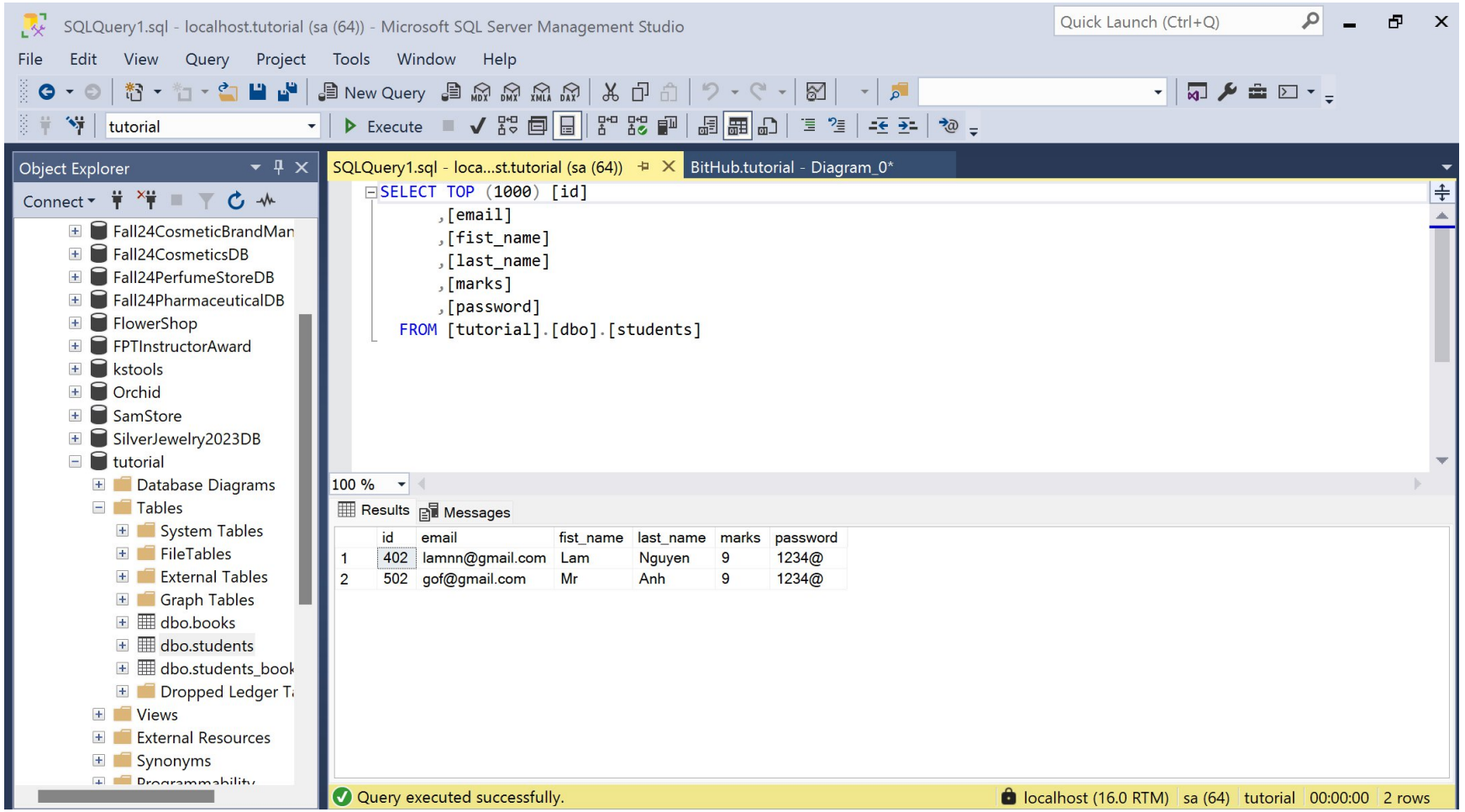
Edit main function



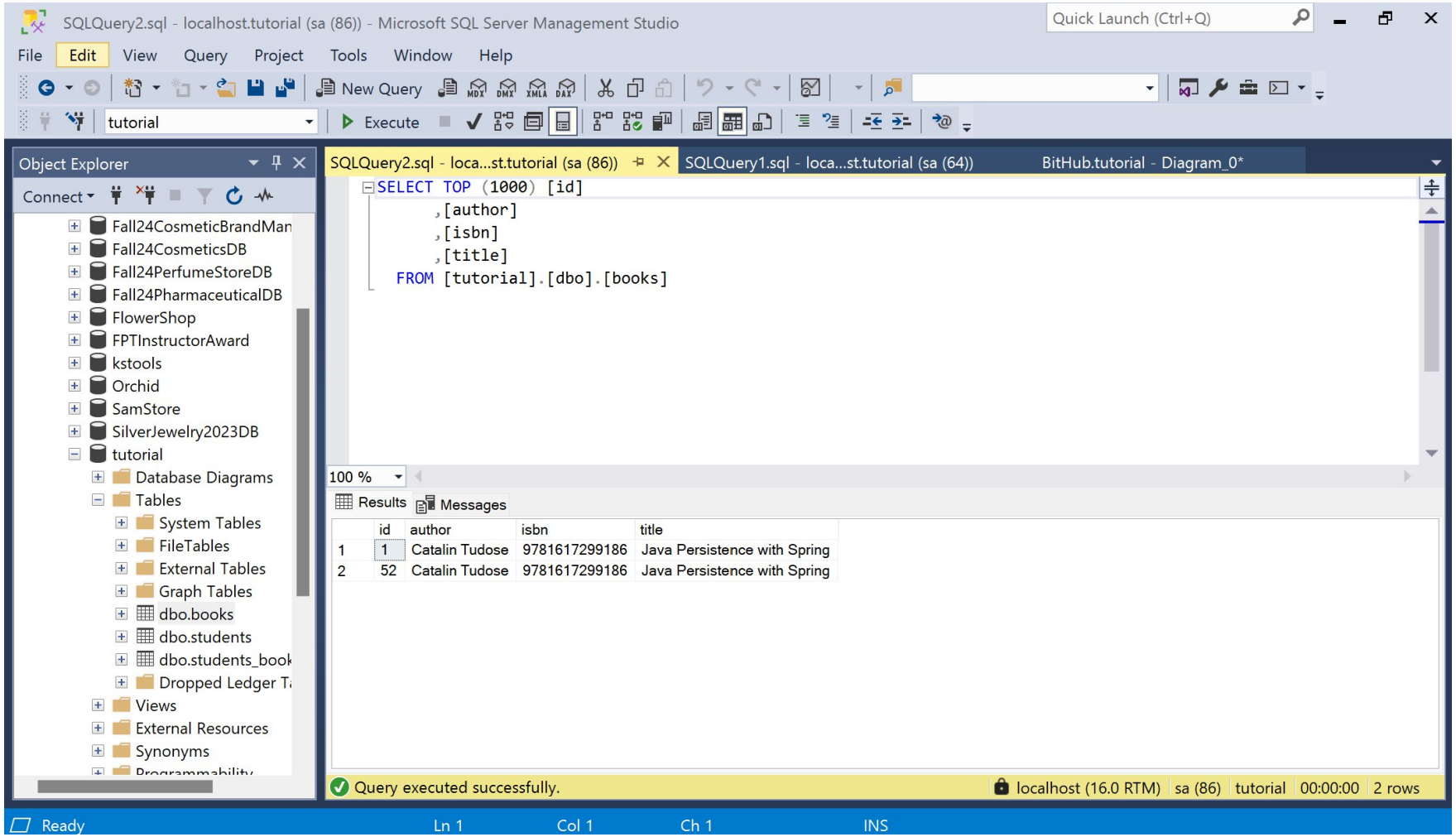
Run Program


BitHub.tutorial - Diagram_0* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Project Table Designer Tools Window Help
New Query
Execute
Object Explorer
Connect
Fall24CosmeticBrandMan
Fall24CosmeticsDB
Fall24PerfumeStoreDB
Fall24PharmaceuticalDB
FlowerShop
FPTInstructorAward
kstools
Orchid
SamStore
SilverJewelry2023DB
tutorial
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.books
dbo.students
dbo.students_book
Dropped Ledger T
Views
External Resources
Synonyms
BitHub.tutorial - Diagram_0*
students_books
student_id
book_id
books
id
author
isbn
title
students
id
email
fist_name
last_name
marks
password

