

# SPRING XML CONFIGURATION

GitHub: <https://github.com/HulyaMartli/SpringExercises3-XmlConfiguration>

## Spring Initializr

The screenshot shows the Spring Initializr web application interface. It has a dark theme with green accents. The top left features the 'spring initializr' logo. The main area is divided into several sections: 'Project' with radio buttons for 'Gradle - Groovy' (selected), 'Gradle - Kotlin', and 'Maven'; 'Language' with radio buttons for 'Java' (selected), 'Kotlin', and 'Groovy'; 'Spring Boot' with radio buttons for versions 3.2.0 (SNAPSHOT), 3.2.0 (M1), 3.1.3 (SNAPSHOT), 3.1.2, 3.0.10 (SNAPSHOT), 3.0.9, 2.7.15 (SNAPSHOT), and 2.7.14 (selected); 'Project Metadata' with input fields for Group (org.hulyam), Artifact (SpringExercises3-XmlConfiguration), Name (SpringExercises3-XmlConfiguration), Description (A project to examine Xml Configuration In Spring), and Package name (org.hulyam); 'Packaging' with radio buttons for 'Jar' (selected) and 'War'; and 'Language' with radio buttons for 'Java' (selected), '20', '17' (selected), '11', and '8'. On the right, there is a 'Dependencies' section with a button 'ADD DEPENDENCIES... CTRL + B'. Below this, there are two dependency cards: 'Lombok' with a 'DEVELOPER TOOLS' tag and description 'Java annotation library which helps to reduce boilerplate code.', and 'Spring Web' with a 'WEB' tag and description 'Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.'

## BEANS EXAMPLE

(gameConsoleXmlBeans package)

## Nesne Yaratılmasının Spring'e Devri

### Classes & Interfaces

```
public interface IRunnable {  
    void up();  
  
    void down();  
  
    void left();  
}
```

```
void right();  
}
```

```
/*  
 * up --> "Go up."  
 * down --> "Duck!"  
 * left --> "Go back."  
 * right --> "Fire!"  
 *  
 */  
public class Contra implements IRunnable {  
    @Override  
    public void up() {  
        System.out.println("Go up.");  
    }  
  
    @Override  
    public void down() {  
        System.out.println("Duck!");  
    }  
  
    @Override  
    public void left() {  
        System.out.println("Go back.");  
    }  
  
    @Override  
    public void right() {  
        System.out.println("Fire!");  
    }  
}
```

```
public class Mario implements IRunnable {  
  
    public void up() {  
        System.out.println("Jump!");  
    }  
  
    public void down() {  
        System.out.println("Enter the tube.");  
    }  
  
    public void left() {
```

```
        System.out.println("Go back.");
    }

    public void right() {
        System.out.println("Accelerate");
    }
}
```

```
/*
 * up --> "Go up."
 * down --> "Go down."
 * left --> "Go left."
 * right --> "Go right."
 *
 */

public class Pacman implements IRunnable {

    @Override
    public void up() {
        System.out.println("Go up.");
    }

    @Override
    public void down() {
        System.out.println("Go down.");
    }

    @Override
    public void left() {
        System.out.println("Go left.");
    }

    @Override
    public void right() {
        System.out.println("Go right.");
    }
}
```

```
public class GameRunner {

    IRunnable game;

    public GameRunner(IRunnable game) {
        this.game = game;
    }
}
```

```
public void run() {  
    System.out.println("Game is running ==> " +  
game.getClass().getSimpleName() + " - " + game.hashCode());  
    game.up();  
    game.down();  
    game.left();  
    game.right();  
}
```

```
public class GameConsoleRunner {  
    public static void main(String[] args) {  
  
        // to specify the XML configuration file:  
        ClassPathXmlApplicationContext context = new  
ClassPathXmlApplicationContext("gameConsoleApplicationContext.xml");  
  
        // instead of creating a new game object (new Mario, new  
Contra...)   
        // we passed this to Spring via XMLApplicationContext  
        //      GameRunner gameRunner = new  
GameRunner(context.getBean("mygame", IRunnable.class));  
  
        // Now, to also pass creating the new GameRunner object to  
Spring:  
        GameRunner gameRunner = context.getBean("gamerunner",  
GameRunner.class);  
        gameRunner.run();  
  
        // Spring creates objects with respect to SINGLETON  
        GameRunner gameRunner1 = context.getBean("gamerunner",  
GameRunner.class);  
        gameRunner1.run();  
    }  
}
```

