* 学习目标

* 能够理解Spring的概述

  * 解决业务层与其他层松耦合问题

  * 2003兴起，IOC，AOP，...

* 能够理解Spring 架构图

  * IOC,AOP,Test,JDBC,ORM,WEB,WebSocket,集成其他框架

* 能够理解Spring的优点

  * ....

* 能够掌握自定义IOC

  * BeanFactory--->Map--->配置--->getBean

    * dom4j,反射

* 能够掌握Spring的IOC的XML开发方式

  * 快速开发

  * 细节：SpringEL表达式

* 能够掌握Spring的IOC注解的开发方式

  * @Component，@Repository，@Service,@Controller

  * @Value，@AutoWired @Qualifier,@Resource,@Scope

  * @Configuration,@ComponentScan,@Bean,@PropertySource,@Import

---

* 回顾

  * MyBatis

  * 动态SQL

    * JDBC:拼接SQL语句

    * MyBatis提供动态SQL标签

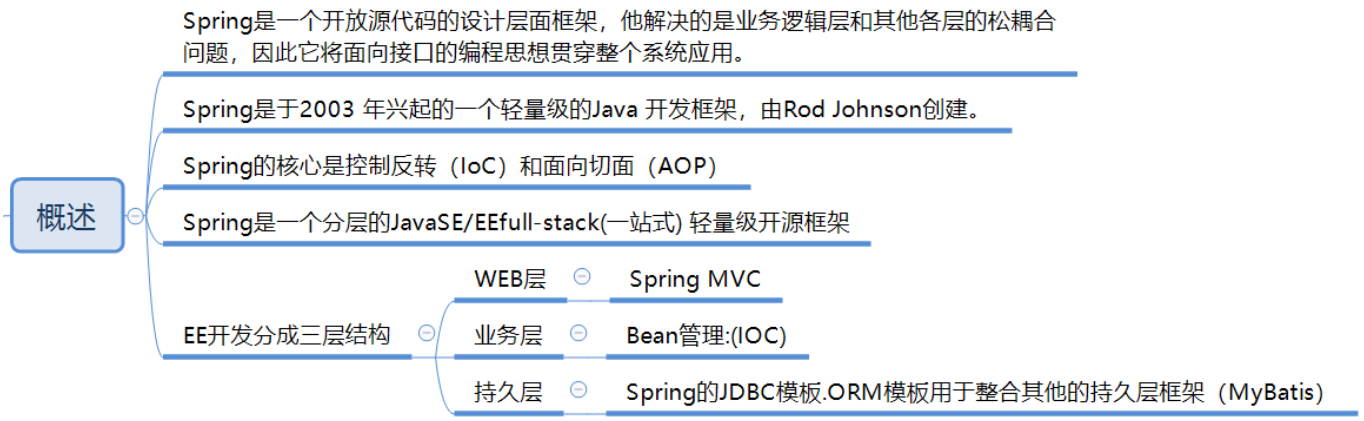
      * if，where，set，foreach,choosewhenotherwise，trim，sql,...

  * 缓存

    * 一级缓存，二级缓存

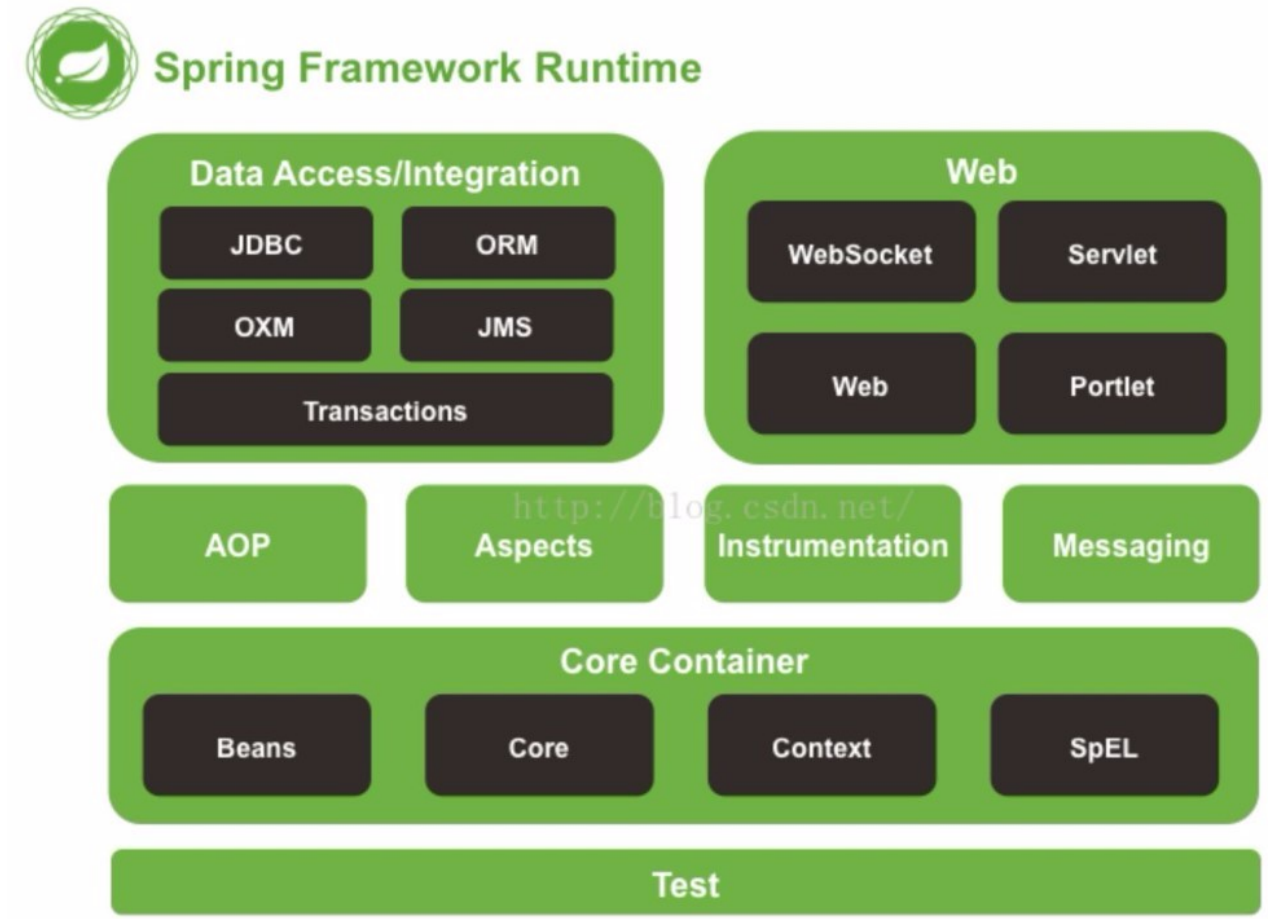
* 调用存储过程,...