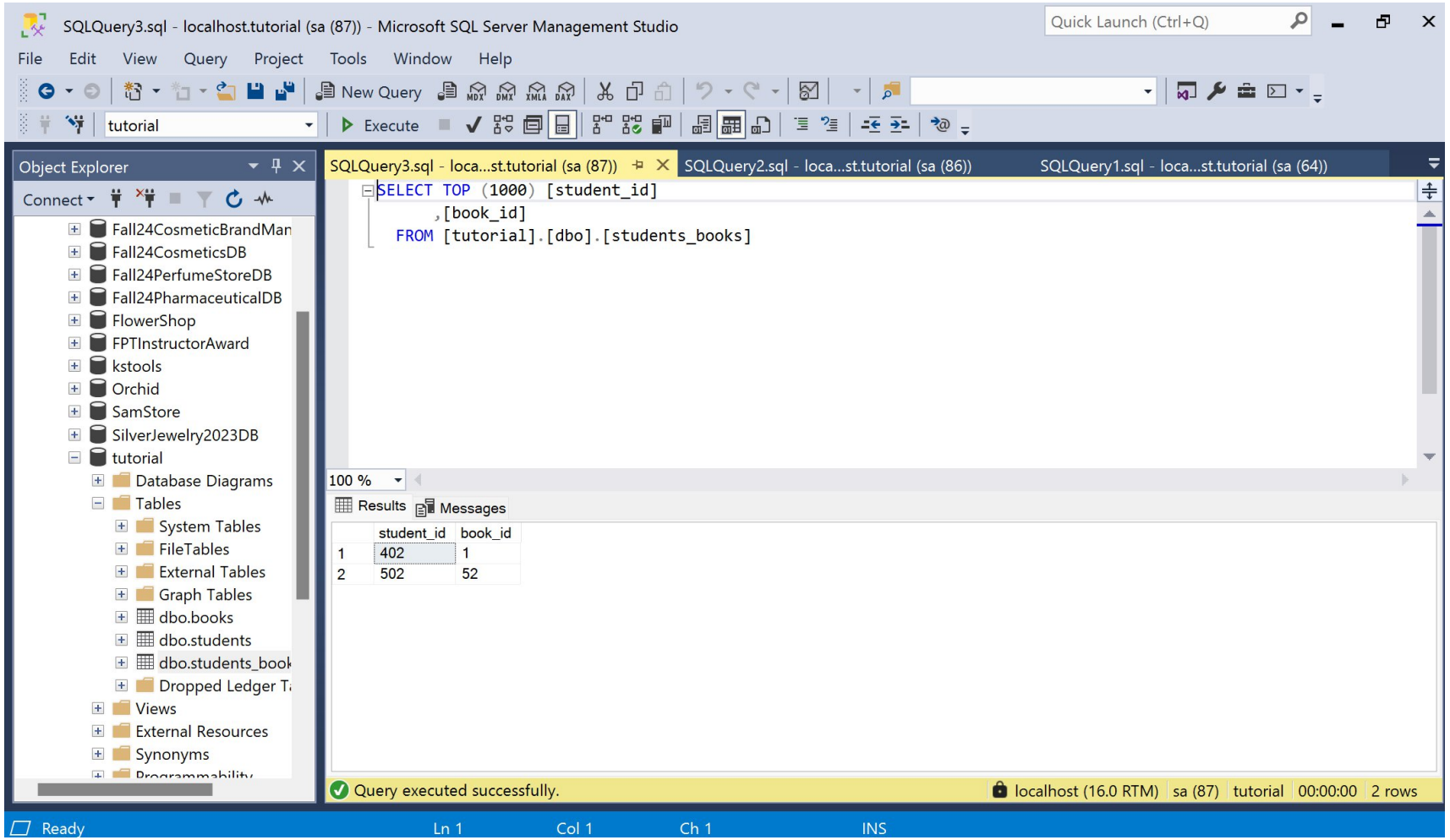
Result student table



Result book table



Result student-book table



Summary

Concepts were introduced:

- Spring Data JPA
- Advantages of using Spring Data JPA
- Key features of Spring Data JPA
  - Dependency Injection and Inversion of Control
  - N-Layer