## XML Config: Util Schema

resources packageında applicationContext.xml dosyasını oluşturduktan sonra örnek XML dosyasını aşağıdaki linkten aldık.

spring io - XML Schema-based configuration --> 40.2.2 the **util schema**

**ÖNEMLİ:** Burada Util Schema'yı sadece örnek bir xml taslağı olarak aldık. Dosya içindeki util özelliklerini hiç kullanmadık:

```
xmlns:util="http://www.springframework.org/schema/util"
    http://www.springframework.org/schema/util
http://www.springframework.org/schema/util/spring-util.xsd
```

*Yalnızca beanler oluşturduk, yani aşağıdaki özellikleri kullandık sadece:*

```
xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
```

*Ama sonraki CONTEXT SCHEMA EXAMPLE'da context özelliğini kullandık:*

```
<!-- The tag used for taking data from application.properties file -
->
    <context:property-placeholder location="application.properties">
</context:property-placeholder>
```

### **gameConsoleBeans.xml:**

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
    <!-- bean definitions here -->

</beans>
```

Spring'in yöneteceği beanleri bu XML dosyasında tanımlayacağız:

```
.  
.   
.   
    <!-- bean definitions here -->  
    <bean id="mygame" class="org.hulyam.gameConsoleXml.Pacman">  
</bean>  
    <bean id="gamerunner"  
class="org.hulyam.gameConsoleXml.GameRunner"></bean>  
</beans>
```

Main classta oluşturulan bu XML dosyasını tanıtmamız gerekiyor. Bunun için kullanılan kod parçasığı - ***ClassPathXMLApplicationContext***:

```
public class GameConsoleRunner {  
    public static void main(String[] args) {  
  
        // to specify the XML configuration file:  
        ClassPathXmlApplicationContext context = new  
ClassPathXmlApplicationContext("applicationContext.xml");  
  
        GameRunner gameRunner = new  
GameRunner(context.getBean("mygame", IRunnable.class));  
        gameRunner.run();  
    }  
}
```

Böylelikle yeni IRunnable nesnesinin (new Mario, new Contra...) oluşturulmasını XMLApplicationContext ile Spring'e devretmiş olduk.

*IoC Configuration ile @Bean anotasyonu kullanılarak konfigüre edilen Spring projelerinde ise beanlere ulaşmak için kullandığımız kod parçasığı **AnnotationConfigApplicationContext** idi.*

## Bean içinde constructor-arg tanımlama

GameRunner nesnesinin de Spring tarafından oluşturulması için ise GameRunner constructorunda IRunnable bir nesneyi parametre olarak aldığı için burada parametre içeren bir bean yazılmalı.

```

<!-- bean definitions here -->
    <bean id="mygame" class="org.hulyam.gameConsoleXml.Pacman">
</bean>
    <bean id="gamerunner"
class="org.hulyam.gameConsoleXml.GameRunner">
        <constructor-arg ref="mygame"></constructor-arg>
    </bean>

```

Artık Main'de herhangi bir newleme işlemi yapmamıza gerek kalmadan aşağıdaki şekilde beanleri kullanarak oyunu çalıştırabiliriz:

```

public class GameConsoleRunner {
    public static void main(String[] args) {

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

        GameRunner gameRunner = context.getBean("gamerunner",
GameRunner.class);

        gameRunner.run();
    }
}

```

***SPRING - SINGLETON:*** Spring objeleri Singleton prensibine göre oluşturur.

*run() metodunda hangi oyunun çağırıldığını ve bu objenin hangi obje olduğunu görebilmek için aşağıdaki şekilde simple name ve hash code printi aldık:*

```

public class GameRunner {
    IRunnable game;

    public GameRunner(IRunnable game) {
        this.game = game;
    }

    public void run() {
        System.out.println("Game is running ==> " +
game.getClass().getSimpleName() + " - " + game.hashCode());
        game.up();
        game.down();
        game.left();
        game.right();
    }
}

```

```
}
```

*NOT: Mario, Contra ve Pacman sınıfları IRunnable implement eder.*

*Daha sonra Main sınıfta iki adet GameRunner oluşturarak ikisinin de run() metotlarını çağırdığımızda çıktıda aynı oyun objesinin kullanıldığını görebiliriz:*

```
public class Main {  
    public static void main(String[] args) {  
  
        ClassPathXmlApplicationContext context = new  
ClassPathXmlApplicationContext("applicationContext.xml");  
  