* 能够理解Spring的概述

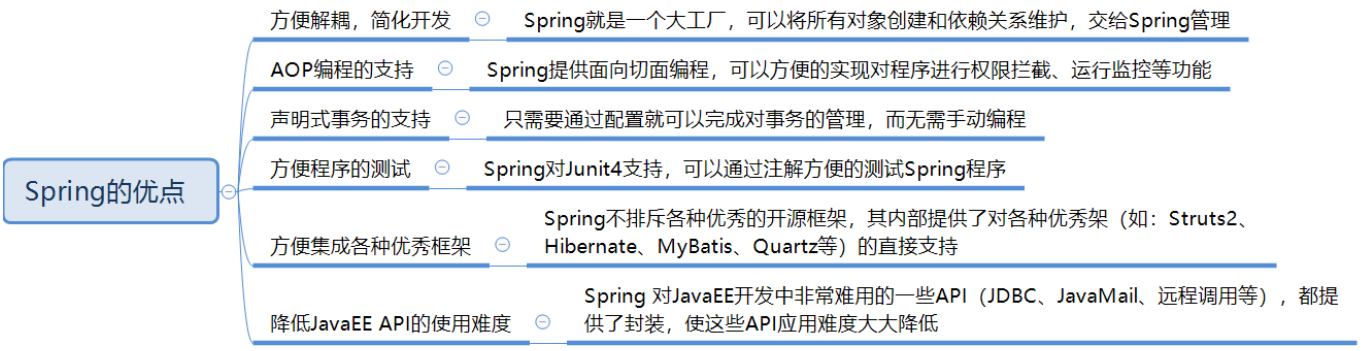



```
Spring 介绍：
https://baike.baidu.com/item/spring/85061?fr=aladdin
Rod Johnson 介绍：
https://baike.baidu.com/item/Rod%20Johnson/1423612?fr=aladdin
Spring 家族：
http://spring.io/projects
```

* 能够理解Spring 架构图

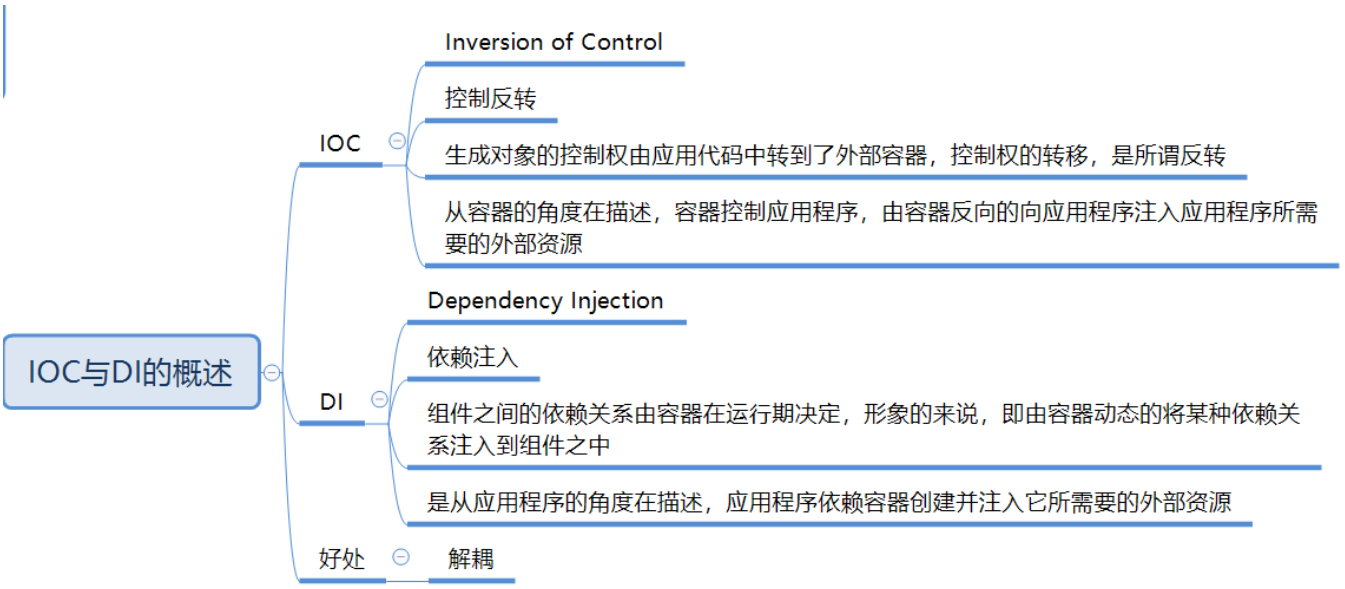


* 能够理解Spring的优点



* 能够掌握自定义IOC

* IOC&DI的概述



* 自定义IOC

BeanFactory：加载配置文件，生成bean对象 ,提供方法（通过id可以获得到对象）

配置文件：xml

```
<beans>
<bean id="userDao"  class=".."/>
<bean id="userService" class="...">
<property name="userDao" ref="userDao">
</property>
</bean>
</beans>
```

