Building Web App Using Spring MVC, Hibernate, Bootstrap, and REST Services

INTRODUCTION TO SPRING FRAMEWORK

Sekhar Srinivasan

@sekharonline4u www.sekhartheguru.net

What We Will Cover in This Course

Course Outline

Introduction to Spring Framework

Configuring your Environment for Spring MVC Application Development

Creating Controllers and Views

Handling Spring Tags and Data Bindings

Handling Request Parameters and Request Mappings

Applying Built-in Validation Rules

Course Outline

Performing CRUD Operations with Hibernate

Modifying the Frond-End with Bootstrap

Securing the Application

Managing Exceptions through Aspect Oriented Programming

Creating REST Services

Directing Users based on Role with Spring Web Flow

Introduction to Spring Core Framework

Why Spring?
Spring Framework is used to simplify Java Enterprise Development

Goals of Spring

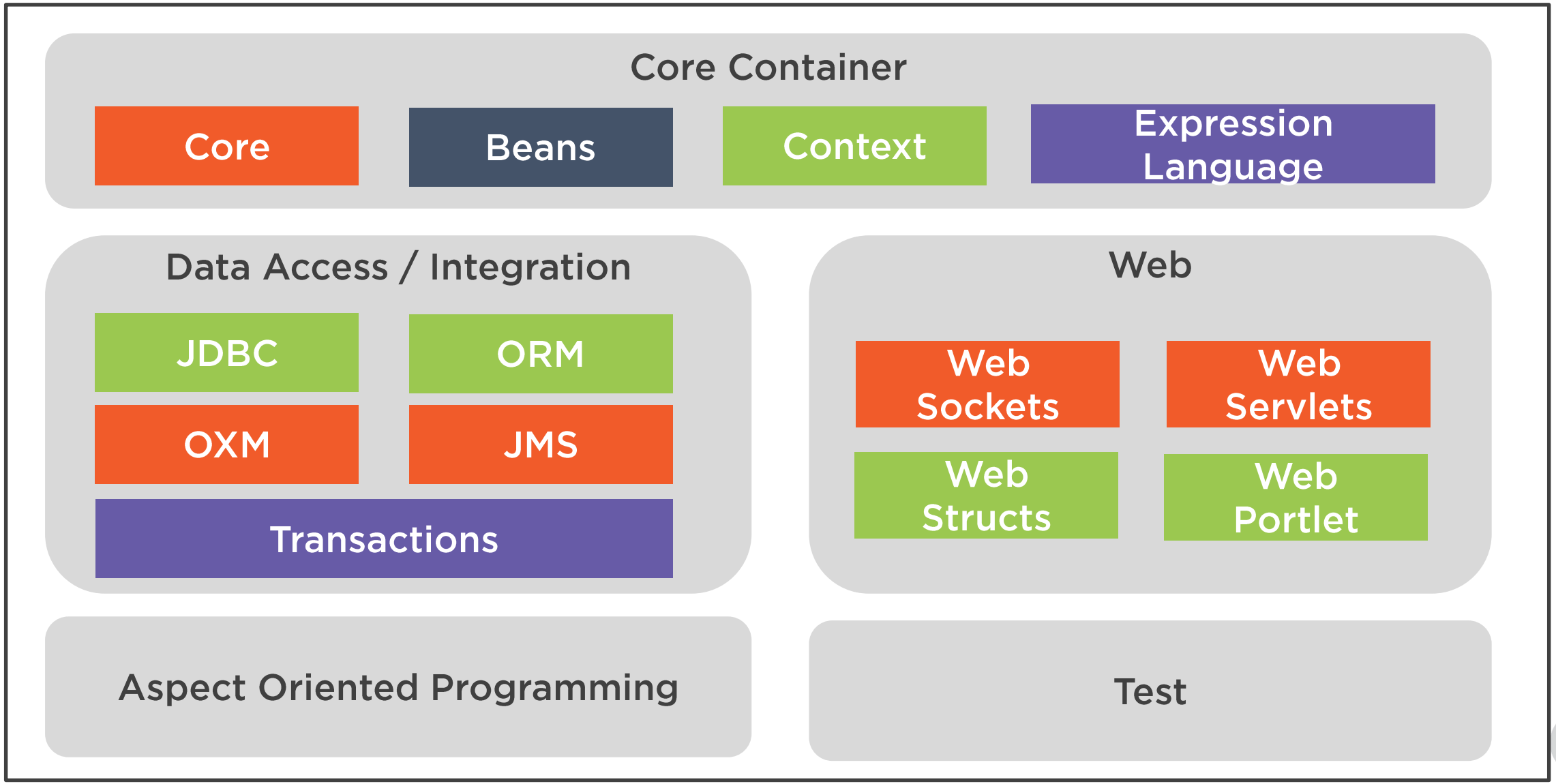
Light Weight Development Model

Loose coupling using Dependency Injection

Declarative Programming

Reduce Boilerplate Coding

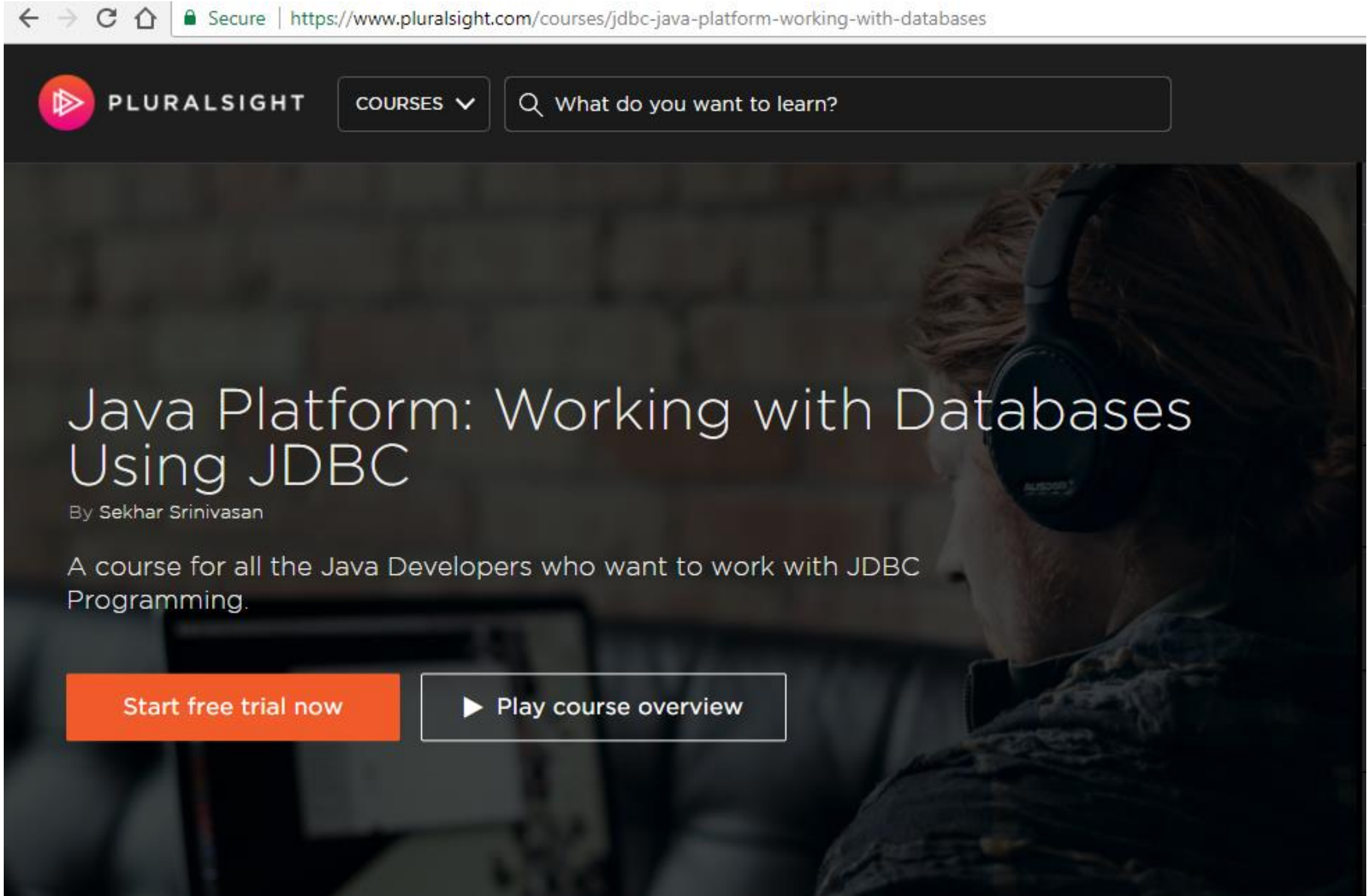
Spring Architecture



Setting up Development Environment for Spring

Prerequisite

