装载Bean2

applicationContext

1. applicationContext.getBean("id名");
2. applicationContext.getBean(类名.Class);

1的好处是当配置有多个类需要去注入时，使用2会引起歧义，而1就不会。

依赖注入的方式

1. 属性注入
2. 构造器注入
3. 工厂方法注入（很少使用，不推荐）

属性注入

- 属性注入即通过 setter 方法注入Bean 的属性值或依赖的对象
- 属性注入使用 元素, 使用 name 属性指定 Bean 的属性名称，value 属性或 子节点指定属性值
- 属性注入是实际应用中最常用的注入方式

```
<bean id="user" class="com.my_spring.pojo.User">
 <property name="userName" value="张三"></property>
 </bean>
```

构造器注入

```
<bean id="user" class="com.my_spring.pojo.User">
   <constructor-arg name="password" value="123"index="0" type="java.lang.String"></constructor-arg>
   <constructor-arg name="id" value="001" index="1" type="java.lang.String"></constructor-arg>
   <constructor-arg name="username" value="Merz" index="2" type="java.lang.String"></constructor-arg>
 </bean>
```

当有多个bean时，可去配置index（位置），type(数据类型)，去完成构造方法的重载。

内部Bean，就是在属性内部去配置一个Bean，外部不能访问。

```
<bean id="user" class="com.my_spring.pojo.User">
 <property name="order">
  <bean  class="com.my_spring.pojo.Order">
   <property name="price" value="10"></property>
  </bean>
 </property>
 </bean>
```

补充： 元素标签为 Bean 的字符串或其它对象类型的属性注入 null 值

```
 <constructor-arg name="id"><null/></constructor-arg>
```

- 应用其他bean

将使用rel属性，属性值为在bean中配置好的id即可。

- 级联赋值

先实例化级联的bean，再进行级联（ognl的方式）。 很少用

- 集合赋值
  - 在 Spring中可以通过一组内置的 xml 标签(例如: , 或) 来配置集合属性.

- 配置 java.util.List 类型的属性, 需要指定 标签, 在标签里包含一些元素. 这些标签可以通过 指定简单的常量值, 通过 指定对其他 Bean 的引用. 通过 指定内置 Bean 定义. 通过 指定空元素. 甚至可以内嵌其他集合.
- 数组的定义和 List 一样, 都使用 配置 java.util.Set 需要使用 标签, 定义元素的方法与 List 一样
- Java.util.Map 通过 标签定义, 标签里可以使用多个 作为子标签. 每个条目包含一个键和一个值. 必须在 标签里定义键
- 因为键和值的类型没有限制, 所以可以自由地为它们指定 value, ref或元素.
- 可以将 Map 的键和值作为 的属性定义: 简单常量使用 key 和 value 来定义; Bean 引用通过 key-ref 和 value-ref 属性定义
- 使用 定义 java.util.Properties, 该标签使用多个 作为子标签. 每个 标签必须定义 key 属性.