解析文件：dom4j
反射：产生对象，设置属性
放进容器
提供方法，从容器获取对象

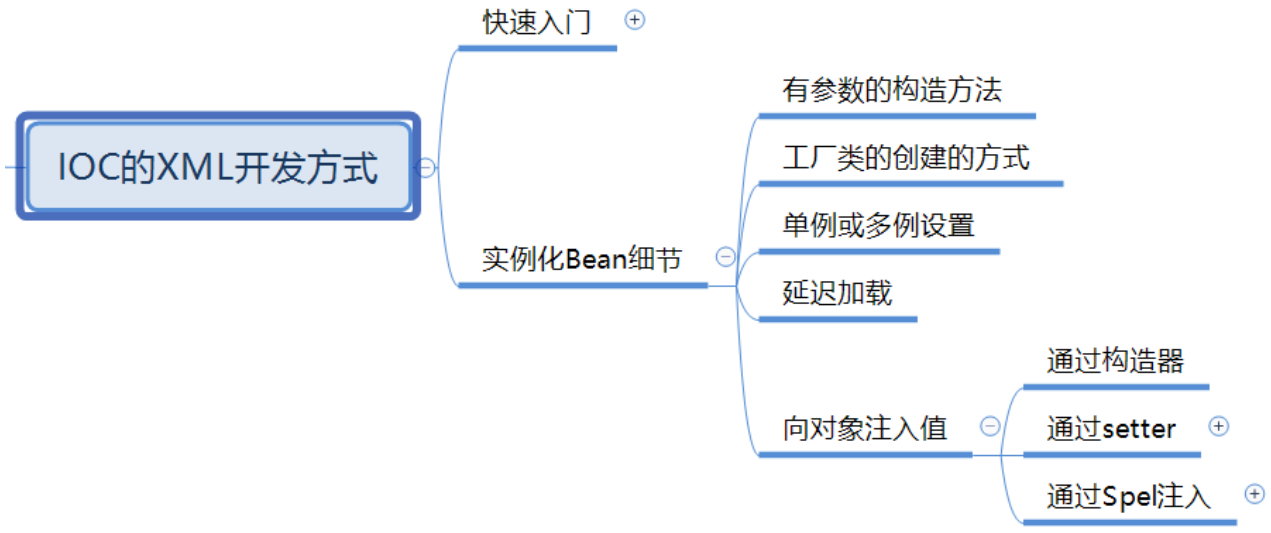
```
* 自定义IOC
 * 前期准备
@Data
@AllArgsConstructor
@NoArgsConstructor
public class User implements Serializable {
    private int id;
    private String username;
    private String psw;
    private String sex;
}
```

```
public interface UserDao {
    public void addUser(User user);
}
public interface UserService {
    public void reg(User user);
}
public class UserServiceImpl implements UserService {
    private UserDao userDao;
    public UserDao getUserDao() {
        return userDao;
    }
    public void setUserDao(UserDao userDao) {
        this.userDao = userDao;
    }
    @Override
    public void reg(User user) {
        userDao.addUser(user);
    }
}
* 配置文件
<?xml version="1.0" encoding="utf-8"?>
<beans>
    <bean id="userDao" class="com.lg.dao.impl.UserDaoImpl"></bean>
    <bean id="userService" class="com.lg.service.impl.UserServiceImpl">
        <property name="userDao" ref="userDao"></property>
    </bean>
</beans>
* IOC工厂
public class BeanFactory {
    /**
     * 存储对象
     * key：对象别名（其实就是id）
     * value：具体的对象
     */
    private static HashMap<String,Object> beans=new HashMap<String, Object>();

    public BeanFactory(String path) {
        InputStream is = BeanFactory.class.getClassLoader().getResourceAsStream
        SAXReader reader = new SAXReader();
        try {
```

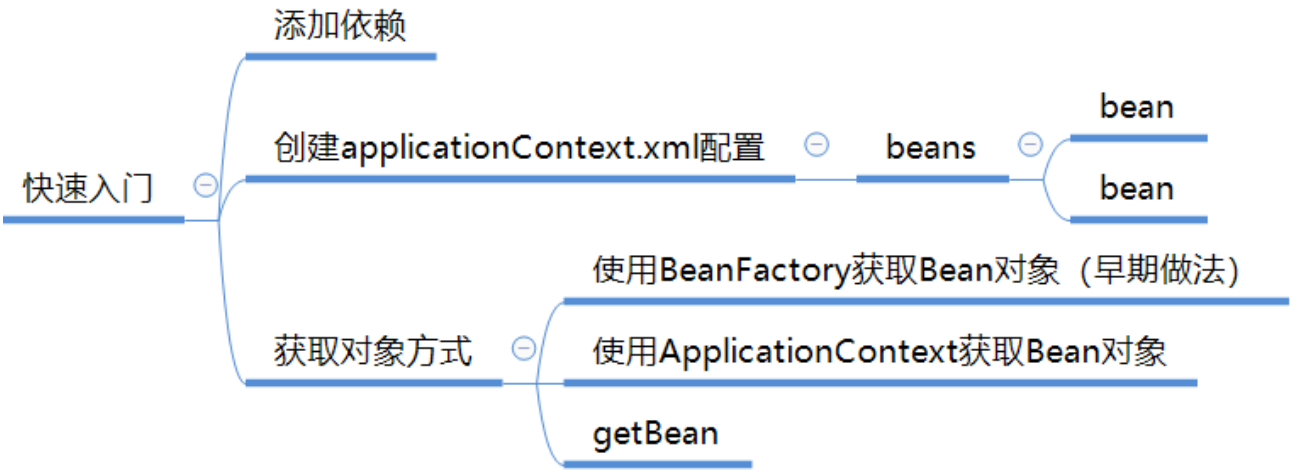
```
            Document doc = reader.read(is);
            // 获得根节点
            Element root = doc.getRootElement();
            // 获取所有bean节点
            List<Element> beanElements = root.elements();
            for (Element beanElement : beanElements) {
                // 对象别名
                String id = beanElement.attributeValue("id");
                String className = beanElement.attributeValue("class");
                // 通过反射产生对象
                Object obj = Class.forName(className).newInstance();
                // 存放对象
                beans.put(id, obj);
            }
            // 处理：property 标签
            for (Element beanElement : beanElements) {
                List<Element> propertyElements = beanElement.elements("property
                if (propertyElements != null) {
                    // 处理多个properties标签
                    // 从容器获得当前的对象
                    Object obj = beans.get(beanElement.attributeValue("id"));
                    for (Element propertyElement : propertyElements) {
                        String name = propertyElement.attributeValue("name");
                        String ref = propertyElement.attributeValue("ref");
                        // 从容器获取需要被赋值的对象
                        Object refObj = beans.get(ref);
                        // 获得当前的对象的setter方法并把需要被赋值的对象设置进去
                        // 构建setter的方法
                        // setUserDao
                        String setter = "set" + name.substring(0, 1).toUpperCas
                        Method method = obj.getClass().getDeclaredMethod(setter
                        // 调用setter的方法
                        method.invoke(obj, refObj);
                    }
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
92      /**
93       * 通过别名获取对象
94       * @param name
95       * @return
96       */
97      public Object getBeans(String name){
98          return beans.get(name);
99      }
100 }
101 * 单元测试
102   @Test
103   public void test1(){
104      String path="beans.xml";
105      BeanFactory beanFactory=new BeanFactory(path);
106      UserService userService = (UserService) beanFactory.getBeans("userService"
107      userService.reg(new User());
108    }
```