Eclipse


Secure | https://www.pluralsight.com/courses/jdbc-java-platform-working-with-databases
PLURALSIGHT
COURSES
What do you want to learn?
Java Platform: Working with Databases Using JDBC
By Sekhar Srinivasan
A course for all the Java Developers who want to work with JDBC Programming.
Start free trial now
Play course overview

Setting up Development Environment for Spring

1. Create Eclipse Project
2. Download Spring Jar Files
3. Download Common Logging Jar Files
4. Add the Jar files to Eclipse Project

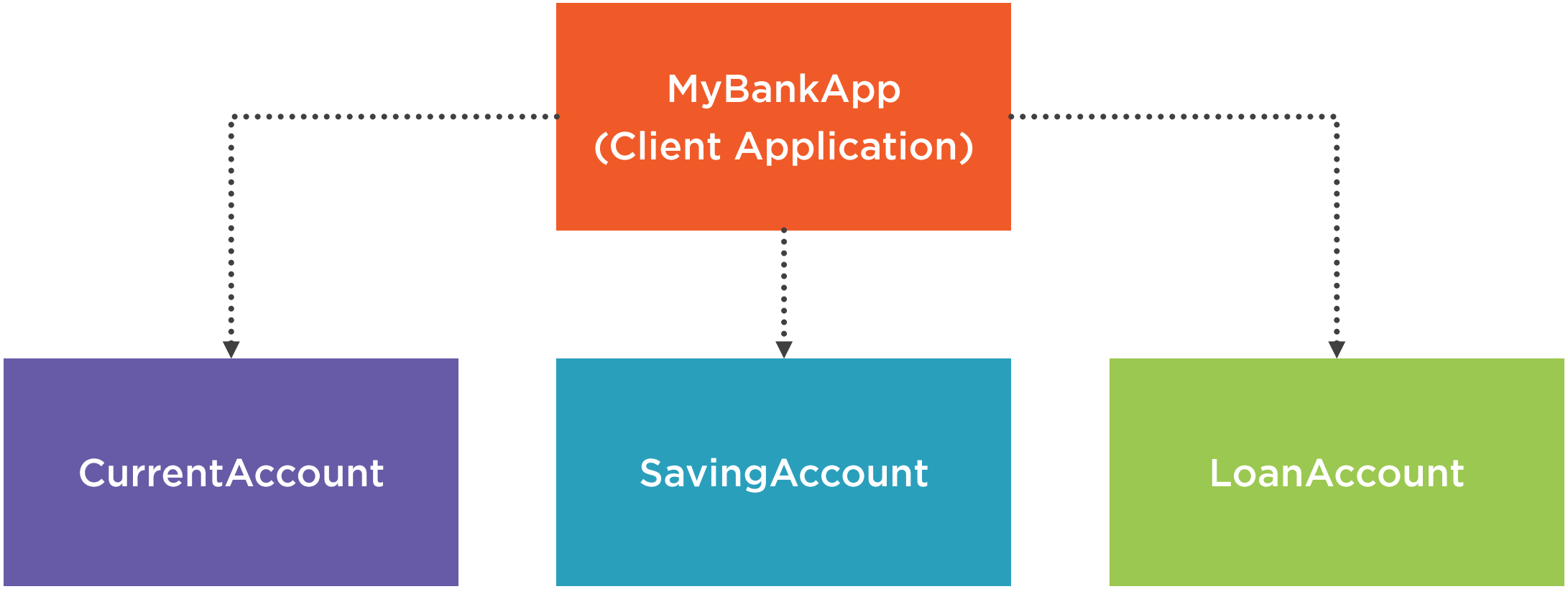
Why Inversion of Control?

