Spring framework

Trương Phước Lộc

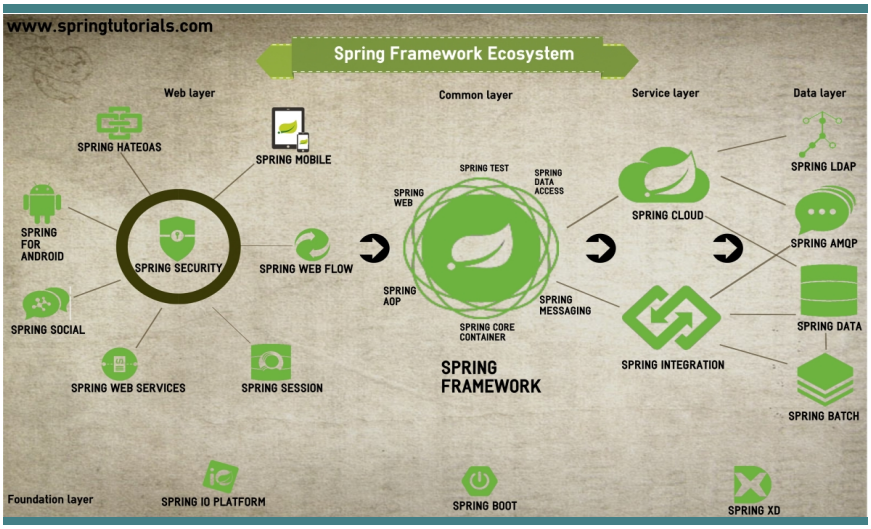
1. Giới thiệu Spring
2. Giới thiệu Dependency Injection
3. Ví dụ Bean Configuration
4. Wiring bean – XML
5. Wiring bean - Annotation

1. Spring projects

- Spring IO Platform
- Spring Boot
- Spring framework
- Spring data
- Spring integration
- Srping batch
- Spring security
- Spring Web services
- Spring mobile
- …

https://spring.io/projects

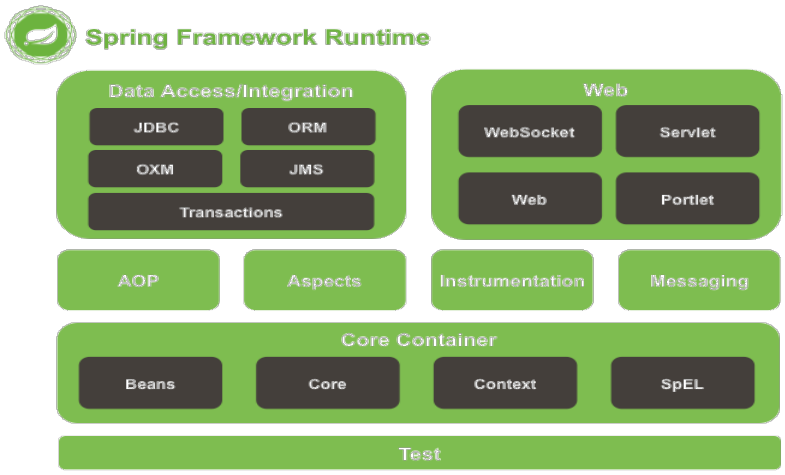
1. Spring projects



1. Spring Framework

- Spring Framework là một Java Platform cung cấp infrastructure một cách toàn diện cho việc phát triển các Java Application.
- Spring Framework chịu trách nhiệm xử lylý infrastructure, Java Developer chỉ cần tập trung vào Application.
- Thông tin thêm về spring

1. Spring framework



2. Cấu hình dependency

```
public interface AccountDAO {
}

public class JdbcAccountDao implements AccountDAO {
  private BasicDataSource dataSource;
  public JdbcAccountDao() {
    dataSource = new BasicDataSource();
    dataSource.setDriverClassName("com.mysql.jdbc.Driver");
    dataSource.setUrl("jdbc:mysql://localhost:3306/sping"
                +. "?autoReconnect=true");
    dataSource.setUsername("root");
    dataSource.setPassword("");
  }
}
```


2. Cấu hình dependency

- JdbcAccountDAO phải hiểu rõ việc implementation, creation, configuration của BasicDataSource.
- Nhiều DAO chia sẻ cùng thông tin connection. Mỗi thời điểm implementation hoặc configuration của DataSource bị thay đổi dẫn đến nhiều code change, recompilation, redeployment.

