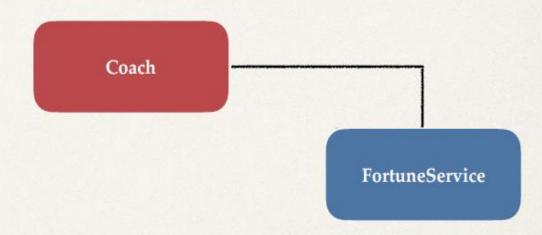
Spring Dependency Injection with Annotations and Autowiring



Demo Example

Our Coach already provides daily workouts

- Now will also provide daily fortunes
 - New helper: FortuneService
 - This is a dependency



FortuneService

For dependency injection, Spring can use auto wiring

- Spring will look for a class that *matches* the property
 - matches by type: class or interface

Spring will inject it automatically ... hence it is autowired

Injecting FortuneService into a Coach implementation

Spring will scan @Components

Any one implements FortuneService interface???

• If so, let's inject them. For example: HappyFortuneService

Autowiring Injection Types

Constructor Injection

Setter Injection

Field Injections

Development Process - Constructor Injection

1. Define the dependency interface and class



2. Create a constructor in your class for injections

3. Configure the dependency injection with @Autowired Annotation

Step 1: Define the dependency interface and class

```
File: FortuneService.java
```

```
public interface FortuneService {
  public String getFortune();
}
```

File: HappyFortuneService.java

```
@Component
public class HappyFortuneService implements FortuneService {
   public String getFortune() {
     return "Today is your lucky day!";
   }
}
```

Step 2: Create a constructor in your class for injections

Coach File: TennisCoach.java **FortuneService** @Component public class TennisCoach implements Coach { private FortuneService fortuneService; public TennisCoach(FortuneService theFortuneService) { fortuneService = theFortuneService;

Step 2: Create a constructor in your class for injections

Coach File: TennisCoach.java **FortuneService** @Component public class TennisCoach implements Coach { private FortuneService fortuneService; public TennisCoach(FortuneService theFortuneService) { fortuneService = theFortuneService;

Step 3: Configure the dependency injection @Autowired annotation

Coach File: TennisCoach.java @Component **FortuneService** public class TennisCoach implements Coach { private FortuneService fortuneService; @Autowired public TennisCoach(FortuneService theFortuneService) { fortuneService = theFortuneService;

Spring Container Spring Object give me a "Coach" object **Factory** My App configuration No Assembly Required dependencies (helper objects) HockeyCoach BaseballCoach CricketCoach dependencies dependencies

(helper objects)

(helper objects)

Setter Injection with Annotations and Autowiring



Spring Injection Types

Constructor Injection

Setter Injection

Field Injection

Inject dependencies by calling setter method(s) on your class

Injecting FortuneService into a Coach implementation

Spring will scan @Components

Any one implements FortuneService interface???

• If so, let's inject them. For example: HappyFortuneService

Development Process - Setter Injection

1. Create setter method(s) in your class for injections



2. Configure the dependency injection with @Autowired Annotation

Setter Injection with Annotations and Autowiring



Spring Injection Types

Constructor Injection

Setter Injection

Field Injection

Inject dependencies by calling setter method(s) on your class

Autowiring Example

FortuneService

Injecting FortuneService into a Coach implementation

Spring will scan @Components

Any one implements FortuneService interface???

• If so, let's inject them. For example: HappyFortuneService

Development Process - Setter Injection

1. Create setter method(s) in your class for injections



2. Configure the dependency injection with @Autowired Annotation

Step1: Create setter method(s) in your class for injections

```
@Component
public class TennisCoach implements Coach {
 private FortuneService fortuneService;
 public TennisCoach() {
 public void setFortuneService(FortuneService fortuneService) {
   this.fortuneService = fortuneService;
```

Step 2: Configure the dependency injection with Autowired Annotation

```
@Component
public class TennisCoach implements Coach {
 private FortuneService fortuneService;
 public TennisCoach() {
 @Autowired
 public void setFortuneService(FortuneService fortuneService) {
   this.fortuneService = fortuneService;
```