Design process of externalizing the construction and management of your objects

Demo

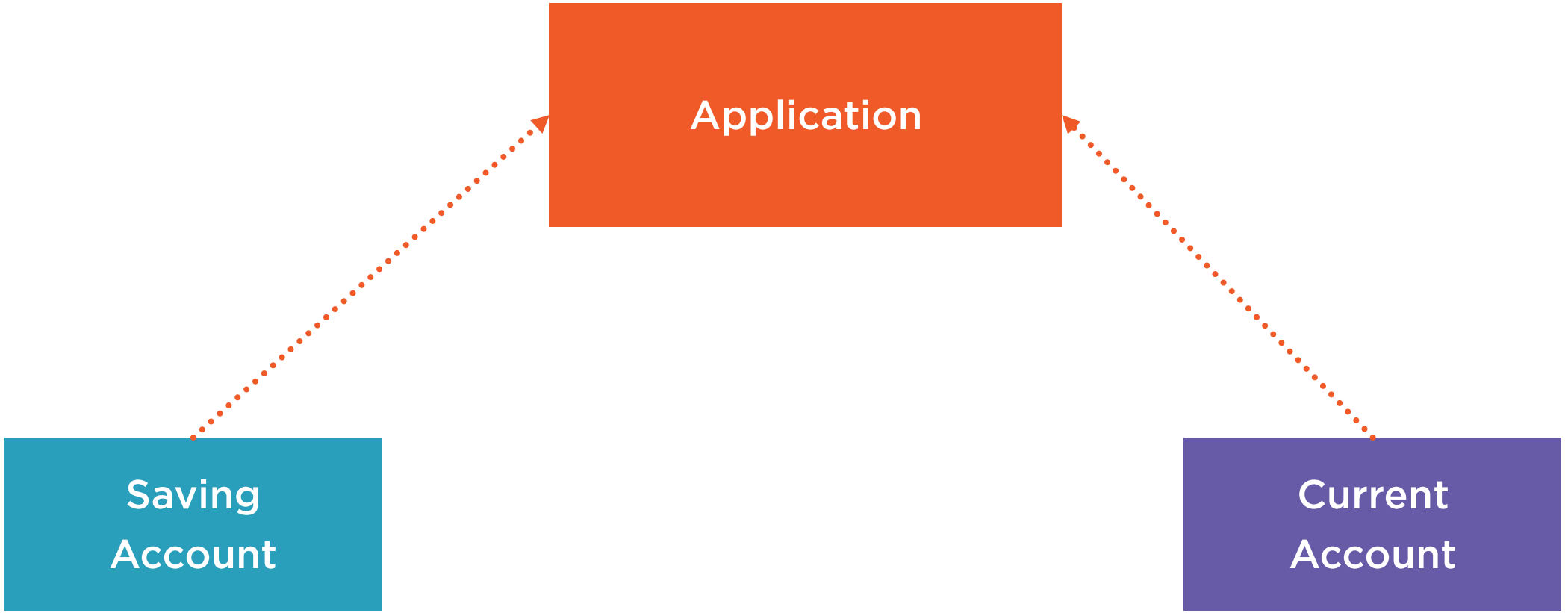
Why Inversion of Control

Coding Senario

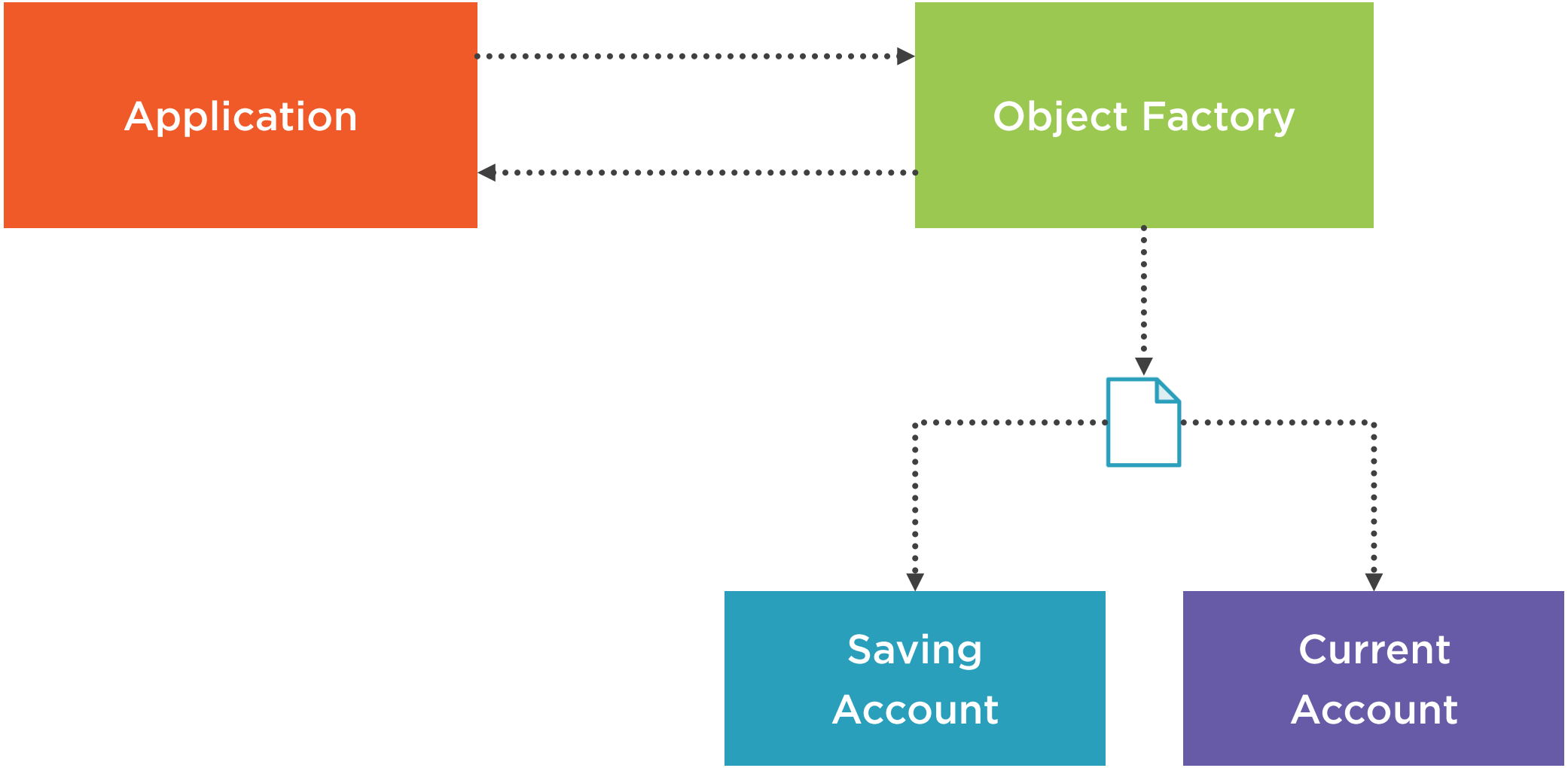


Spring Inversion of Control

Without Spring Inversion of Control



Spring Inversion of Control



Spring Core Framework Primary Functionalities

Create and Manage the Object

(Inversion of Control)

Inject Object's Dependencies

(Dependency Injection)

Spring Core Configuration

XML Configuration File

Java Annotations

Java Code

Spring Development Process

1. Configure Spring Beans
2. Create a Spring Container
3. Retrieve Beans information from Spring Container

```
<beans …>

   <bean id=“myCurrentAccount”

   class=“com.ps.springDemos.CurrentAccount” />

</beans>
```

