

尚马教育 JAVA 课程

Spring 类配置

文档编号: C04_1

创建日期: 2021-01-21

最后修改日期: 2021-03-10

版本号: V3.6

电子版文件名: 尚马教育-第三阶段-4_1.spring 类配置.docx

文档修改记录:

更新日期	更新作者	更新说明	版本号
2021-01-21	冯勇涛	加入 spring 类配置容器	V3.5
2021-03-10	冯勇涛	加入 dom4j 解析 xml	V3.6

目录

尚马教育 JAVA 课程.....	1
Spring 类配置	1
1. 类配置取代 xml 配置.....	3
2. class 类配置环境	3
2.1. 创建配置类: AppConfig .java	3
2.1.1. 测试代码.....	5

知识点:

知识点 1: 认识 spring 类配置常用注解。

1. 类配置取代 xml 配置

Spring 在实际开发中有两种应用方式：

基于 xml 的配置：一般在 ssm 项目中使用。

基于类的配置：一般在 springBoot 项目中使用。

2. class 类配置环境

2.1. 创建配置类：AppConfig.java

主要注解：

@Configuration, @Bean, @Value, @ComponentScan, @Import, @EnableAspectJAutoProxy

```
/**
 * 该类的职责与 spring.xml 的职责是一样
 * 1. 声明该类是一个配置类@Configuration
 * 2. 包扫描@ComponentScan
 * 3. 引入外部资源文件@PropertySource
 * 4. 嵌套配置类@Import
 * 5. 注册 bean 对象进入容器@Bean
 * 6. 获取容器中的 property 数据@Value
 * 7. @EnableAspectJAutoProxy 开启 aop 注解识别
 */
@Configuration
@ComponentScan("com.javasm")
@Import(DaoConfig.class)
@PropertySource(value="classpath:jdbc.properties", ignoreResourceNotFound = true)
@EnableAspectJAutoProxy
public class ContextConfig {

    @Value("${jdbc.url}")

    private String url;
```

```

@Value("${jdbc.driver}")

private String driver;

@Value("${jdbc.username}")

private String username;

@Value("${jdbc.password}")

private String password;


//先按照形参名找，再按照形参类型找 bean 进行注入

@Bean

public SysuserController createUserController(ISysuserService sysuserService) {

    SysuserController sysuserController = new SysuserController();

    sysuserController.setSysuserService(sysuserService);

    return sysuserController;

}


@Bean

public ISysuserService createUserService() {

    return new SysuserServiceImpl();

}


@Bean(initMethod = "init", destroyMethod = "close")

// @Scope("prototype")

public DataSource initDruid() {

    DruidDataSource ds = new DruidDataSource();

    ds.setUrl(url);

    ds.setDriverClassName(driver);

    ds.setUsername(username);

    ds.setPassword(password);

    ds.setInitialSize(2);

```

```
        return ds;
    }
```

2.1.1. 测试代码

```
@Test
public void test1_loadClzConfig() {
    ApplicationContext ac = new AnnotationConfigApplicationContext(ContextConfig.class);
    SysuserController bean = ac.getBean(SysuserController.class);
    ((AnnotationConfigApplicationContext) ac).close();
}
```

3. Dom4J 解析 xml 文件

Mybatis.spring 等框架都采用 xml 作为配置文件,那么在框架内部必须要进行 xml 解析,这里学习如果解析 xml.

3.1. 常用组件

PC 端解析 xml,一般使用 dom4j 解析;

移动端解析 xml,一般使用 SAX 解析.

3.2. 特点

DOM4j 解析 XML 文件时,会将 XML 的所有内容读取到内存中,然后使用 DOM API 遍历 XML 树、检索所需的数据。因为 DOM 需要将所有内容读取到内存中,所以内存的消耗比较大,一般服务器端使用;

SAX (Simple API for XML) 是一个解析速度快并且占用内存少的 XML 解析器,非常适合用于 Android 等移动设备。SAX 解析 XML 文件采用的是事件驱动,也就是说,它并不需要解析完整个文档,在按内容顺序解析文档的过程中,SAX 会判断当前读到的字符是否合法 XML 语法中的某部分,如果符合就会触发事件。

3.3. Dom4j 代码

学习 dom4j 只要求熟记核心 SAXReader 对象.类似于 Properties 核心对象.

这里以解析 spring 风格 xml 文件为例:

3.3.1. Spring.xml

① jdbc.properties 文件如下:

```
jdbc.url=jdbc:mysql://127.0.0.1:3306/720A?useUnicode=true&characterEncoding=utf8&
useSSL=true&serverTimezone=UTC
jdbc.driver=com.mysql.jdbc.Driver
jdbc.username=root
jdbc.password=root
```

② Spring.xml 文件如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<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"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
https://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/aop
https://www.springframework.org/schema/aop/spring-aop.xsd">

    <context:component-scan base-package="com.javasm4"></context:component-scan>

    <aop:aspectj-autoproxy></aop:aspectj-autoproxy>

    <context:property-placeholder location="jdbc.properties"></context:property-
```

```
placeholder>

<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource" init-
method="init" destroy-method="close">

    <property name="url" value="${jdbc.url}"></property>

    <property name="driverClassName" value="${jdbc.driver}"></property>

    <property name="username" value="${jdbc.username}"></property>

    <property name="password" value="${jdbc.password}"></property>

    <property name="initialSize" value="3"></property>

</bean>

</beans>
```

3.3.2. 解析 xml,模拟 ApplicationContext 容器对象

解析 spring.xml 文件,创建出 bean 对象.

