# Injection Techniques

## Overview

In this lab you’ll enhances your “online retailer” application from the previous lab, to make use of SpEL value injection and command-line arguments.

## IntelliJ starter project

If you’re happy to continue where you left off in the previous lab, use the following project:

* **student\student-online-retailer**

If you’d prefer a fresh start, use the solution project from the previous lab instead:

* **solutions\solution-beans-dependencyinjection**

## IntelliJ solution project

The solution project for this lab is located here:

* **solutions\solution-injection-techniques**

## Roadmap

There are 3 exercises in this lab, of which the last exercise is “if time permits”. Here is a brief summary of the tasks you will perform in each exercise; more detailed instructions follow later:

1. Defining a component class named **Timestamp**
2. Using SpEL value injection
3. (If time permits) Using command-line arguments

## Exercise 1: Defining a component class named Timestamp

Define a component class named **Timestamp**. In the class, define an instance variable of type **LocalDateTime** to hold the date and time when the bean was created. Initialize this field with the current data and time, by calling **LocalDateTime.now()**.

Now define two methods in your class:

* **date()** - returns the date part of the **LocalDateTime** field
* **time()** - returns the time part of the **LocalDateTime** field

When your Spring Boot application starts, it will create a singleton bean named **timestamp**. This will come in handy in Exercise 2 below…

## Exercise 2: Using SpEL value injection

Define a component class named **ShoppingSession**. The purpose of this class is to display the date and/or time when the user’s shopping session started.

Define two instance variables as follows:

* **LocalDate startDate**  
  Initialize this field via value injection, using the SpEL expression **timestamp.date**.
* **LocalTime startTime**  
  Initialize this field via value injection, using the SpEL expression **timestamp.time**.

Now define a method named **displayStartDateTime()**. In the method, display the **startDate** and **startTime** values on the console.

Now go to the **Application** class and add some code to get the **ShoppingSession** bean and invoke its **displayStartDateTime()** method. Verify the date/time info is displayed on the console.**Exercise 3 (If time permits): Using command-line arguments**

Enhance your **ShoppingSession** class so that it can access command-line arguments.   
Hint: Define a constructor and inject an **ApplicationArguments** parameter.

The idea is to allow the user to choose how they want to see timestamp info displayed, via a command-line argument named **displayTimestampMode**. This is how we want it to work:

**--displayTimestampMode=date** Display just the date

**--displayTimestampMode=time** Display just the time

**--displayTimestampMode=both** Display both the date and time

**--displayTimestampMode=none** Display neither the date nor time

Modify your **displayStartDateTime()** method so that it makes use of this command-line argument to decide what to display (date, time, both, or neither).