2. Dependency Injection

- Một cách để loại bỏ dependency trên BasicDataSource là
  - (1) Đưa dependency ra bên ngoài
  - (2) Tiêm (inject) dependency ngược lại
- Dễ dàng thay đổi implementation và configuration một nơi

2. Dependency Injection

- (1) Tạo thể hiện của DataSource bên ngoài
- (2) Tiêm thể hiện đó vào JdbcAccountDAO thông qua setter method hoặc constructor method

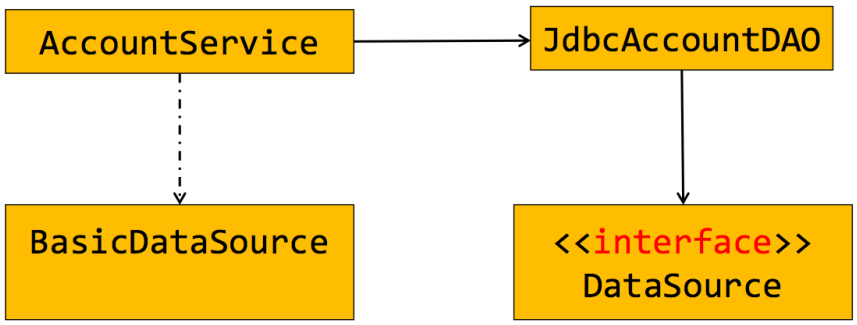
2. Dependency Injection

```
public class JdbcAccountDao implements
AccountDAO {
  private BasicDataSource dataSource;
  public JdbcAccountDao() {
  }

  public void setDataSource(DataSource ds){
    this.dataSource = ds;
  }
}
```


2. Dependency Injection

```
public class AccountService {
  private JdbcAccountDao accountDAO;
  public AccountService() {
    Properties props = new Properties();
    InputStream inputStream = this.getClass().getClassLoader()
                  .getResourceAsStream("dataSource.properties");
    props.load(inputStream);
    BasicDataSource ds = new BasicDataSource();
    ds.setDriverClassName(props.getProperty("driverClassName"));
    ds.setUrl(props.getProperty("url"));
    ds.setUsername(props.getProperty("username"));
    ds.setPassword(props.getProperty("password"));
    accountDao = new JdbcAccountDao();
    accountDao.setDataSource(ds);
  }
}
```

2. Dependency Injection



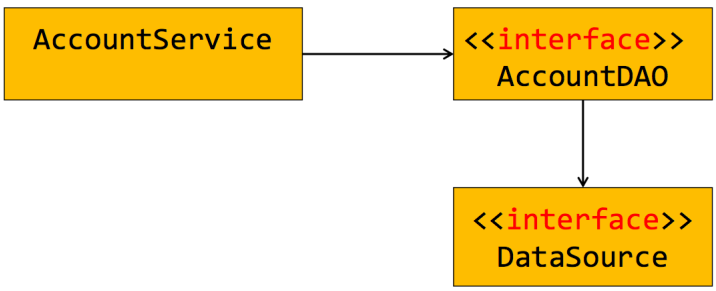
2. Dependency Injection – Inversion of control

```
package com.sip.service;
import com.sip.dao.AccountDao;

public class AccountService {
  private AccountDao accountDAO;
  public AccountService() {
  }

  public void setAccountDao(AccountDao. ad) {
    this.accountDao = ad;
  }
}
```

2. Dependency Injection – Inversion of control

```
<beans ...>
   <bean id="dataSource"
         class="org.apache.commons.dbcp.BasicDataSource"
          destroy-method="close">
      <property name="driverClassName"
                value="com.mysql.jdbc.Driver" />
      <property name="url"
                value="jdbc:mysql://localhost:3306/..." />
      <property name="username" value="someusername" />
      <property name="password" value="somepassword" />
   </bean>
   <bean id="accountDao"
         class="com.sip.dao.jdbc.JdbcAccountDao">
      <property name="dataSource" ref="dataSource" />
   </bean>
   <bean id="accountService"
         class="com.sip.service.AccountService">
      <property name="accountDao" ref="accountDao" />
   </bean>
</beans>
```


2. Dependency Injection – Inversion of control



2. Dependency Injection – Inversion of control

```
public class ConsoleApp {
  public static void main(…) {
    ApplicationContext appCtx =
              new ClassPathXmlApplicationContext(
                    "applicationContext.xml");
    AccountService accountService =
       (AccountService)appCtx.getBean("accountService");
  }
}
```


3. Ví dụ Bean Configuration

- Xây dựng chương trình cho phép lấy danh sách các tài khoản trễ hạn thanh toán từ tập tin csv có định dạng (số tài khoản, số dư, lần trả cuối)

  100,0,09012008
  200,100,08012008
  300,-100,09012008
- Tài khoản trễ hạn thanh toán là tài khoản có số dư chưa thanh toán từ 30 ngày trở lên.