Step 1: Configure Spring Beans

File: applicationContext.xml

```
ClassPathXmlApplicationContext

AnnotationConfigApplicationContext

GenericWebApplicationContext

Others…
```

Step 2: Create a Spring Container

Spring Container is also known as ApplicationContext

```
ClassPathXmlApplicationContext context = new

ClassPathXmlApplicationContext(“applicationContext.xml”);
```

Step 2: Create a Spring Container

Spring Container is also known as ApplicationContext

```
InterfaceName objectName = context.getBean( “beanId” ,
                                    InterfaceName.class );
```

Step 3 : Retrieve Beans from Container

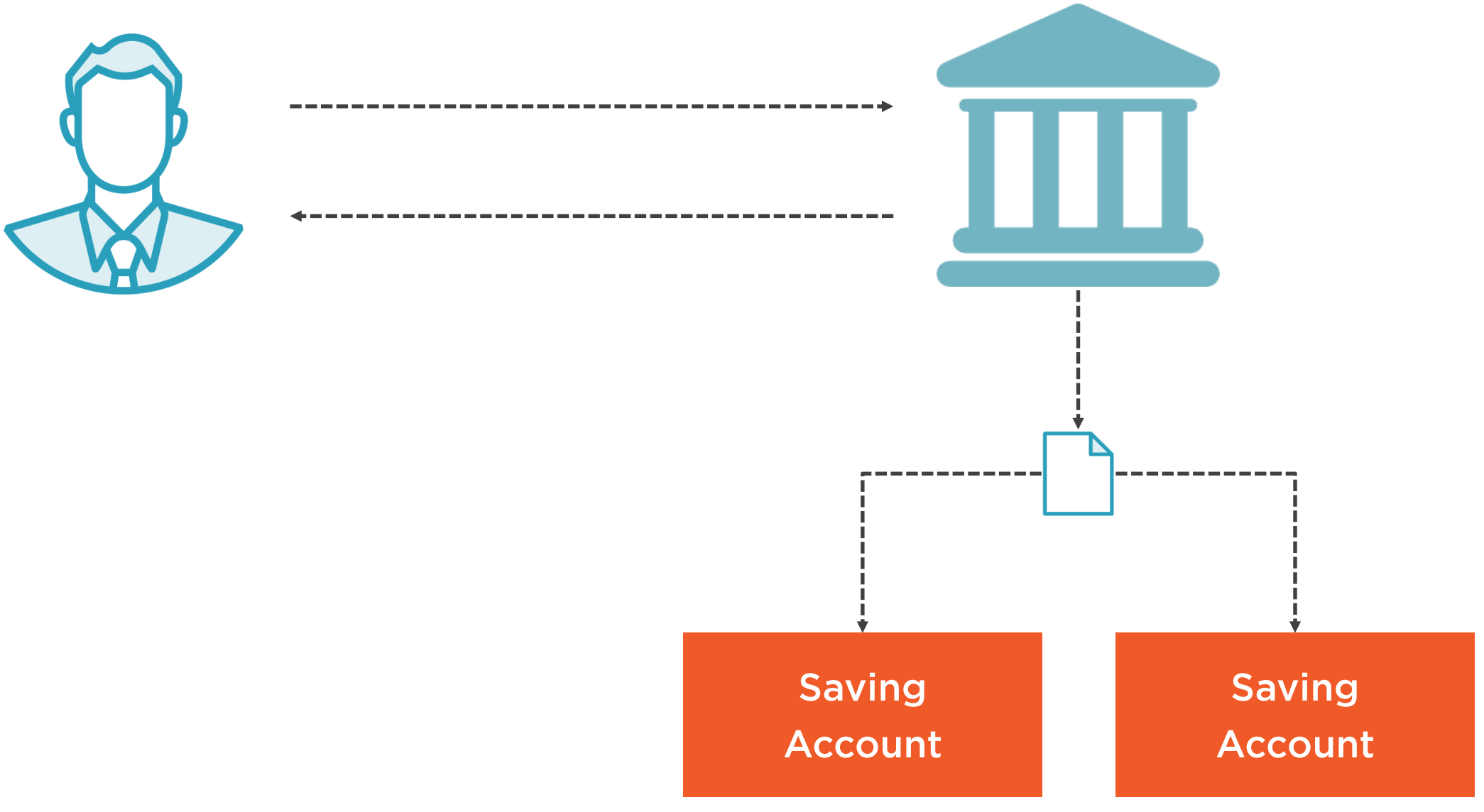
Demo

Demo: Spring Inversion of Control

Spring Dependency Injection

A technique whereby one object supplies the dependencies of another object.