Step 2: Configure the dependency injection with Autowired Annotation

```
@Component
public class TennisCoach implements Coach {
 private FortuneService fortuneService;
 public TennisCoach() {
 @Autowired
 public void setFortuneService(FortuneService fortuneService) {
   this.fortuneService = fortuneService;
```

Inject dependencies by calling

ANY method on your class

Simply give: @Autowired

Step 2: Configure the dependency injection with Autowired Annotation

```
@Component
public class TennisCoach implements Coach {
 private FortuneService fortuneService;
 public TennisCoach() {
 @Autowired
 public void doSomeCrazyStuff(FortuneService fortuneService) {
   this.fortuneService = fortuneService;
```

Field Injection with Annotations and Autowiring



Spring Injection Types

Constructor Injection

Setter Injection

Field Injection

Inject dependencies by setting field values on your class directly

(even private fields)

Accomplished by using Java Reflection

Development Process - Field Injection



- 1. Configure the dependency injection with Autowired Annotation
 - Applied directly to the field
 - No need for setter methods

Step 1: Configure the dependency injection with Autowired Annotation

```
public class TennisCoach implements Coach {
 @Autowired
 private FortuneService fortuneService;
 public TennisCoach() {
 // no need for setter methods
```

Spring Injection Types

Constructor Injection

Setter Injection

Field Injection

Spring Injection Types

Constructor Injection

Setter Injection

Choose a style

Stay consistent in your project

Field Injection

Annotation Autowiring and Qualifiers



FortuneService

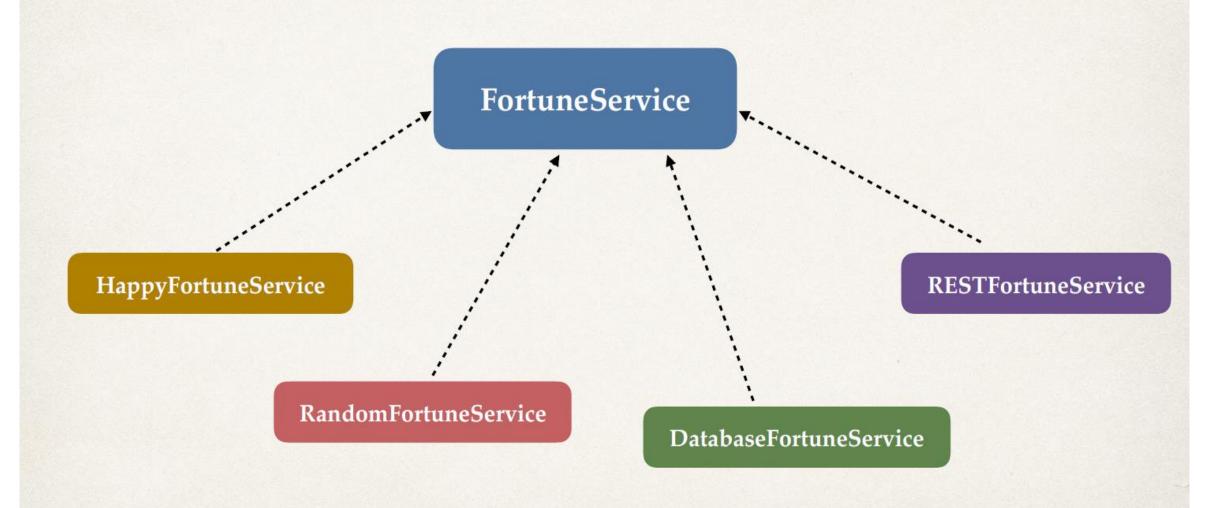
Injecting FortuneService into a Coach implementation

Spring will scan @Components

Any one implements FortuneService interface???

• If so, let's inject them ... oops which one?