* 能够掌握Spring的IOC的XML开发方式



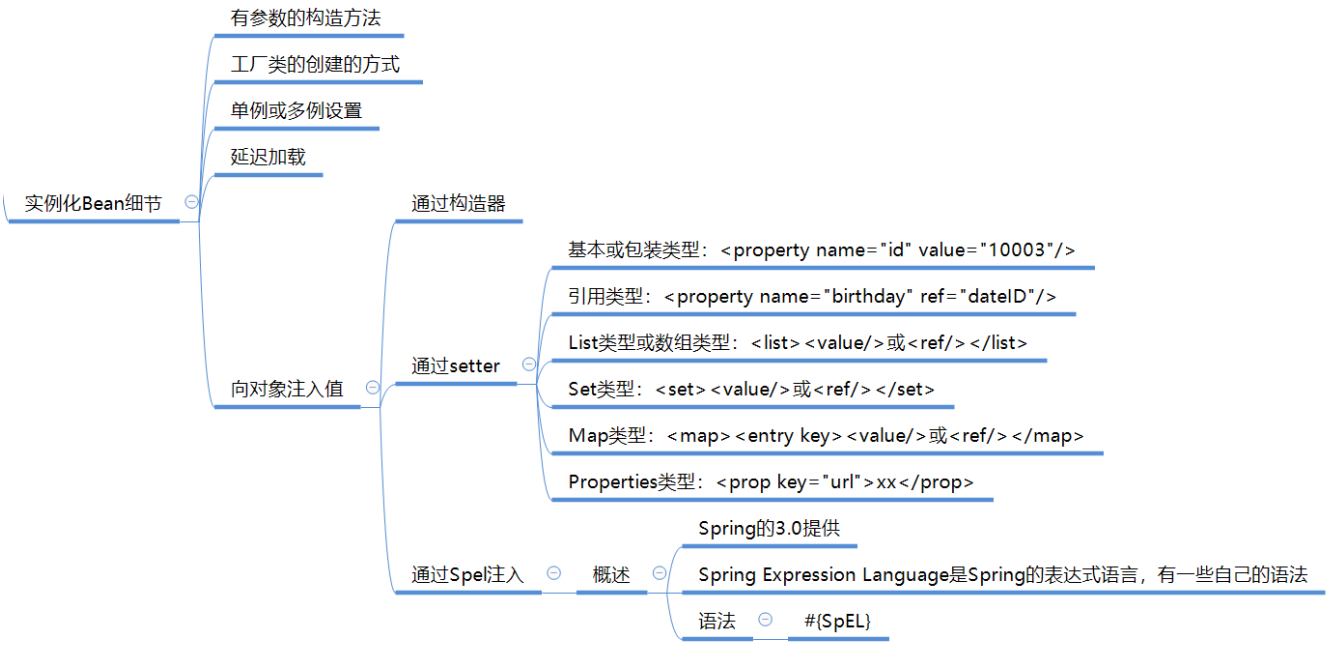
* 快速入门




```
* 案例一（快速入门）
 * 前期的准备：
    * 实体Bean：User,UserDao,UserDaoImpl
 * 添加依赖
 <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.2.2.RELEASE</version>
 </dependency>
* 温馨提醒：观察依赖，发现aop，beans，core，expression依赖也引进来了
 * 添加配置文件
<?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 id="userDao" class="com.lg.dao.impl.UserDaoImpl"></bean>
</beans>

* 单元测试
  @Test
public void test1(){
   //了解：早期的做法
   BeanFactory beanFactory=new XmlBeanFactory(new ClassPathResource("applicatio
   UserDao userDao = (UserDao) beanFactory.getBean("userDao");
   userDao.addUser(new User());
```

```
 }
@Test
public void test2(){
  ApplicationContext context=new ClassPathXmlApplicationContext("applicationCo
  UserDao userDao = (UserDao) context.getBean("userDao");
  userDao.addUser(new User());
 }
```