        GameRunner gameRunner = context.getBean("gamerunner",  
GameRunner.class);  
        gameRunner.run();  
  
        // Spring creates objects with respect to SINGLETON  
        GameRunner gameRunner1 = context.getBean("gamerunner",  
GameRunner.class);  
        gameRunner1.run();  
    }  
}
```

*ÇIKTI:*

```
Game is running ==> Pacman - 1388278453  
Go up.  
Go down.  
Go left.  
Go right.  
Game is running ==> Pacman - 1388278453  
Go up.  
Go down.  
Go left.  
Go right.
```

*İki Pacman de aynı hash koda sahip aynı objedir.*



# CONTEXT SCHEMA: Property-Placeholder + APPLICATION.PROPERTIES EXAMPLE

(databaseXmlContext package)

In this example;

1- There should be logging methods in PostgreSQL and MySQL classes --> e.g. "Logged to PostgreSQL"

2- In the Database Service class, there should be a method called logToDatabase; in it, data should

be logged to whichever database is being used.

3- Write a databaseContext.xml configuration file to run this application and test it in DatabaseRunner

## Classes & Interfaces

```
@SuperBuilder
@Data
@NoArgsConstructor
@AllArgsConstructor
public abstract class Database {
    String username;
    String password;
    String url;
    public abstract void log();
}
```

```
public class MySql extends Database{
    @Override
    public void log() {
        System.out.println("Connected to "+url);
        System.out.println("Username: "+username);
        System.out.println("Logged to MySql.");
    }
}
```

```
public class PostgreSQL extends Database{
    @Override
    public void log() {
        System.out.println("Logged to PostgreSQL.");
        System.out.println("Connected to "+url);
        System.out.println("Username: "+username);
    }
}
```

```
public class DatabaseService {
    Database database;

    public DatabaseService(Database database){
        this.database=database;
    }

    public void logToDatabase(){
        database.log();
    }
}
```

```
public class DatabaseRunner {
    public static void main(String[] args) {
        ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("databaseApplicationContext.xml");
        DatabaseService databaseService =
context.getBean("databaseservice", DatabaseService.class);
        databaseService.logToDatabase();
    }
}
```

XML Config file

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:util="http://www.springframework.org/schema/util"
       xsi:schemaLocation="
           http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/util
http://www.springframework.org/schema/util/spring-util.xsd">
    <!-- bean definitions here -->
    <bean id="database" class="org.hulyam.databaseXml.MySql"></bean>
    <bean id="databaseservice"
class="org.hulyam.databaseXml.DatabaseService">
        <constructor-arg ref="database"></constructor-arg>
    </bean>
</beans>
```

## Fieldlara XML Config Dosyasında Elle Değer Atama

```
.
.
.
.
<!-- bean definitions here -->
    <bean id="database" class="org.hulyam.databaseXml.MySql">
        <property name="username" value="mysql"></property>
        <property name="url" value="jdbc:mysql..."></property>
    </bean>
```

DatabaseRunner çalıştırıldığında ÇIKTI:

```
Connected to jdbc:mysql...
Username: mysql
Logged to MySql.
```

## Fieldlara XML Config Dosyasında application.properties | application.yml ile Değer Atama

Application özelliklerini XML dosyasında elle değiştirmek yerine artık bütün uygulamanın ayarlarını kontrol edebileceğimiz application.properties veya application.yml dosyasını kullanabiliriz.

Böylelikle başka bir dosyada herhangi bir değişiklik yapmadan sadece bu ayar dosyasındaki değişikliklerle application değişikliklerini gerçekleştirebiliriz.

### **application.properties:**

```
database.url="jdbc:/postgresql..."
database.password="root"
database.username="postgres"
database.name="UsersDB"
```

## **Xml Config: Context Schema**

Artık bu dosyayı xml config dosyasında kullanabiliriz. Bunun için config dosyasına **context schema** kod parçasığını eklememiz gerekiyor:

[spring io - XML Schema-based configuration](#) --> 40.2.8 the context schema

```
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">
```

application.properties'deki verileri çekmek için aşağıdaki tagi kullanırız:

```
<context:property-placeholder location="application.properties">
</context:property-placeholder>
```