Multiple FortuneService Implementations



Umm, we have a little problem

```
b) TemmisCoach: inside default constructor
Hav. 83. 2819 5:47:53 AM. org.snring@ramework.context.support.AbstractAnplicationContext refresh
WARLING: Exception encountered during contest in tiglization conselling refresh attempt, organize from some factory. Under infriedling contest in tiglization conselling refresh attempt, organize from the factory. Under infriedling contest in tiglization, conselling refresh attempt, organize from the factory. Under infriedling contest in tiglization, conselling refresh attempt, organized in the factory of the factory.
Exception in thread "main" one springframework beans factory. Unset: sfreeDependency Exception: Error creating bean with name 'tennisCoach': Unset: sfree devendency expression.
              at org. springframework, bases. Factory, smnotstion. Automired/AnnotationDeanPostProcessorS/utomired/lethod Deanert, inject (Automired/AnnotationDeanPostProcessor, java: /
             at ore, springframework, beans, factory, annotation, ImjectionMetadata, injectionMetadata, injectionMetadata,
              at neg springframework hoors fortney amontation AuthorizationResearch postProcessor postProcessor (available)
              at org. spring!rarework.beans.lactory.support.AbstractAutowireCapsbleBeanFactory.populateBean(AbstractAutowireCapableBeanFactory.java:1411)
              at org. springframework, heaps, factory, support. AbstractAurowine apshies earEactory, dolineateBean (AbstractAutowine Capable BeanFactory, java: 592)
              all ong. pringframework, beans, for Lory, support, Abstrac, Autewire Capable Scanfactory, ere ale Scan (AbstractAutowire Capable Scanfactory, jave: 515)
              at org. springframework, beens, tactory, support, AbstractSeenFactory, lerb dafdeGatBean56 (AbstractBeanFactory, neve: 326)
              at org. springframework, beand, factory, support, DefaultSingleton)earRegistry, getSingleton(DefaultSingletonDeanRegistry, jave: 222)
              artions, spring(narework, heans, factory, support, obstracthean-actory, done mean(obstracthean-actory, java; hus)
              at any paring from much has not for they support Abetwee Kears are great (abstract Norman and paring)
              at org. springframework, beens, factory, support, DefaultlistebleDeam'actory, prefixentiateSingletons (DefaultlistebleDeam'actory java:646)
              at org. springfrarework.cortext.support.AbstractApplicationContext.firishDeanFactoryInitialization(AbstractApplicationContext.fava:877)
              at any, the ingle account, context, support. AbstractApplicationContext, sofrecing AbstractApplicationContext, java: 545)
              ut ore, terring runnework, cortext, tupport, C. as sPathXml@policationCortext, AirLt*(ClassPathAmplicationContext, java; 144)
              at org. springframework.context.support.C.assPathXmlApplicationContext.cinit*(ClassPathXmLApplicationContext.fave:85)
              at com.springdemc.AnnotationDemoApp.main(AnnotationDemoApp.java:11)
Caused by: one spring(ramework, beans, factory, NoDmigueReanDefinitionEscention: No qualifying bean of type "com, springcemo, FortuneService" available: expected single m
              attions, springfrancework, beans, Factory, config. Bependency Descriptor, resolve Not Unique (Depondency Descriptor, java, 221)
              at org. springframework, beans, taktory, support, DetaultListableEsanfactory, doResolveDependency(DetaultListableEsanfactory, maya 1225)
              at org. pringframework, beine . Factory . capport. Defaulttic . abia@aariae.com. macolocDapondersor (Dafaulttic tabla@aariae.tabla@aariae.com.) acceptation
              at ong. springtramework, beans tactory, annotation, automiredannotationse annotationse annotation descriptions annotation description and a springtramework, beans tactory, annotation descriptions annotation and a springtramework, beans tactory annotation description and a springtramework, beans tactory annotation and a springtramework, beans tactory annotation and a springtramework, beans tactory annotation and a springtramework and a s
              15 more
```

Solution: Be specific! - @Qualifier

```
@Component
public class TennisCoach implements Coach {
 @Autowired
 @Qualifier("happyFortuneService")
 private FortuneService fortuneService;
```

Solution: Be specific! - @Qualifier

```
@Component
public class TennisCoach implements Coach {
 @Autowired
 @Qualifier("happyFortuneService")
 private FortuneService fortuneService;
```

Injection Types

- Can apply @Qualifier annotation to
 - Constructor injection
 - Setter injection methods
 - Field injection