Externalizing Configuration with @Value Annotation

Spring Feature Description

Spring allows you to externalize configuration to properties files, enabling you to modify application behavior without changing code. The @Value annotation can inject property values into beans, facilitating dynamic configuration.

Why Do We Need It in Flight Application?

Currently, the path to the CSV file is hardcoded in CSVDataLoader, making it difficult to change without modifying code. By externalizing this configuration, we can easily change the file path as needed, enhancing flexibility and maintainability.

Refactoring Application

We will:

- Create an application.properties file.
- Move the CSV file path into this properties file.
- Use the @Value annotation to inject the property into CSVDataLoader.

The Updated Code

- **Create application.properties**

Add a file named application.properties in src/main/resources:

```
csv.file.path=flights_small.csv
```

- **Inject Property Using @Value**

Modify CSVDataLoader to use @Value:

```java
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

@Component
public class CSVDataLoader {

    @Value("${csv.file.path}")
    private String csvFilePath;
    // Existing code...

    public void loadData() throws Exception {
        List<String> lines = Files.readAllLines(Paths.get(csvFilePath));
        // Existing code...
    }
}
```

Benefits of Using Externalized Configuration

- **Flexibility**: Easily change configuration without modifying code.
- **Environment-Specific Settings**: Different properties files can be used for different environments.
- **Maintainability**: Centralizes configuration, making it easier to manage.

Further Advices

For sensitive configurations, consider using encrypted properties or external configuration servers. Also, explore using @ConfigurationProperties for complex configuration mappings.