## **Değerlerin elle girilmiş ve application.properties'den alınmış halleri:**

```
<!-- Manually added -->
<bean id="database" class="org.hulyam.databaseXml.MySql">
    <property name="username" value="mysql"></property>
    <property name="url" value="jdbc:mysql..."></property>
    <property name="password" value="root"></property>
</bean>

<!-- Taken from application.properties -->
<bean id="database2" class="org.hulyam.databaseXml.MySql">
    <property name="username" value="${database.username}">
</property>
    <property name="url" value="${database.url}"></property>
    <property name="password" value="${database.password}">
</property>
</bean>
```

## databasePropertyPlaceholderContext.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="
           http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- The tag used for taking data from application.properties
file -->
    <context:property-placeholder location="application.properties">
</context:property-placeholder>

    <!-- Manually added -->
    <bean id="database" class="org.hulyam.databaseXml.MySql">
        <property name="username" value="mysql"></property>
        <property name="url" value="jdbc:mysql..."></property>
        <property name="password" value="root"></property>
    </bean>

    <!-- Taken from application.properties -->
    <bean id="database2" class="org.hulyam.databaseXml.PostgreSql">
        <property name="username" value="${database.username}">
</property>
        <property name="url" value="${database.url}"></property>
        <property name="password" value="${database.password}">
</property>
    </bean>

    <bean id="databaseservice"
class="org.hulyam.databaseXml.DatabaseService">
        <constructor-arg ref="database2"></constructor-arg>
    </bean>

</beans>
```

Bu sayede artık database değişikliği yapılacağı zaman sadece id=database2 olan beande class olarak PostgreSql yerine MySql seçilir ve application.properties dosyasına MySql için veriler girilir. Yani sınıflarda kodda hiçbir değişiklik yapmadan sadece XML config dosyası ve

application.properties dosyası üzerinde değişiklik yapılarak database değiştirilebiliyor. Bu da esneklik, daha kolay değişiklik yapma imkanı sağlıyor.

## CONTEXT SCHEMA: Component-Scan + ANNOTATION EXAMPLE

(gameConsoleAnnotation package)

### XML Config File

Anotasyonlar kullanarak Spring Framework'e devir işlemi gerçekleştireceğiz. Bu nedenle xml dosyasında "context:component-scan" tagini kullanacağız. component-scan ile bir adres vereceğimizi ve bu adresteki/packagedaki bütün componentlerin (Java'nın yönettiği yapıların) taranmasını istediğimizi belirtmiş oluyoruz.

*Spring anotasyonlarında @Service, @Repository, vs. gibi anotasyonların hepsi @Component anotasyonun alt kırılımıdır. Aslında hepsi birer componenttir.*

**gameConsoleAnnotationContext.xml:**

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="
         http://www.springframework.org/schema/beans
         http://www.springframework.org/schema/beans/spring-beans.xsd
         http://www.springframework.org/schema/context
         http://www.springframework.org/schema/context/spring-context.xsd">

  <context:component-scan base-
package="org.hulyam.gameConsoleAnnotation"></context:component-scan>

</beans>
```

Bu noktadan sonra xml dosyasına başka hiçbir şey yazmıyoruz. Artık anotasyonları kullanarak devir işlemine devam edeceğiz.

# Annotations

## @Component

### Contra Class + Mario Class + Pacman Class: @Component

```
@Component
public class Contra implements IRunnable {
    @Override
    public void up() {
        System.out.println("Go up.");
    }

    @Override
    public void down() {
        System.out.println("Duck!");
    }

    @Override
    public void left() {
        System.out.println("Go back.");
    }

    @Override
    public void right() {
        System.out.println("Fire!");
    }
}
```

### GameRunner Class: @Component

```
@Component
public class GameRunner {
    IRunnable game;

    public GameRunner(IRunnable game) {
        this.game = game;
    }

    public void run() {
        System.out.println("Game is running ==> " +
game.getClass().getSimpleName() + " - " + game.hashCode());
        game.up();
        game.down();
    }
}
```

```
        game.left();
        game.right();
    }
}
```

## Main - GameConsole Runner Class:

```
public class GameConsoleRunner {
    public static void main(String[] args) {

        // to specify the XML configuration file:
        ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("gameConsoleAnnotationContext.xml");

        GameRunner gameRunner = context.getBean(GameRunner.class);
        gameRunner.run();
    }
}
```

Eğer bu şekilde direkt olarak çalıştırırsak GameRunner IRunnable olarak çalıştığından ve IRunnable tipinde 3 farklı sınıfı da (Mario, Packman, Contra) @Component ile işaretlediğimizden aşağıdaki şekilde hata alırız:

## HATA:

```
Caused by:
org.springframework.beans.factory.NoUniqueBeanDefinitionException:
No qualifying bean of type
'org.hulyam.gameConsoleAnnotation.IRunnable' available: expected
single matching bean but found 3: contra,mario,pacman
```

## @Primary - Qualifying Beans with Annotations

Yukarıdaki hatanın önüne geçmek için @Primary anotasyonu kullanılır.

