Lecture 23: Token Text Splitter

Why Do We Need a Vector Store?

- We use a **Vector Store** to perform **semantic search** or **similarity search**.
- Instead of exact keyword matching, semantic search retrieves results based on meaning.
- Vector Store methods:
    - add() → add documents.
    - delete() → remove documents.
    - similaritySearch() → search for relevant results.

Controller Setup:

- Inject VectorStore using @Autowired (or constructor injection).
- In the controller, call vectorStore.similaritySearch(text) to return matching documents.
- Return type: **List of Documents**.

Document Splitting:

- Input text is **split into chunks (tokens/characters)** before being stored.
- Chunk size affects accuracy:
    - Smaller chunks → more precise, but fragmented.
    - Larger chunks → broader context, but may mix irrelevant data.

Running an Example Search:

- **Example:** searching "tea" returned results like water bottles, kettles, and mugs.
- Sometimes irrelevant results appear (AI hallucination).
- Fine-tuning chunk size improves accuracy.

Limiting Search Results:

- Use SearchRequest.builder() to customize search:
    - query(text) → search query
    - topK(2) → limit results (e.g., top 2 documents only)

```
@PostMapping("/api/product")
public List<Document> getProducts(@RequestParam String text) {
    return vectorStore.similaritySearch(
        SearchRequest.builder()
            .query(text)
            .topK(2)
            .build()
    );
}
```

Key Takeaways:

- **Vector Store** is central for similarity/semantic search.
- Always control **chunk size** for better accuracy.
- Use **SearchRequest** for filtering results and limiting output.