Lecture 22: Simple Vector Store

Why Do We Need It?

- We already have product data (title, description, price, category, and features).
- To search semantically (not just by exact words), we need to store this data in a **vector database**.
- Each product (or chunk of text) is converted into embeddings and stored in the **VectorStore**.

Steps to Implement

- **Load the data file**
  - File: product_details.txt stored in resources.
  - Use TextReader (from Spring AI) to read file content.
- **Break into Chunks**
  - Large documents must be split into smaller pieces (chunks).
  - Each chunk is embedded and stored in a vector database.
- **Create DataInitializer Class**
  - Marked with @Component.
  - Method initData() runs at startup (@PostConstruct).
  - Reads the file, splits it into chunks, and stores documents in VectorStore.
- **VectorStore Setup**
  - We use **SimpleVectorStore** (in-memory, good for learning).
  - Defined as a bean in AppConfig.
  - Injected wherever needed (e.g., controller, initializer).

Code Implementation:

```java
@Component
public class DataInitializer {

    @Autowired
    private VectorStore vectorStore;

    @PostConstruct
    public void initData() {
        TextReader textReader = new TextReader(new ClassPathResource("product_details.txt"));

//        TokenTextSplitter splitter = new TokenTextSplitter();
        TokenTextSplitter splitter = new TokenTextSplitter(500, 30, 20, 500, false);
        List<Document> documents
                = splitter.split(textReader.get());
        vectorStore.add(documents);

    }
}
```

Key Takeaway:

- SimpleVectorStore helps you load documents, split them into embeddings, and store them for **semantic search**. Start with it for practice, then move to real vector DBs like **PgVector** for production.