* 实现Bean的细节




```
* 案例一（有参数的构造器）
 * 配置
  * 方式一
<bean id="user" class="com.lg.bean.User">
        <constructor-arg name="id" value="1"/>
        <constructor-arg name="username" value="xiaoming"/>
        <constructor-arg name="psw" value="123"/>
        <constructor-arg name="sex" value="男"/>
</bean>
  * 方式二
<bean id="user" class="com.lg.bean.User">
        <constructor-arg value="2"/>
        <constructor-arg value="xiaohei"/>
        <constructor-arg value="456"/>
        <constructor-arg value="女"/>
```

```
</bean>
* 单元测试
 @Test
public void test3(){
    ApplicationContext context=new ClassPathXmlApplicationContext("applicationC
    User user = (User) context.getBean("user");
     System.out.println(user);
   }

* 案例二（工厂类创建对象形式）
public class UserFactory {
    public User getInstance(){
        return new User(10002,"小黑","123123","男");
    }
    public static User getStaticInstance(){
        return new User(10003,"小白","456789","女");
    }
}
 * 配置
<bean id="userFactory" class="com.lg.factory.UserFactory"/>
<bean id="u1" factory-bean="userFactory" factory-method="getInstance"/>
<bean id="u2" class="com.lg.factory.UserFactory" factory-method="getStaticInsta
 * 单元测试

* 案例三（单例或者多例）
  * 温馨提醒：默认是单例的
 * 配置
 <bean id="u3" class="com.lg.bean.User" scope="singleton|prototype"></bean>
 * 单元测试
  @Test
   public void test3(){
      ApplicationContext context=new ClassPathXmlApplicationContext("applicatic
      User user = (User) context.getBean("u3");
      User user1 = (User) context.getBean("u3");
      System.out.println(user==user1);
    }
* 案例四：(延迟加载)
  * 对于多例来说，都是延迟加载的
  * 只对单例有效
   * lazy-init 为fasle：工厂加载xml之后，会产生对象
```

```
   * lazy-init 为true：工厂加载xml之后，不会产生对象
 * 代码
@Data
@AllArgsConstructor
public class User implements Serializable {
    private int id;
    private String username;
    private String psw;
    private String sex;
    public User(){
        System.out.println("User 初始化");
    }
}
* 配置
<bean id="u5" class="com.lg.bean.User" scope="singleton" lazy-init="true"></bea
* 单元测试
 @Test
public void test5(){
  ApplicationContext context=new ClassPathXmlApplicationContext("applicationCor
}

* 案例五：通过setter注入
@Data
public class TestBean {
    /**注入Integer类型参数*/
    private Integer id;
    /**注入String类型参数*/
    private String name;
    /**注入实体Bean*/
    private User user;
    /**注入数组*/
    private Object[] array;
    /**注入List集合*/
    private List<Object> list;
    /**注入Set集合*/
    private Set<Object> set;
    /**注入Map键值对*/
    private Map<Object, Object> map;
    /**注入properties类型*/
    private Properties properties;
```

```
    /**注入空字符串*/
    private String emptyValue;
    /**注入null值*/
    private String nullValue = "";
    /**检测注入的属性是否全部正确*/
    public Boolean checkAttr() {
        if(id == null) {
            return false;
        } else {
            System.out.println("id:" + id);
        }
        System.out.println("--------------------------");
        if(name == null) {
            return false;
        } else {
            System.out.println("name:" + name);
        }
        System.out.println("--------------------------");
        if(user == null) {
            return false;
        } else {
            System.out.println("Bean:" + user.getId() + "|" +
                    user.getUsername()+ "|" + user.getPsw());
        }
        System.out.println("--------------------------");
        if(array == null) {
            return false;
        } else {
            System.out.println("array:");
            for (Object object : array) {
                System.out.println(object.toString());
            }
        }
        System.out.println("--------------------------");
        if(list == null) {
            return false;
        } else {
            System.out.println("list:");
            for (Object object : list) {
                System.out.println(object.toString());
```

```
            }
        }
        System.out.println("--------------------------");
        if(set == null) {
            return false;
        } else {
            System.out.println("set:");
            for (Object object : set) {
                System.out.println(object.toString());
            }
        }
        System.out.println("--------------------------");
        if(map == null) {
            return false;
        } else {
            Set<Map.Entry<Object, Object>> set = map.entrySet();
            System.out.println("map:");
            for (Map.Entry<Object, Object> entry : set) {
                System.out.println(entry.getKey() + "|" + entry.getValue());
            }
        }
        System.out.println("--------------------------");
        if(properties == null) {
            return false;
        } else {
            Set<Map.Entry<Object, Object>> set = properties.entrySet();
            System.out.println("properties:");
            for (Map.Entry<Object, Object> entry : set) {
                System.out.println(entry.getKey() + "|" + entry.getValue());
            }
        }
        System.out.println("--------------------------");
        if(!"".equals(emptyValue)) {
            return false;
        }
        System.out.println("--------------------------");
        if(!(null == nullValue)) {
            return false;
        }
        System.out.println("--------------------------");
```

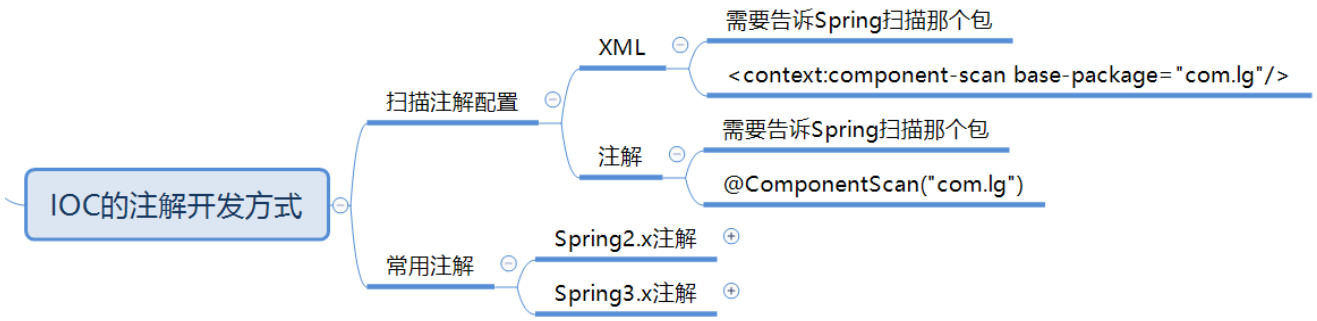
```
        System.out.println("全部正确！！！");
        return true;
    }
}

* 配置
 <bean id="test1" class="com.lg.bean.TestBean">
        <!-- 注入id属性 -->
        <property name="id" value="1"/>
        <!-- 使用<![CDATA[]]>标记处理XML特 殊字符 -->
        <property name="name">
            <!-- 也可以使用P&G -->
            <value><![CDATA[P&G]]></value>
        </property>
        <!-- 定义内部Bean注入 -->
        <property name="user">
            <bean class="com.lg.bean.User">
                <property name="id" value="1"/>
                <property name="username" value="xiaohei"/>
                <property name="psw" value="123123"/>
            </bean>
        </property>
        <!-- 注入数组类型 -->
        <property name="array">
            <array>
                <!-- 定义数组元素 -->
                <value>array01</value>
                <value>array02</value>
                <value>array03</value>
            </array>
        </property>
        <!-- 注入List类型 -->
        <property name="list">
            <list>
                <!-- 定义list中元素 -->
                <value>list01</value>
                <value>list02</value>
                <value>list03</value>
            </list>
        </property>
```

