1- Defining injection point by using @Autowire annotation

The annotation @Autowired can be used to declare an injection point.

In other words, this annotation instructs the Spring container to find a registered bean of the same type as of the annotated type and perform dependency injection.

Example

@Autowired can be used at various places. Following example shows how to use it on a field.

```
public class GreetingService {

  public String getGreeting(String name) {

      return "Hi there, " + name;

  }

}
```

Using @Autowired

```
public class Greeter {

  @Autowired

  private GreetingService greetingService;


  public void showGreeting(String name) {

      System.out.println(greetingService.getGreeting(name));

  }

}
```

Defining beans and running the example application

```
@Configuration
public class AppRunner {

  @Bean
  public GreetingService greetingService() {
     return new GreetingService();
  }

  @Bean
  public Greeter greeter() {
     return new Greeter();
  }

  public static void main(String... strings) {
     AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(AppRunner.class);
     Greeter greeter = context.getBean(Greeter.class);
     greeter.showGreeting("Piseth");
  }
}
```

Output: Hi there, Piseth

@Autowire can also be used on setters , constructors and to any methods having multiple arguments.

2 - Defining Injection point by using @Inject annotation

JSR 330's javax.inject.@Inject annotation can be used in place of Spring's @Autowired annotation.

Starting with Spring 3.0, Spring offers support for JSR 330 standard annotations (Dependency Injection). Those annotations are scanned in the same way as the Spring annotations.

Example

```
public class GreetingService {
  public String getGreeting(String name) {
      return "Hi there, " + name;
  }
}
```

Using @Inject annotation

```
public class Greeter {
  @Inject
  private GreetingService greetingService;

  public void showGreeting(String name){
      System.out.println(greetingService.getGreeting(name));
  }
}
```

Defining beans and running the example app

```
@Configuration
public class AppRunner {

  @Bean
  public GreetingService greetingService() {
      return new GreetingService();
  }

  @Bean
  public Greeter greeter() {
      return new Greeter();
  }

  public static void main(String... strings) {
      AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(AppRunner.class);
      Greeter greeter = context.getBean(Greeter.class);
      greeter.showGreeting("Piseth");
  }
}
```

Output: Hi there, Piseth

3 - Using @Autowired annotation on arbitrary methods

@Autowired annotation can be used on the methods with arbitrary names and multiple arguments:

```
  @Autowired
  public void configure(GreetingService greetingService, LocalDateTime appStartTime) {
    ....
  }
```

Example

```
public class GreetingService {

  public String getGreeting(String name) {
      return "Hi there, " + name;
  }
}
```

Using @Autowired at arbitrary methods

```
public class Greeter {
  private String greetingFormat;

  @Autowired
  public void configure(GreetingService greetingService, LocalDateTime appServiceTime) {
      greetingFormat = String.format("%s. This app is running since: %s%n",
greetingService.getGreeting("<NAME>"),
              appServiceTime.format(DateTimeFormatter.ofPattern("YYYY-MMM-d")));
  }

  public void showGreeting(String name) {
      System.out.printf(greetingFormat.replaceAll("<NAME>", name));
  }
}
```

Defining beans and running the example

```
@Configuration
public class AppRunner {

  @Bean
  public GreetingService greetingService() {
      return new GreetingService();
  }

  @Bean
  public LocalDateTime appServiceTime() {
      return LocalDate.of(2022, 7, 17).atStartOfDay();
  }

  @Bean
  public Greeter greeter() {
      return new Greeter();
  }


  public static void main(String... strings) {
      AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(AppRunner.class);
      Greeter greeter = context.getBean(Greeter.class);
      greeter.showGreeting("Piseth");
  }
}
```

Output

```
Hi there, Piseth. This app is running since: 2022-July-17
```