3. Ví dụ Bean Configuration

- Bước 1: cài đặt Account POJO
- Bước 2: Khai báo interface AccountDAO
- Bước 3: Cài đặt CsvAccountDAO
- Bước 4: Cài đặt AccountService
- Bước 5: Cấu hình các bean trong applicationContext.xml
- Bước 6: thử nghiệm

3. Account

```
public class Account {
  private String accountNo;
  private int balance;
  private Date lastPaidOn;

  public Account(String accNo, int b, Date d) {
    accountNo = accNo;
    balance = b;
    lastPaidOn = d;
  }

  //Getters …
}
```

3. AccountDAO

```
public interface AccountDAO {
  List<Account> findAll() throws Exception;
}
```


3. CsvAccountDAO

```
public class CsvAccountDAO implements
                AccountDAO {
  public void setCsvResource(Resource csvFile) {
    this.csvResource = csvFile;
  }
  …
```

3. CsvAccountDAO

```
public List<Account> findAll() throws Exception {
  List<Account> results = new ArrayList<Account>();
  DateFormat fmt = new SimpleDateFormat("MMddyyyy");
  BufferedReader br = new BufferedReader(new FileReader(.
  csvResource.getFile()));
  String line;
  while ((line = br.readLine()) != null) {
    String[] fields = line.split(",");
    String accountNo = fields[0];
    int balance = int.parseInt(fields[1]);
    Date lastPaidOn= fmt.parse(fields[2]);
    Account acc= new Account(accountNo, balance, lastPaidOn);
    results.add(account);
  }
  br.close();
  return results;
}
```


3. AccountService

```
public class AccountService {
  private AccountDao accountDAO;
  public AccountService() {
  }

  public void setAccountDao(AccountDao. ad) {
    this.accountDao = ad;
  }
  …
```

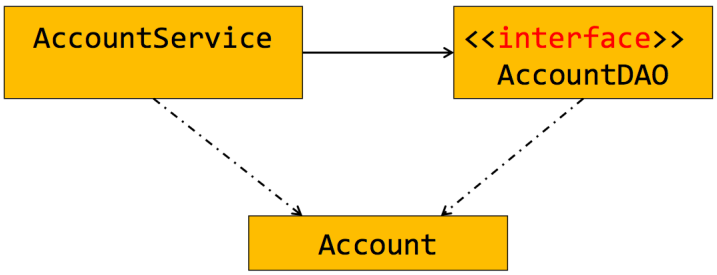
3. AccountService

```
public List<Account> findDeliquentAccounts() throws Exception {
  List<Account> dAccounts = new ArrayList<Account>();
  List<Account> accounts = accountDao.findAll();
  Date thirtyDaysAgo = daysAgo(30);
  for (Account account : accounts) {
    boolean owesMoney =
          account.getBalance().compareTo(BigDecimal.ZERO) > 0;
    boolean thirtyDaysLate =
          account.getLastPaidOn().compareTo( thirtyDaysAgo) <= 0;
    if(owesMoney && thirtyDaysLate) {
      dAccounts.add(account);
    }
  }
  return delinquentAccounts;
}
```


3. applicationContext.xml

```
<beans . . .>
 <bean id="accountDao"
                class="com.sip.dao.csv.CsvAccountDao">
   <property name="csvResource" value="accounts.csv" />
 </bean>
 <bean id="accountService"
                class="com.sip.service.AccountService">
   <property name="accountDao" ref="accountDao" />
 </bean>
</beans>
```

3. ApplicationContext.xml



3. ConsoleApp

```
public class ConsoleApp {
public static void main(String[] args) throws Exception {
  ApplicationContext ac= new
     ClassPathXmlApplicationContext(
                        "applicationContext.xml");
  AccountService accountService =
  (AccountService) ac.getBean("accountService");
  List<Account> delinquentAccounts = accountService
           .findDeliquentAccounts();
  for (Account a : delinquentAccounts) {
    System.out.println(a.getAccountNo());
  }
}
}
```