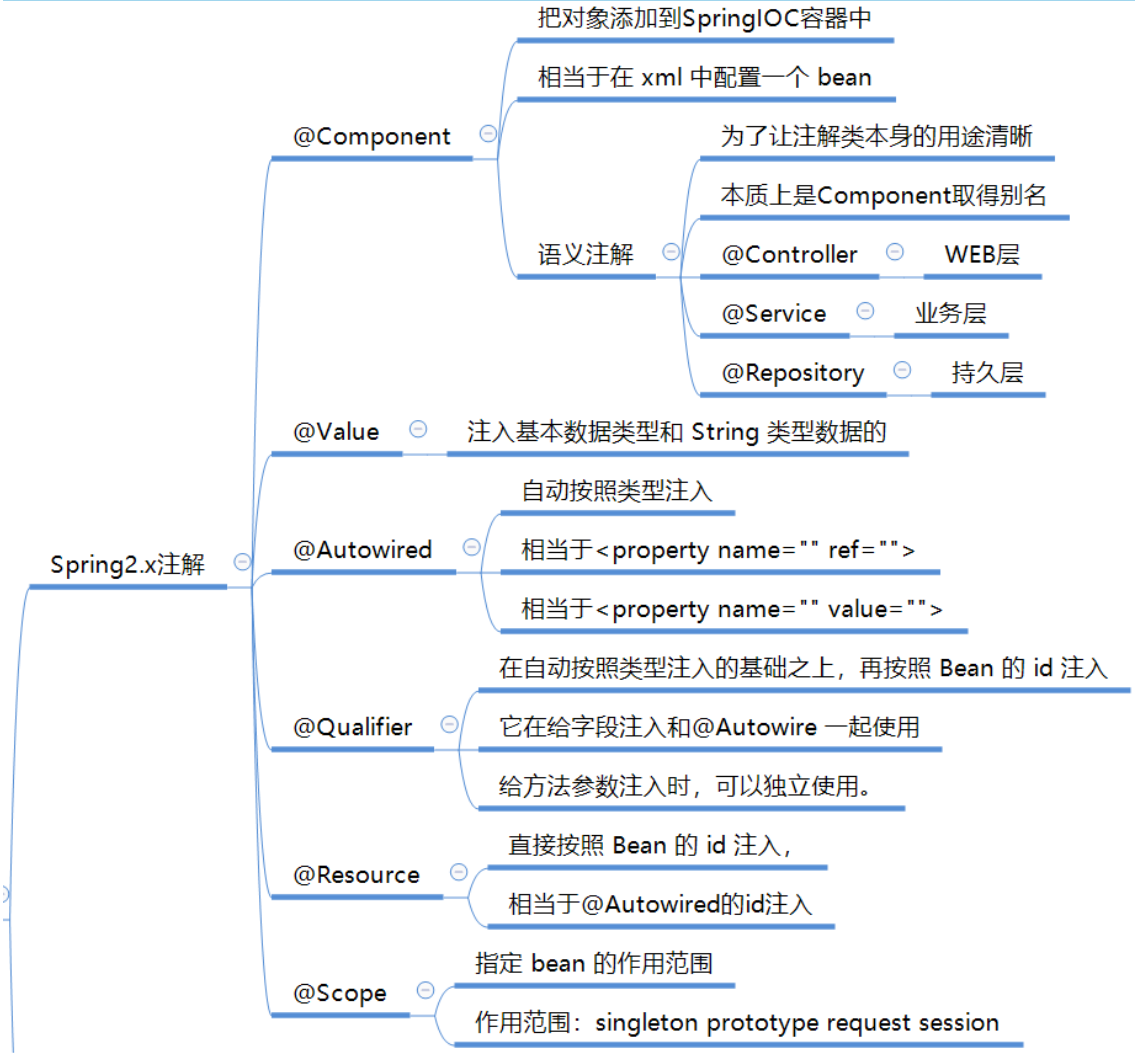
```
        <!-- 注入Set类型 -->
        <property name="set">
            <set>
                <!-- 定义set中元素 -->
                <value>set01</value>
                <value>set02</value>
                <value>set03</value>
            </set>
        </property>
        <!-- 注入Map类型 -->
        <property name="map">
            <map>
                <!-- 定义map中的键值对 -->
                <entry>
                    <key>
                        <value>mapKey01</value>
                    </key>
                    <value>mapValue01</value>
                </entry>
                <entry>
                    <key>
                        <value>mapKey02</value>
                    </key>
                    <value>mapValue02</value>
                </entry>
            </map>
        </property>
        <!-- 注入properties类型 -->
        <property name="properties">
            <props>
                <!-- 定义properties中的键值对 -->
                <prop key="propKey1">propValue1</prop>
                <prop key="propKey2">propValue2</prop>
            </props>
        </property>
        <!-- 注入空字符串 -->
        <property name="emptyValue">
            <value></value>
        </property>
        <!-- 注入null值 -->
```

