# Spring-demo-one

## Interfaces:

EmployeeService.java
getMarketSaleStatus(); =>Method
Manager.java
getDailyTaskUpdate(); =>Method
getDailySale(); =>Method

## Classes:

EngineeringTeam.java
MarkettingTeam.java
TestSpringApp.java

### Configuration of Spring Container

The primary function of configuring spring container is
- To create and manage objects (Inversion of Control)
- To inject object's dependencies (dependency injection)

XML Configuration File: applicationContext.xml

### Steps 1. Configure Spring Beans

```
<beans>
<bean id="myTeam"
        class="com.branch.springwork.MarkettingTeam">
</bean>
</beans>
```

### Steps 2. Create a Spring Container

**TestSpringApp.java**

```
ClassPathXmlApplicationContext context =
new ClassPathXmlApplicationContext("applicationContext.xml");
```

## Steps 3. Retrieve Beans from Container

Manager manager = context.getBean("myTeam", Manager.class);

File: applicationContext.xml:

```
<bean id="myTeam"
        class="com.branch.springwork.MarkettingTeam">
```

## Injection Types: In this project I have used Constructor injection

## How Spring Framework process the config File:

In applicationContext.xml file, we have following lines of bean id code and constructor injection

```
<bean id="myService"
        class="com.branch.springwork.MyEmployeeService" />

<bean id="myTeam"
        class="com.branch.springwork.MarkettingTeam">

        <constructor-arg ref="myService" />
```

**With the reference of above bean id, Spring Framework creates behind the scene:**

EmployeeService myService = new MyEmployeeService()

Manager myteam = new MarkettingTeam(myService)