4. Wiring Bean: XML

- Các module như Data Access/Integration, Web, AOP, Instrument, Test được hỗ trợ bởi Core Container
- Tìm hiểu cách thức để wire các bean là cực kỳ quan trọng để hiểu và sử dụng Spring Framework

4.1 Wiring Bean: XML – bean namespace

- Bean namespace được sử dụng để
  - (1) Define các bean
  - (2) Wire các dependency relationship

4.1 Wiring Bean: XML – bean namespace

```
<?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-4.0.2.xsd">
  <!-- Define các bean -->
  ...
  <!-- Wire các dependency relationship -->
  ...
</beans>
```


4.1 Wiring Bean: XML – bean namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<beans  …>
  <!-- Define các bean -->
  <bean id="accountService"
   class="org.nhanh.model.service.AccountService"/>
  <!-- Wire các dependency -->
  <bean id="accountDao"
    class="com.sip.dao.jdbc.JdbcAccountDao" />
  <bean id="accountService"
    class="com.sip.service.AccountService">
    <property name="accountDao" ref="accountDao" />
  </bean>
</beans>
```

4.2 Wiring Bean: XML – bean namespace

- Có 2 loại injection
  - Setter injection
  - Constructor injection

4.2 Wiring Bean: XML – setter injection

```
public class AccountService {
  private AccountDao accountDao;
  public void setAccountDao(AccountDao accountDao){
    this.accountDao = accountDao;
  }
  // ...
}
```

```
<bean id="accountDao"
  class="com.sip.dao.jdbc.JdbcAccountDao" />
<bean id="accountService"
  class="com.sip.service.AccountService">
   <property name="accountDao" ref="accountDao" />
</bean>
```

4.2 Wiring Bean: XML – constructor injection

```
public class AccountService {
  private AccountDao accountDao;
  public AccountService(AccountDao accountDao) {
    this.accountDao = accountDao;
  }
  // ...
}
<bean id="accountDao"
  class="com.sip.dao.jdbc.JdbcAccountDao" />
<bean id="accountService"
  class="com.sip.service.AccountService">
    <constructor-arg ref="accountDao" />
</bean>
```


4.3 Wiring Bean: XML – configure bean property

```
<bean id="dataSource"
    class="org.apache.commons.dbcp.BasicDataSource"
    destroy-method="close">
  <property name="driverClassName"
            value="com.mysql.jdbc.Driver" />
  <property name="url“ value="jdbc:mysql://..." />
  <property name="username" value="someusername" />
  <property name="password" value="somepassword" />
</bean>

<bean id="accountDao"
     class="com.sip.dao.jdbc.JdbcAccountDao">
  <property name="dataSource" ref="dataSource" />
</bean>
```

4.3 Wiring Bean: XML – configure bean property

```
<bean class="org.springframework.beans.factory
            .config.PropertyPlaceholderConfigurer">
  <property name="location"
            value="springbook.properties" />
</bean>
<bean id="accountService"
      class="com.sip.service.AccountService">
  <property name="accountDao" ref="accountDao" />
</bean>
<bean id="accountDao"
      class="com.sip.dao.jdbc.JdbcAccountDao">
  <property name="dataSource" ref="dataSource" />
</bean>
```


4.3 Wiring Bean: XML – configure bean property