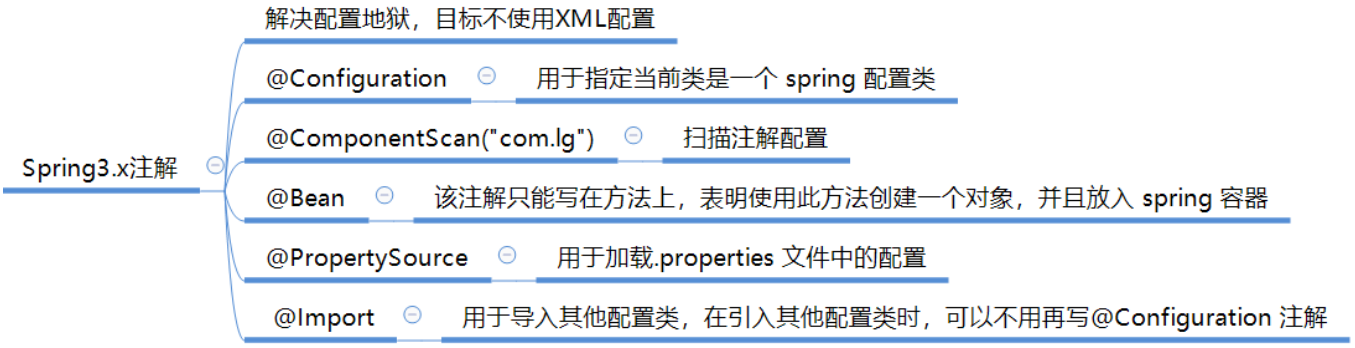
```
        <property name="nullValue">
            <null/>
        </property>
    </bean>
 * 单元测试
 @Test
public void test6() {
    ApplicationContext context = new ClassPathXmlApplicationContext("applicatio
    TestBean testBean= (TestBean) context.getBean("test1");
    testBean.checkAttr();
 }

* 案例六：Spel的注入
public class Person {
    public String say(String value){
        System.out.println(value);
        return value;
    }
    public String getName(){
        return "xiaohei";
    }
}
@Data
public class SpringEL {
    private Integer num;
    private User user;
    private String sayWhat;
    private String name;
    private int rand;
    private boolean flag;
}
* 配置
 <bean id="person" class="com.lg.bean.Person"/>
 <bean id="el1" class="com.lg.bean.SpringEL">
        <property name="num" value="#{3*5}"></property>
        <property name="sayWhat" value="#{person.say('HelloWorld')}"></property
        <property name="name" value="#{person.getName()}"></property>
        <property name="user" value="#{u5}"></property>
        <property name="rand" value="#{T(Math).random()*100}"></property>
        <property name="flag" value="#{u5.id==10003}"></property>
```

```
296  </bean>
297
298 * 单元测试
299 @Test
300 public void test8() {
301    ApplicationContext context = new ClassPathXmlApplicationContext("applicatior
302    SpringEL el= (SpringEL) context.getBean("el1");
303    System.out.println(el);
304  }
305
```