### Mario Class: @Component + @Primary

```
@Component
@Primary
public class Mario implements IRunnable {

    public void up() {
        System.out.println("Jump!");
    }
}
```



```
    public void down() {  
        System.out.println("Enter the tube.");  
    }  
  
    public void left() {  
        System.out.println("Go back.");  
    }  
  
    public void right() {  
        System.out.println("Accelerate");  
    }  
}
```

## @Autowired - Constructor Injection

### Constructor Injection

```
@Component  
public class GameRunner {  
    IRunnable game;  
  
    // Constructor Injection  
    public GameRunner(IRunnable game) {  
        this.game = game;  
    }  
  
    public void run() {  
        System.out.println("Game is running ==> " +  
game.getClass().getSimpleName() + " - " + game.hashCode());  
        game.up();  
        game.down();  
        game.left();  
        game.right();  
    }  
}
```

## @Autowired ile Injection

```
@Component
public class GameRunner {
    @Autowired
    IRunnable game;

    public void run() {
        System.out.println("Game is running ==> " +
game.getClass().getSimpleName() + " - " + game.hashCode());
        game.up();
        game.down();
        game.left();
        game.right();
    }
}
```

## WORKING WITH QUALIFIERS EXAMPLE

(gameConsoleQualifiers package)

**gameConsoleQualifiersContext.xml:**

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

    <context:component-scan base-
package="org.hulyam.gameConsoleQualifiers"></context:component-scan>
</beans>
```

**Main:**

```

public class GameConsoleRunner {
    public static void main(String[] args) {

        // to specify the XML configuration file:
        ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("gameConsoleQualifiersContext.xml");

        GameRunner gameRunner = context.getBean(GameRunner.class);
        gameRunner.run();
    }
}

```

Artık @Primary yerine @Qualifier kullanarak beanleri nitelendireceğiz:

### Contra:

```

@Component
@Qualifier("contra")
public class Contra implements Runnable {

    @Override
    public void up() {
        System.out.println("Go up.");
    }

    @Override
    public void down() {
        System.out.println("Duck!");
    }

    @Override
    public void left() {
        System.out.println("Go back.");
    }

    @Override
    public void right() {
        System.out.println("Fire!");
    }
}

```

### Mario:

```

@Component
@Qualifier("mario")
public class Mario implements Runnable {

```

```

    public void up() {
        System.out.println("Jump!");
    }

    public void down() {
        System.out.println("Enter the tube.");
    }

    public void left() {
        System.out.println("Go back.");
    }

    public void right() {
        System.out.println("Accelerate");
    }
}

```

### **Pacman:**

```

@Component
@Qualifier("pacman")
public class Pacman implements IRunnable {
    @Override
    public void up() {
        System.out.println("Go up.");
    }

    @Override
    public void down() {
        System.out.println("Go down.");
    }

    @Override
    public void left() {
        System.out.println("Go left.");
    }

    @Override
    public void right() {
        System.out.println("Go right.");
    }
}

```

Her `IRunnable` cinsinden obje için bir qualifier tanımladıktan sonra `GameRunner` sınıfında `@Autowired` ile Constructor Injection yaptığımız yerde hangi beanin kullanılacağını yine `@Qualifier` anotasyonu ile belirtiyoruz:

### **GameRunner with @Autowired:**

```
@Component
public class GameRunner {
    @Autowired
    @Qualifier("contra")
    IRunnable game;

    // Constructor Injection
    /*    public GameRunner(IRunnable game) {
        this.game = game;
    }*/

    public void run() {
        System.out.println("Game is running ==> " +
game.getClass().getSimpleName() + " - " + game.hashCode());
        game.up();
        game.down();
        game.left();
        game.right();
    }
}
```

### **GameRunner with Constructor:**

```
@Component
public class GameRunner {
    IRunnable game;