Spring Dependency Injection



Points to Remember

One form of the broader technique of Inversion of Control

Allows the high-level code to receive the lower level code

Helps in loosely coupling

Types of Injections

Constructor Injection

Setter or Property Injection

Demo: Spring Dependency Injection using Constructor

Steps Required for Spring Dependency Injection Using Constructor

1. **Define Interface and Class**
2. **Create a Constructor for Injections**
3. **Configure the Dependency Injection in Spring Config file**

Demo

Spring Dependency Injection using Constructor

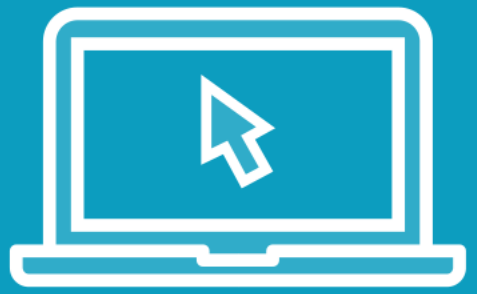
Demo: Spring Dependency Injection by Property

Development Process

Create Setter Methods for Injections

Configure the Dependency Injection in Spring Config file

Demo



Spring Dependency Injection by Property

Auto Wiring

Spring container can Autowire relationships between collaborating Beans

Advantages of Autowiring

Reduces the amount of code written in XML Configuration

Update Configuration as the objects of the application evolve.

```
<beans …>
   <bean id=“myCurrentAccount”
         class=“com.ps.springDemos.CurrentAccount”
         autowire = “mode”

   />
</beans>
```

Syntax: Autowiring

File: applicationContext.xml

Modes of Autowiring

no (Default)

- No Auto wiring

byName

- Injects the Object Dependency according to the name of the Bean

byType

- Injects the object dependency according to Type

constructor

- Similar to byType but this mode applies to constructor arguments

autodetect (Deprecated)

- Uses either of two modes i.e. constructor or byType modes

Demo: Auto Wiring

Limitations of Auto Wiring

Explicit dependencies in property and constructor-arg settings will always override Autowiring.

Cannot be applicable for simple properties such as primitives, Strings etc.

Spring Bean Scopes

Spring Bean Scopes

singleton:

- Only one Shared instance of the Bean will be created
- All Requests for the Bean will return a shared reference of the same Bean

prototype:

- Creates a new Bean instance for each container request

request:

- Scoped to an HTTP web request

session:

- Scoped to an HTTP web session

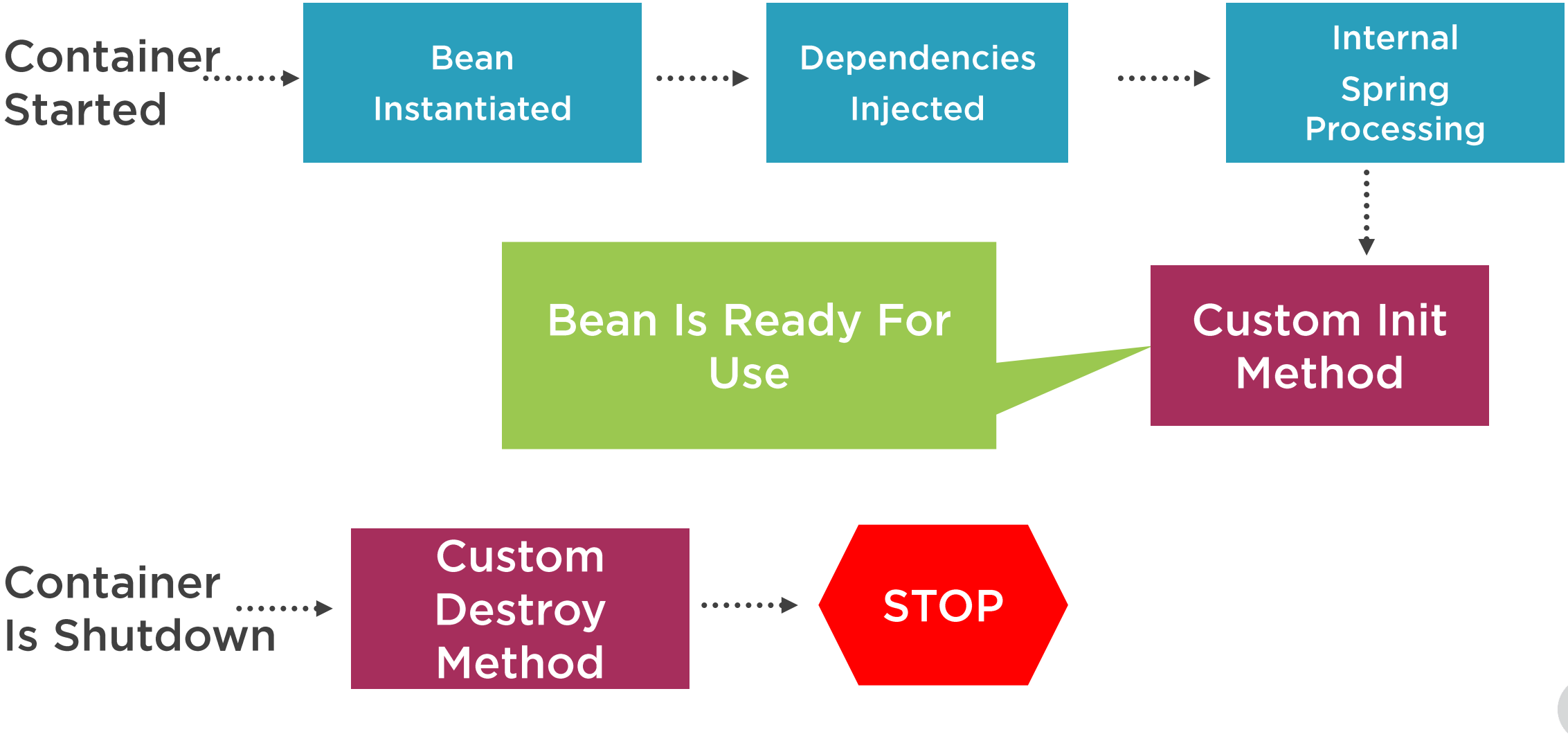
global-session:

- Scoped to a global HTTP web session

Demo: Spring Bean Scopes

Spring Bean Life Cycle

Bean Lifecycle



```
<beans …>

  <bean id="myCurrentAccount"
        class="com.ps.springDemos.CurrentAccount"
        init-method="methodNameofStartupStuff"
        destroy-method="methodNameofCleanupStuff"
  />
</beans>
```

Bean Life Cycle Methods Configuration

Demo: Spring Bean Life-Cycle

Development Process

Define your methods for init and destroy

Configure the method names in Spring Config file

Understanding Spring Core using Annotations

What are Java Annotations?

Special Labels/Markers added to Java Classes

Provide meta-data about the class

Processed at compile time or run-time

Why to Use Spring Configuration with Annotations?

Minimize the XML Configuration

- Requires less time
- Concise

Spring Core Annotations

@autowired

- Used to declare a constructor, field or setter method or configuration method

@configurable

- Used to declare types whose properties should be injected

@qualifier

- Used to guide autowiring to be performed by means other than byType

@required

- Used to specify that a particular property must be injected

@scope

- Used to specify the scope of a bean either singleton, prototype, request, session etc.

Demo: Understanding Spring Core using Annotations

Summary

- Understood the Spring Core Framework
- How to set up the development environment
- Inversion of Control
- Dependency Injection
- Auto Wiring
- Spring Bean Scopes
- Spring Bean Life Cycle
- How to use Annotations