```
<bean id="dataSource"
      class="org.apache.commons.dbcp.BasicDataSource"
      destroy-method="close">
  <property name="driverClassName"
            value="${dataSource.driverClassName}" />
  <property name="url" value="${dataSource.url}" />
  <property name="username"
            value="${dataSource.username}" />
  <property name="password"
            value="${dataSource.password}" />
</bean>
```

4.4 Wiring Bean: XML – bean scope

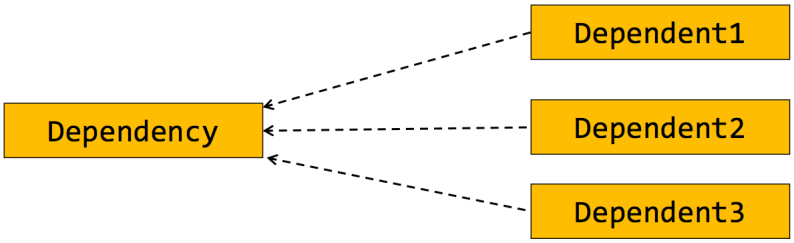
- Singleton
- Prototype
- Request
- Session
- Global session

4.4 Wiring Bean: XML – singleton scope

- Singleton scope là phạm vi mặc định khi định nghĩa các bean trong Spring
- Singleton-scope bean trong Spring bảo đảm một instance tồn tại trên một container

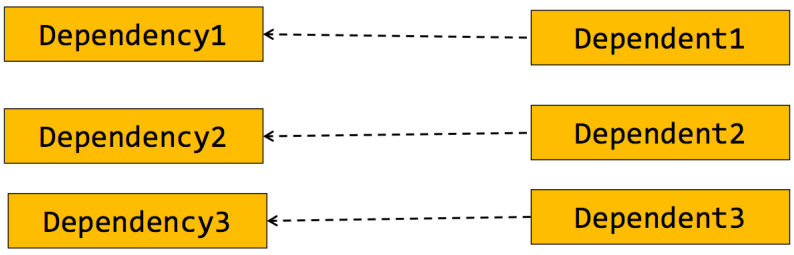
4.4 Wiring Bean: XML – singleton scope

```
<bean id="accountDao"
      class="com.sip.dao.jdbc.JdbcAccountDao"
      scope="singleton"/>
<bean id="accountDao"
      class="com.sip.dao.jdbc.JdbcAccountDao"/>
```



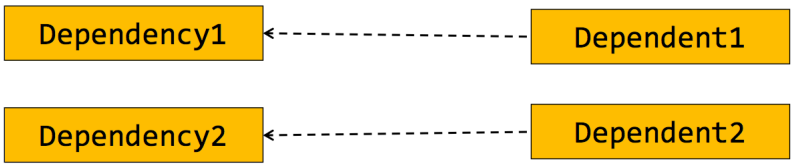
4.4 Wiring Bean: XML – prototype scope

- **Prototype-scoped bean** trong Spring **bảo đảm mỗi instance được tạo mới** khi gọi phương thức **ApplicationContext#getBean**().



4.4 Wiring Bean: XML – prototype scope

```
<bean id="accountDao"
      class="com.sip.dao.jdbc.JdbcAccountDao"
      scope="prototype" />
<bean id="accountService" <!–singleton scope
      class="com.sip.service.AccountService">
  <property name="accountDao" ref="accountDao" />
</bean>
```



4.4 Wiring Bean:XML request | session | global session

- Ba phạm vi request, session, global session đư ợc sử dụng trong các web application.
- **Request-scoped bean** trong Spring bảo đảm mỗi instance được tạo cho mỗi **HTTP Request**.
- **Session-scoped bean** trong Spring bảo đảm mỗi instance được tạo cho mỗi **Session**.
- **Global session-scoped bean** trong Spring bảo đảm mỗi instance được tạo cho một **Global Session**. Chỉ có thể ứng dụng trong **portlet web application**

4.4 Wiring Bean:XML

request | session | global session

```
<web-app>
  ...
  <listener>
    <listener-class>
      org.springframework.web.context.
      request.RequestContextListener
    </listener-class>
  </listener>
  ...
</web-app>
```


4.5 Wiring Bean: XML – p namespace

- Thông thường các Property được cấu hình thông qua XML Element.
- **P namespace** cho phép **cấu hình** các **Property** thông qua **Attribute**.
- Cách dùng
  - **xmlns:p="http://www.springframework.org/schema/p"**
  - **p:[propertyName] = "Value"**

4.5 Wiring Bean: XML – p namespace

```
<beans
   xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:p="http://www.springframework.org/schema/p"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-4.0-2.xsd">
  ...
  <bean id="dataSource"
        class="org.apache.commons.dbcp.BasicDataSource"
        destroy-method="close"
      p:driverClassName="${dataSource.driverClassName}"
      p:url="${dataSource.url}"
      p:username="${dataSource.username}"
      p:password="${dataSource.password}" />
  ...
</beans>
```


4.5 Wiring Bean: XML – p namespace

```
<bean id="accountService"
     class="com.sip.service.AccountService".
     p:accountDao-ref="accountDao" />
```

4.5 Wiring Bean: XML – p namespace

```
public Person(String firstName, String lastName) {
      this.firstName = firstName;
      this.lastName = lastName;
}