    // Constructor Injection
    public GameRunner(@Qualifier("contra") IRunnable game) {
        this.game = game;
    }

    public void run() {
        System.out.println("Game is running ==> " +
game.getClass().getSimpleName() + " - " + game.hashCode());
        game.up();
    }
}
```

```
        game.down();
        game.left();
        game.right();
    }
}
```

## IoC CONFIGURATION WITH BEANS EXAMPLE

(gameConsoleIoC)

Artık bağlantılarımızı XML üzerinden yapmıyoruz, bir Configuration sınıfı kullanacağız. ***"AnnotationConfigApplicationContext"*** ile configurationı yaptığımız sınıfa ulaşacağız.

IoC Configuration = With a Config class, @Bean and @Configuration only

Main:

```
public class GameConsoleRunner {
    public static void main(String[] args) {

        // to specify the configuration file:
        AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(IocConfig.class);

        GameRunner gameRunner =
context.getBean("gameRunner",GameRunner.class);
        gameRunner.run();
    }
}
```

IocConfig:

```
@Configuration
public class IocConfig {

    @Bean
    @Primary
    public IRunnable getMario(){
        return new Mario();
    }
}
```

```

@Bean
public IRunnable getContra(){
    return new Contra();
}

@Bean
public IRunnable getPacman(){
    return new Pacman();
}

@Bean
public GameRunner gameRunner(){
    return new GameRunner(getPacman());
}
}

```

## GameRunner

```

public class GameRunner {
    IRunnable game;

    // Constructor Injection
    public GameRunner(IRunnable game) {
        this.game = game;
    }

    public void run() {
        System.out.println("Game is running ==> " +
game.getClass().getSimpleName() + " - " + game.hashCode());
        game.up();
        game.down();
        game.left();
        game.right();
    }
}

```

Mario (Contra ve Pacman de aynı bu şekilde hiçbir anotasyon barındırmıyor):

```

public class Mario implements IRunnable {

    public void up() {
        System.out.println("Jump!");
    }
}

```

```
public void down() {  
    System.out.println("Enter the tube.");  
}  
  
public void left() {  
    System.out.println("Go back.");  
}  
  
public void right() {  
    System.out.println("Accelerate");  
}  
}
```

ÇIKTI: Mario beanini IocConfig dosyasında @Primary ile işaretli ancak GameRunner beaninde Pacmanı method calling yöntemi ile çağırdığımız için

```
Game is running ==> Pacman - 1442191055  
Go up.  
Go down.  
Go left.  
Go right.
```

## IOC CONFIG WITH ANNOTATIONS EXAMPLE

**@ComponentScan** config dosyasına eklenir. Burada component scan yapılacak package yolu belirtilir.

***Ve artık GameRunner için bir bean olmadığından onu da @Component anotasyonu ile işaretliyoruz.***

IocConfig:

```
@Configuration  
@ComponentScan("org.hulyam.gameConsoleIoAnnotation")  
public class IocConfig {  
}
```

Mario (Contra ve Pacman'e de @Component anotasyonu eklendi):

```
@Component  
@Primary  
public class Mario implements IRunnable {
```



```
public void up() {  
    System.out.println("Jump!");  
}  
  
public void down() {  
    System.out.println("Enter the tube.");  
}  
  
public void left() {  
    System.out.println("Go back.");  
}  
  
public void right() {  
    System.out.println("Accelerate");  
}  
}
```

ÇIKTI: Mario sınıfı @Component ile beraber @Primary ile işaretlendiği için

```
Game is running ==> Mario - 552936351  
Jump!  
Enter the tube.  
Go back.  
Accelerate
```

## Without Config File (@Configuration) & @ComponentScan in Main (NOT IOC CONFIG)

Just to understand the transition to Spring Boot. **@SpringBootApplication annotation includes @ComponentScan, @Configuration, etc.**

Main:

```
@ComponentScan
public class GameConsoleRunner {
    public static void main(String[] args) {

        // to specify the configuration file:
        AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(GameConsoleRunner.class);

        GameRunner gameRunner =
context.getBean("gameRunner",GameRunner.class);
        gameRunner.run();
    }
}
```

!!! AnnotationConfigApplicationContext artık main classın adı ile ayarlandı

```
AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(GameConsoleRunner.class);
```

ÇIKTI: Mario sınıfı @Component ile beraber @Primary ile işaretlendiği için

```
Game is running ==> Mario - 825936265
Jump!
Enter the tube.
Go back.
Accelerate
```