**5. Spring Dependency Injection**

**Dependency Injection**:

In software engineering, dependency injection is a technique whereby one object (or static method) supplies the dependencies of another object. A dependency is an object that can be used (a service). An injection is the passing of a dependency to a dependent object (a client) that would use it. The service is made part of the client's state. Passing the service to the client, rather than allowing a client to build or find the service, is the fundamental requirement of the pattern.

Dependency injection is a pattern through which to implement IoC, where the control being inverted is the setting of object’s dependencies.

The act of connecting objects with other objects, or “injecting” objects into other objects, is done by an assembler rather than by the objects themselves.

**Injection Types**:

There are many types of injection with Spring. Here we will cover the two most common.

1. Constructor Injection
2. Setter Injection

**Constructor Injection**:

Constructor injection injects dependency by calling constructor on our class.

**Development process of constructor injection**:

1. Define the dependency interface and class
2. Create a constructor in our class for injection
3. Configure the dependency injection in Spring configuration file.

**a) Define the dependency interface and class**:

**File: FortuneService.java**:

**package** com.odduu.ruhul;

**public** **interface** FortuneService {

**public** String getFortune();

}

**File: HappyFortuneService.java**:

**package** com.odduu.ruhul;

**public** **class** HappyFortuneService **implements** FortuneService {

@Override

**public** String getFortune() {

**return** "Today is your lucky day!";

}

}

**b)Create a constructor in our class for injection**:

**File: BaseballCoach.java**:

**package** com.odduu.ruhul;

**public** **class** BaseballCoach **implements** Coach {

//define a private field for the dependency

**private** FortuneService fortuneService;

//define a constructor for dependency injection

**public** BaseballCoach(FortuneService fortuneService) {

**this**.fortuneService = fortuneService;

}

@Override

**public** String getDailyWorkout() {

**return** "Spend 30 minutes on batting practice";

}

@Override

**public** String getDailyFortune() {

//use my fortuneService to get a fortune

**return** fortuneService.getFortune();

}

}

**c) Configure the dependency injection in Spring configuration file**:

**File: applicationContext.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"*>

<!-- Define your beans here -->

<!-- define the dependency -->

<bean id=*"myFortune"*

class=*"com.odduu.ruhul.HappyFortuneService"*>

</bean>

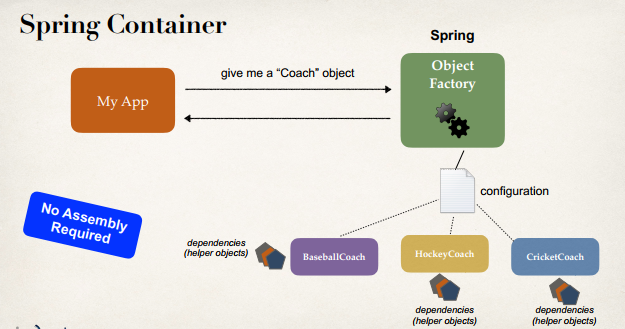
<bean id=*"myCoach"*

class=*"com.odduu.ruhul.BaseballCoach"*>

<constructor-arg ref=*"myFortune"*/>

</bean>

</beans>



**Constructor injection behind the scenes**:



**Setter Injection**:

Setter injection injects dependency by calling setter method(s) on our class.

**Development process of setter injection**:

1. Create setter method(s) in our class for injection
2. Configure the dependency injection in Spring configuration file.

**a) Create setter method(s) in our class for injection**:

**File: CricketCoach.java**:

**package** com.odduu.ruhul;

**public** **class** CricketCoach **implements** Coach {

**private** FortuneService fortuneService;

//create a no-arg constructor

**public** CricketCoach() {

System.***out***.println("CricketCoach: inside no-arg constructor");

}

//create setter method for injection

**public** **void** setFortuneService(FortuneService fortuneService) {

System.***out***.println("CricketCoach: inside setter method - setFortuneService");

**this**.fortuneService = fortuneService;

}

@Override

**public** String getDailyWorkout() {

**return** "Practice fast bowling for 15 minutes";

}

@Override

**public** String getDailyFortune() {

**return** fortuneService.getFortune();

}

}

**b) Configure the dependency injection in Spring configuration file**:

**File: applicationContext.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"*>

<!-- Define your beans here -->

<!-- define the dependency -->

<bean id=*"myFortune"*

class=*"com.odduu.ruhul.HappyFortuneService"*>

</bean>

<bean id=*"myCoach"*

class=*"com.odduu.ruhul.BaseballCoach"*>

<!-- set up constructor injection -->

<constructor-arg ref=*"myFortune"*/>

</bean>

<bean id=*"myCricketCoach"*

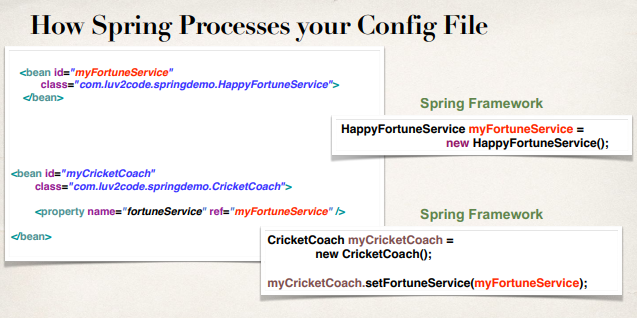
class=*"com.odduu.ruhul.CricketCoach"*>

<!-- set up our setter injection -->

<property name=*"fortuneService"* ref=*"myFortune"*/>

</bean>

</beans>



**Injecting Literal Values**:

Now we inject literal values into our Spring objects.

