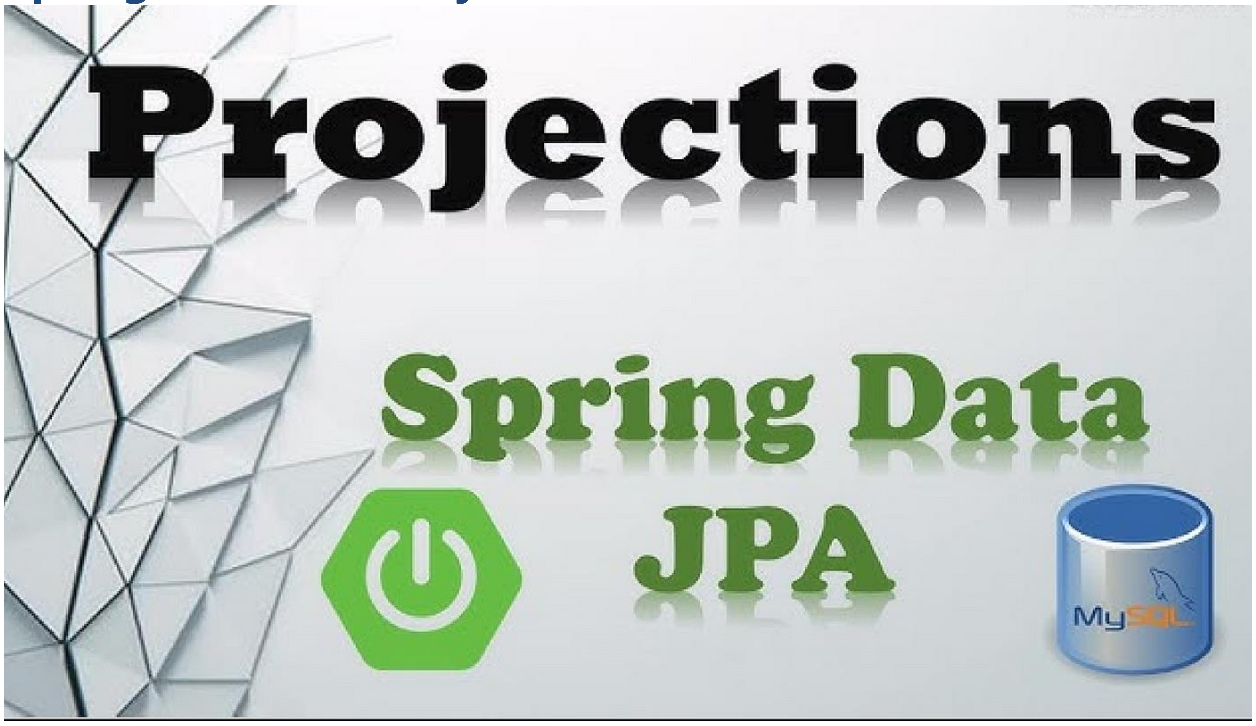
Spring Data JPA Projections



What is Projection?

If you already know about Spring Data JPA, you must be knowing that all the query methods of Repository classes will return an entity object. There may be cases where we do not want an entire entity from the query method.
We may be **interested only in few attributes of that entity or subset of that entity** with some manipulation in it. In those cases, we will be using Projection which **projects us only the required data out of entire entity class**.

Broadly there are two types of projections, **Interface-based Projections, and Class-based Projections**..

- **interface-based Projections**
- **Class-based (DTO) Projections**

1. Interface-based Projections

You define a **Java interface** with **getter methods** that match **field names** (or aliases) from your entity

Class-based (DTO) Projections

You create a **class (DTO)** with a **constructor** that matches the fields you want.

```
package com.cts.projections;

public interface BookTitleAndAuthorProjection {
    String getTitle();

    AuthorInfo getAuthor();

    interface AuthorInfo {
        String getFirstName();
        String getLastName();
    }
}
```

- **This is a Projection interface.**
- It tells Spring Data JPA:
- "When fetching data, **I don't want the whole Book entity**, I **only want specific fields**: **title**, and from author, only **firstName and lastName."**

Endpoint:

```
   //http://localhost:2025/books/titles-authors
    @GetMapping("/titles-authors")
    public List<BookTitleAndAuthorProjection> getTitlesAndAuthors() {
        return bookService.getTitlesAndAuthors();
    }
```

In bookService:

```
public List<BookTitleAndAuthorProjection> getTitlesAndAuthors() {
        return bookRepository.findByPublishedYearGreaterThan(2000);
     }
```

bookRepository probably has:

```
public interface BookRepository extends JpaRepository<Book, Long> {

    // Interface-based Projection
    List<BookTitleAndAuthorProjection>
findByPublishedYearGreaterThan(int year);
```

So, what does projection mean here?

Projection = selecting only parts of an entity, not the full entity, when fetching from the database.

- You **don't** fetch the entire `Book` object (**which may have lots of fields like ISBN, price, description, publisher, etc.**)
- You **only fetch**:
  - **title**
  - **author.firstName**
  - **author.lastName**

This **improves performance** by:

- **Reducing the amount of data fetched from database**
- **Reducing memory usage**
- **Making API responses faster**
- **2000 = only fetch books published after year 2000.**

Quick analogy :

Imagine you have a library with books.
A *full book* = 300 pages.
A *projection* = just the book's title and author name printed on a sticky note.
Much lighter to carry, right?

Why does Spring Data JPA support projection like this?

Because sometimes you **don't care about the full entity** — you just need a **small, specific view** of the data for your API or UI.

Spring Data automatically maps only the fields defined in projection interface!