* 能够掌握Spring的IOC注解的开发方式








```
* 案例一（@Component,@Repository,Service,@Controller）：
 * 添加配置
 <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="com.lg.*"/>
 * 代码
public interface EmployeeDao {
    void addEmployee(Employee employee);
}
@Component("employeeDao")
public class EmployeeDaoImpl implements EmployeeDao {
    @Override
    public void addEmployee(Employee employee) {
        System.out.println("addEmployee...");
    }
}
* 单元测试
@Test
public void test8(){
   ApplicationContext context = new ClassPathXmlApplicationContext("applicatior
   EmployeeDao employeeDao = (EmployeeDao) context.getBean("employeeDao");
   employeeDao.addEmployee(new Employee());
}
* 温馨提醒
 * @Component替换成@Controller或者@Service或者@Repository效果一样
   它们只是Component取的别名，表达不同的层

案例二：（@Value，@AutoWired,@Qualifier,@Resource）
@Repository("employeeDao1")
public class EmployeeDaoImpl implements EmployeeDao {
    @Override
    public void addEmployee(Employee employee) {
        System.out.println("addEmployee...");
    }
}
@Repository("employeeDao2")
public class EmployeeDaoImpl2 implements EmployeeDao {
    @Override
    public void addEmployee(Employee employee) {
        System.out.println("addEmployee2...");
    }
}
public interface EmployeeService {
```

```
    void saveEmployee(Employee employee);
}
@Service("employeeService")
public class EmployeeServiceImpl implements EmployeeService {
    @Value("5")
    private int num;
    @Autowired
    @Qualifier("employeeDao2")
//    @Resource(name = "employeeDao2")
    private EmployeeDao employeeDao;
    @Override
    public void saveEmployee(Employee employee) {
        System.out.println(num);
        employeeDao.addEmployee(employee);
    }
}

@Test
public void test9(){
   ApplicationContext context = new ClassPathXmlApplicationContext("applicatior
   EmployeeService employeeService = (EmployeeService) context.getBean("employe
   employeeService.saveEmployee(new Employee());
}

* 案例三（@Scope）
@Scope("singleton|prototype")
public class EmployeeServiceImpl implements EmployeeService
单元测试
 @Test
public void test10(){
  ApplicationContext context = new ClassPathXmlApplicationContext("applicationC
  EmployeeService employeeService = (EmployeeService) context.getBean("employee
  EmployeeService employeeService1 = (EmployeeService) context.getBean("employe
  System.out.println(employeeService==employeeService1);
}
 * 温馨提醒：分别测试多例和单例的效果

 * 案例四：（无xml配置，使用注解）
  * 基本替换xml配置的功能
@Configuration
```

```java
@ComponentScan("com.lg")
public class SpringConfiguration {
    @Bean(name = "user")
    public User user(){
        return new User(2,"xiaohei","456","男");
    }
    @Bean(name = "array")
    public Object[] array(){
        // 跟配置文件有点不一样
        return new Object[]{"array1","array2"};
    }
    @Bean(name = "list")
    public List<Object> list(){
        // 跟配置文件有点不一样
        List<Object> list=new ArrayList<>();
        list.add("list1");
        list.add("list2");
        return list;
    }
    @Bean(name = "set")
    public Set<Object> set(){
        // 跟配置文件有点不一样
        Set<Object> set=new HashSet<>();
        set.add("set1");
        set.add("set2");
        return set;
    }
    @Bean(name = "map")
    public Map<Object,Object> map(){
        Map<Object,Object> map=new HashMap<>();
        map.put("key1","value1");
        map.put("key2","value12");
        return map;
    }
    @Bean(name = "properties")
    public Properties properties(){
        Properties properties=new Properties();
        properties.put("p1","value1");
        properties.put("p2","value12");
        return properties;
```

```
129     }
130     @Bean(name = "e")
131     public String e(){
132         return "";
133     }
134     @Bean(name = "ne")
135     public String ne(){
136         // 不能注入null，在@Bean中
137         return "null";
138     }
139     @Bean
140     public Person person(){
141         return new Person();
142     }
143     @Bean
144     public SpringEL el1(){
145         return new SpringEL();
146     }
147 }
148 
149 @Data
150 @Component("test1")
151 public class TestBean {
152     /**注入Integer类型参数*/
153     @Value("1")
154     private Integer id;
155     /**注入String类型参数*/
156     @Value("<![CDATA[P&G]]>")
157     private String name;
158     /**注入实体Bean*/
159     @Autowired
160     @Qualifier("user")
161     private User user;
162     /**注入数组*/
163     @Autowired
164     @Qualifier("array")
165     private Object[] array;
166     /**注入List集合*/
167     @Autowired
168     @Qualifier("list")
```

```
    private List<Object> list;
    /**注入Set集合*/
    @Autowired
    @Qualifier("set")
    private Set<Object> set;
    /**注入Map键值对*/
    @Autowired
    @Qualifier("map")
    private Map<Object, Object> map;
    /**注入properties类型*/
    @Autowired
    @Qualifier("properties")
    private Properties properties;
    /**注入空字符串*/
    @Autowired
    @Qualifier("e")
    private String emptyValue;
    /**注入null值*/
    @Autowired
    @Qualifier("ne")
    private String nullValue = "";
    /**检测注入的属性是否全部正确*/
    public Boolean checkAttr() {
        if(id == null) {
            return false;
        } else {
            System.out.println("id:" + id);
        }
        System.out.println("--------------------------");
        if(name == null) {
            return false;
        } else {
            System.out.println("name:" + name);
        }
        System.out.println("--------------------------");
        if(user == null) {
            return false;
        } else {
            System.out.println("Bean:" + user.getId() + "|" +
                    user.getUsername()+ "|" + user.getPsw());
```

```
        }
        System.out.println("--------------------------");
        if(array == null) {
            return false;
        } else {
            System.out.println("array:");
            for (Object object : array) {
                System.out.println(object.toString());
            }
        }
        System.out.println("--------------------------");
        if(list == null) {
            return false;
        } else {
            System.out.println("list:");
            for (Object object : list) {
                System.out.println(object.toString());
            }
        }
        System.out.println("--------------------------");
        if(set == null) {
            return false;
        } else {
            System.out.println("set:");
            for (Object object : set) {
                System.out.println(object.toString());
            }
        }
        System.out.println("--------------------------");
        if(map == null) {
            return false;
        } else {
            Set<Map.Entry<Object, Object>> set = map.entrySet();
            System.out.println("map:");
            for (Map.Entry<Object, Object> entry : set) {
                System.out.println(entry.getKey() + "|" + entry.getValue());
            }
        }
        System.out.println("--------------------------");
        if(properties == null) {
```

```
            return false;
        } else {
            Set<Map.Entry<Object, Object>> set = properties.entrySet();
            System.out.println("properties:");
            for (Map.Entry<Object, Object> entry : set) {
                System.out.println(entry.getKey() + "|" + entry.getValue());
            }
        }
        System.out.println("--------------------------");
        if(!"".equals(emptyValue)) {
            return false;
        }
        System.out.println("--------------------------");
        if(!(null == nullValue)) {
            return false;
        }
        System.out.println("--------------------------");
        System.out.println("全部正确！！！");
        return true;
    }
}

@Data
public class SpringEL {
    @Value("#{3*5}")
    private Integer num;
    @Value("#{user}")
    private User user;
    @Value("#{person.say('HelloWorld')}")
    private String sayWhat;
    @Value("#{person.getName()}")
    private String name;
    @Value("#{T(Math).random()*100}")
    private int rand;
    @Value("#{user.id==2}")
    private boolean flag;
}

* 单元测试
public class AppTest2
```

```
289 {
290
291     @Test
292     public void test1(){
293         //了解：早期的做法
294         BeanFactory beanFactory=new AnnotationConfigApplicationContext(SpringCo
295         UserDao userDao = (UserDao) beanFactory.getBean("userDao");
296         userDao.addUser(new User());
297     }
298     @Test
299     public void test2(){
300         ApplicationContext context=new AnnotationConfigApplicationContext(Sprir
301         UserDao userDao = (UserDao) context.getBean("userDao");
302         userDao.addUser(new User());
303     }
304
305     @Test
306     public void test3(){
307         ApplicationContext context=new AnnotationConfigApplicationContext(Sprir
308         User user = (User) context.getBean("user");
309         System.out.println(user);
310     }
311
312     @Test
313     public void test6() {
314         ApplicationContext context = new AnnotationConfigApplicationContext(Spr
315         TestBean testBean= (TestBean) context.getBean("test1");
316         testBean.checkAttr();
317     }
318
319     @Test
320     public void test7() {
321         ApplicationContext context = new AnnotationConfigApplicationContext(Spr
322         SpringEL el= (SpringEL) context.getBean("el1");
323         System.out.println(el);
324     }
325
326     @Test
327     public void test8(){
328         ApplicationContext context = new AnnotationConfigApplicationContext(Spr
```

```
        EmployeeDao employeeDao = (EmployeeDao) context.getBean("employeeDao1")
        employeeDao.addEmployee(new Employee());
    }

    @Test
    public void test9(){
        ApplicationContext context = new AnnotationConfigApplicationContext(Spr
        EmployeeService employeeService = (EmployeeService) context.getBean("em
        employeeService.saveEmployee(new Employee());
    }

    @Test
    public void test10(){
        ApplicationContext context = new AnnotationConfigApplicationContext(Spr
        EmployeeService employeeService = (EmployeeService) context.getBean("em
        EmployeeService employeeService1 = (EmployeeService) context.getBean("e
        System.out.println(employeeService==employeeService1);
    }
}

* 案例五：（@Import,@PropertySource)
* 配置文件db.propertie
lg.driver=com.mysql.jdbc.Driver
lg.url=jdbc:mysql://localhost:3306/lg01?characterEncoding=utf-8
lg.username=root
lg.password=root

* 代码
@Configuration
@PropertySource("db.properties")
@Data
public class DbConfig {
    @Value("${lg.driver}")
    private String driver;
    @Value("${lg.url}")
    private String url;
    @Value("${lg.username}}")
    private String username;
    @Value("${lg.password}")
    private String password;
```

```
}
@Configuration
@ComponentScan("com.lg")
@Import(DbConfig.class)
public class SpringConfiguration {
  ....
   @Bean
  public DbConfig dbConfig(){
     return new DbConfig();
  }
}
* 单元测试
@Test
public void test11(){
  ApplicationContext context = new AnnotationConfigApplicationContext(SpringCor
  DbConfig dbConfig= (DbConfig) context.getBean("dbConfig");
  System.out.println(dbConfig);
 }
```