**Development Process**:

1. Create setter method(s) in your class for injections
2. Configure the injection in Spring configuration file

**Create setter method(s) in your class for injections**:

**File: CricketCoach.java**:

**package** com.odduu.ruhul;

**public** **class** CricketCoach **implements** Coach {

**private** FortuneService fortuneService;

// add new fields for emailAddress and team

**private** String emailAddress;

**private** String team;

// create a no-arg constructor

**public** CricketCoach() {

System.***out***.println("CricketCoach: inside no-arg constructor");

}

**public** String getEmailAddress() {

**return** emailAddress;

}

**public** **void** setEmailAddress(String emailAddress) {

System.***out***.println("CricketCoach: inside setter method - setEmailAddress");

**this**.emailAddress = emailAddress;

}

**public** String getTeam() {

**return** team;

}

**public** **void** setTeam(String team) {

System.***out***.println("CricketCoach: inside setter method - setTeam");

**this**.team = team;

}

// create setter method for injection

**public** **void** setFortuneService(FortuneService fortuneService) {

System.***out***.println("CricketCoach: inside setter method - setFortuneService");

**this**.fortuneService = fortuneService;

}

@Override

**public** String getDailyWorkout() {

**return** "Practice fast bowling for 15 minutes";

}

@Override

**public** String getDailyFortune() {

**return** fortuneService.getFortune();

}

}

**Configure the injection in Spring configuration file**:

**File: applicationContext.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"*>

<!-- Define your beans here -->

<!-- define the dependency -->

<bean id=*"myFortune"* class=*"com.odduu.ruhul.HappyFortuneService"*>

</bean>

<bean id=*"myCoach"* class=*"com.odduu.ruhul.BaseballCoach"*>

<!-- set up constructor injection -->

<constructor-arg ref=*"myFortune"* />

</bean>

<bean id=*"myCricketCoach"* class=*"com.odduu.ruhul.CricketCoach"*>

<!-- set up our setter injection -->

<property name=*"fortuneService"* ref=*"myFortune"* />

<!-- inject literal value -->

<property name=*"emailAddress"* value=*"ruhulmath08@gmail.com"* />

<property name=*"team"* value=*"Dhaka Tigers"* />

</bean>

</beans>

**Note**:

The name of the property file must be the same with the class variable name. For example "emailAddress" and ="team"

Now call this value from SetterDemoApp.

**File: SetterDemoApp.java**:

**package** com.odduu.ruhul;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** SetterDemoApp {

**public** **static** **void** main(String[] args) {

// load the spring configuration file

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

// retrieve bean from spring container

CricketCoach theCoach = context.getBean("myCricketCoach", CricketCoach.**class**);

// call method on the bean

System.***out***.println(theCoach.getDailyWorkout());

System.***out***.println(theCoach.getDailyFortune());

// call our methods to get literal values

System.***out***.println(theCoach.getEmailAddress());

System.***out***.println(theCoach.getTeam());

// close the context

context.close();

}

}

/\*

Output:

CricketCoach: inside no-arg constructor

CricketCoach: inside setter method - setFortuneService

CricketCoach: inside setter method - setEmailAddress

CricketCoach: inside setter method - setTeam

Practice fast bowling for 15 minutes

Today is your lucky day!

ruhulmath08@gmail.com

Dhaka Tigers

\*/

**Injecting Values from a Properties File**:

Now we inject literal values into our Spring objects from property file. In the case of "Inject Literal Value" we insert hardcoded value. But here we insert value from a file.

**Development Process**:

1. Create Properties File
2. Load Properties File in Spring configuration file
3. Reference values from Properties File

**Create Properties File**:

Go to project src -> Right Click -> New -> File -> file\_name.properties

**File:sport. Properties**:

foo.email=myEmail@gmail.com

foo.team=Rajshahi Riders

**Load Properties File in Spring configuration file**:

<!-- load the properties file: sport.properties -->

<context:property-placeholder

location=*"classpath:sport.properties"* />

**Reference values from Properties File**:

<bean id=*"myCricketCoach"* class=*"com.odduu.ruhul.CricketCoach"*>

<!-- set up our setter injection -->

<property name=*"fortuneService"* ref=*"myFortune"* />

<!-- inject literal values from property file -->

<property name=*"emailAddress"* value=*"${foo.email}"* />

<property name=*"team"* value=*"${foo.team}"* />

</bean>

**File: applicationContext.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"*>

<!-- load the properties file: sport.properties -->

<context:property-placeholder

location=*"classpath:sport.properties"* />

<!-- Define your beans here -->

<!-- define the dependency -->

<bean id=*"myFortune"* class=*"com.odduu.ruhul.HappyFortuneService"*>

</bean>

<bean id=*"myCoach"* class=*"com.odduu.ruhul.BaseballCoach"*>

<!-- set up constructor injection -->

<constructor-arg ref=*"myFortune"* />

</bean>

<bean id=*"myCricketCoach"* class=*"com.odduu.ruhul.CricketCoach"*>

<!-- set up our setter injection -->

<property name=*"fortuneService"* ref=*"myFortune"* />

<!-- inject literal values from property file -->

<property name=*"emailAddress"* value=*"${foo.email}"* />

<property name=*"team"* value=*"${foo.team}"* />

</bean>

</beans>

5. Spring Dependency Injection