<beans …>
      <bean id="joshAsPerson" class="foo.Person"
            c:firstName="Joshua"
            c:lastName="White" />
      <bean id="willieAsPerson" class="foo.Person"
            c:_0="Willie”
            c:_1="Wheeler" />
</beans>
```


5. Wiring Bean: Annotation

```
<bean id="accountDao"
      class="com.sip.dao.jdbc.JdbcAccountDao" />
<bean id="accountService"
      class="com.sip.service.AccountService"
      p:accountDao-ref="accountDao" />

import org.springframework.beans.factory.annotation.Autowired;
public class AccountService {
      @Autowired
      private AccountDao accountDao;
      ...
}
```

5. Wiring Bean: Annotation – Context Namespace

```
<context:annotation-config />
<bean id="accountDao"
      class="com.sip.dao.jdbc.JdbcAccountDao" />
<bean id="accountService"
      class="com.sip.service.AccountService"
      p:accountDao-ref="accountDao" />

import org.springframework.beans.factory.annotation.Autowired;
public class AccountService {
      @Autowired
      private AccountDao accountDao;
      ...
}
```


5. Wiring Bean: Annotation – Context Namespace

```
import org.springframework.beans.factory.annotation.Autowired;
public class AccountService {
      @Autowired
      private AccountDao accountDao;
      ...
}

<bean id="jdbcAccountDao"
  class="com.sip.dao.jdbc.JdbcAccountDao" />
<bean id="hibernateAccountDao"
  class="com.sip.dao.hibernate.HibernateAccountDao" />
<bean id="accountService"
  class="com.sip.service.AccountService" />
```

5. Wiring Bean: Annotation – Context Namespace

```
Exception in thread "main"
org.springframework.beans.factory.BeanCreationException:
Error creating bean with name
'accountService': Autowiring of fields failed;
nested exception is
org.springframework.beans.factory.BeanCreationException:
Could not autowire field: private com.sip.dao.AccountDao
com.sip.service.AccountService.accountDao;
nested exception is
org.springframework.beans.factory.NoSuchBeanDefinitionEx
ception: No unique bean of type [com.sip.dao.AccountDao]
is defined:expected single matching bean but found 2:
[jdbcAccountDao, hibernateAccountDao]
```


5. Wiring Bean: Annotation – Context Namespace

```
import org.springframework.beans.factory.annotation.Autowired;
public class AccountService {
      @Autowired
      @Qualifier("hibernateAccountDao")
      private AccountDao accountDao;
      ...
}

<bean id="jdbcAccountDao"
  class="com.sip.dao.jdbc.JdbcAccountDao" />
<bean id="hibernateAccountDao"
  class="com.sip.dao.hibernate.HibernateAccountDao" />
<bean id="accountService"
  class="com.sip.service.AccountService" />
```

5. Wiring Bean: Annotation - Stereotype

- Khi sử dụng **@Autowired**, container chỉ làm đúng **một nhiệm vụ** đó là **wire các bean một cách tự động**.
- Khi đó **vẫn phải define các bean** thông qua cấu hình **XML**.
- Khi sử dụng **stereotype annotation** thì **không cần phải define các bean** một cách tường minh thông qua cấu hình **XML**. Spring sẽ giúp làm việc đó một cách tự động.
- Spring hỗ trợ 4 loại stereotype annotation **@Component, @Repository, @Service, @Controller**

5. Wiring Bean: Annotation - Stereotype

```
import org.springframework.stereotype.Component;
@Component
public class JdbcAccountDao implements AccountDao {
      ...
}
```

5. Wiring Bean: Annotation - Stereotype

- **`@Repository, @Service, @Controller`** là một dạng đặc biệt của **`@Component`**.
- **`@Controller`** đánh dấu một class là một controller trong **spring web mvc**.

5. Wiring Bean: Annotation – component scan

- Trong file cấu hình XML thêm

```
<context:component-scan base-package="com.sip.dao"/>
```

- Có thể khai báo **`context:component-scan. nhiều lần`**

```
<context:component-scan base-package="com.sip.dao"/>
<context:component-scan base-package="com.sip.service"/>
<context:component-scan base-
package="com.sip.controller"/>
```

- Khi khai báo **`context:component-scan`** thì không cần khai báo **`context:annotation-config`**.

Hỏi đáp

Tài liệu tham khảo

- Tài liệu giảng dạy – ThS. Nguyễn Hoàng Anh, ĐH. Khoa học tự nhiên
- What's New in the Spring Framework
- https://www.javatpoint.com/spring-modules