```
package com.javasm3;

import org.dom4j.Document;
import org.dom4j.DocumentException;
import org.dom4j.Element;
import org.dom4j.io.SAXReader;
import java.io.IOException;
import java.io.InputStream;
import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.util.*;

public class MyApplicationContext {
```

```

private Map<String,String> properties = new HashMap<>();

private Map<String,Object> beans = new HashMap<>();

public MyApplicationContext(String path){

    InputStream in =
TestDom4j.class.getClassLoader().getResourceAsStream(path);

    //DOM4j 入口对象
    SAXReader reader = new SAXReader();

    //整个文档对象
    Document doc=null;

    try {

        doc = reader.read(in);

    } catch (DocumentException e) {

        e.printStackTrace();

    }

    //标签元素对象,根标签
    Element root = doc.getRootElement();

    //解析子标签
    List<Element> propertyElements = root.elements("property-placeholder");
    List<Element> beanElements = root.elements("bean");

    parsePropertyElements(propertyElements);
    parseBeanElements(beanElements);

}

private void parsePropertyElements(List<Element> propertyElements){

    Properties p = new Properties();

    ClassLoader classLoader = TestDom4j.class.getClassLoader();

```



```

        for (Element propertyElement : propertyElements) {

            String location =
propertyElement.attributeValue("location");//jdbc.properties

            InputStream in = classLoader.getResourceAsStream(location);

            try {

                p.load(in);

                Set<Object> objects = p.keySet();

                for (Object key : objects) {

                    String value = p.getProperty((String) key);

                    addProperty((String)key,value);

                }

            } catch (IOException e) {

                e.printStackTrace();

            }

        }

    }

    private void parseBeanElements(List<Element> beanElements) {

        for (Element beanElement : beanElements) {

            String id = beanElement.attributeValue("id");

            String aClass = beanElement.attributeValue("class");

            String initMethod = beanElement.attributeValue("init-method");

            //1.初始化 bean 对象

            Object bean = newInstance(aClass);

            //2.给对象中属性赋值(解析 property 标签)

            parseProperty(beanElement,bean);

            //3.执行 init-method 方法

            initBeanMethod(bean,initMethod);
        }
    }

```

```

        //4.注册 bean

        addBean(id,bean);

    }

}

private void initBeanMethod(Object bean,String initmethodName){

    try {

        Method method = bean.getClass().getMethod(initmethodName);

        method.invoke(bean);

    } catch (NoSuchMethodException e) {

        e.printStackTrace();

    } catch (IllegalAccessException e) {

        e.printStackTrace();

    } catch (InvocationTargetException e) {

        e.printStackTrace();

    }

}

/**
 * 解析 property 标签
 * @param beanElement
 * @param bean
 */
private void parseProperty(Element beanElement,Object bean){

    List<Element> property = beanElement.elements("property");

    for (Element element : property) {

        String propertyName = element.attributeValue("name");//bean 中的属性名

        String value = element.attributeValue("value");//值的表达式${}
    
```

```

        if(value.startsWith("${") && value.endsWith("}")){

            String propertyKey = value.substring(2,value.length()-1);

            //给 bean 对象中的属性名,赋值

            value = getProperty(propertyKey);

        }

        setPropertyValue2Bean(bean,propertyName,value);

    }

}

/**
 * 给 bean 对象中某属性赋值
 * @param bean 对象
 * @param name 属性名
 * @param value 属性值
 */
private void setPropertyValue2Bean(Object bean,String name,String value){

    Class<?> aClass = bean.getClass();

    //获取 propertyName 的属性类型
    Class<?> fieldType = null;

    try {

        Field field = aClass.getField(name);

        fieldType = field.getType();

    } catch (NoSuchFieldException e) {

        e.printStackTrace();

    }

    //拼接对应 set 方法名

    String setMethodName =

    "set"+name.substring(0,1).toUpperCase()+name.substring(1);

```

```
//调用 set 方法进行赋值

try {

    Method method = aClass.getMethod(setMethodName, fieldType);

    method.invoke(bean,value);

} catch (NoSuchMethodException e) {

    e.printStackTrace();

} catch (IllegalAccessException e) {

    e.printStackTrace();

} catch (InvocationTargetException e) {

    e.printStackTrace();

}

}
```

//初始化 bean 对象

```
private Object newInstance( String aClass){

    try {

        Class<?> c = Class.forName(aClass);

        return c.newInstance();

    } catch (ClassNotFoundException e) {

        e.printStackTrace();

    } catch (IllegalAccessException e) {

        e.printStackTrace();

    } catch (InstantiationException e) {

        e.printStackTrace();

    }

    return null;

}
```

```

public void addProperty(String key,String value){

    properties.put(key,value);

}

public void addBean(String key,Object bean){

    beans.put(key,bean);

}


public String getProperty(String key)

{

    return properties.get(key);

}

public Object getBean(String id){

    return beans.get(id);

}

public <T> T getBean(String id,Class<T> clz){

    return (T)beans.get(id);

}

public <T> T getBean(Class<T> clz){

    Collection<Object> values = beans.values();

    for (Object value : values) {

        if(clz.isAssignableFrom(value.getClass())){

            return (T)value;

        }

    }

    return null;

}

}

```

3.3.3. 测试

```
public static void main(String[] args) {  
    MyApplicationContext ac = new MyApplicationContext("spring.xml");  
    DataSource bean = ac.getBean(DataSource.class);  
    System.out.println(bean);  
}
```