34 SpEL List和Map 引用应用示例及背后原理探究

更新时间：2020-08-06 10:33:34

> 读书给人以快乐、给人以光彩、给人以才干。——培根

背景

使用 XML 的方式进行 Spring 配置，对于内部元素为 String 的 List 和 Map 属性的注入一般为如下方式：

```
<bean id = "testBean" class = "com.a.b.c.TestBean">
    <property name = "fieldMap">
        <map>
            <entry key = "field1" value = "value1"></entry>
            <entry key = "field2" value = "value2"></entry>
            <entry key = "field3" value = "value3"></entry>
        </map>
    </property>
    <property name = "fieldList">
        <list>
            <value>1</value>
            <value>2</value>
        </list>
    </property>
</bean>
```

那么，如何在一个 Bean 中引用另外一个 Bean 中的 map 和 list 属性的其中一个值呢？正如 SpEL 表达式所表现的动态性，可以这样做到：

```xml
<?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"
    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">

        <bean id="customerBean" class="com.davidwang456.test.Customer">
            <property name="mapA" value="#{testBean.map['MapA']}" />
            <property name="list" value="#{testBean.list[0]}" />
        </bean>

        <bean id="testBean" class="com.davidwang456.test.Test" />
</beans>
```

想知道上面配置实现的背后的原理吗？让我们一起动手来看看吧！

SpEL List 和 Map 引用示例

想要了解内部的原理，就需要 debug，想要 debug，就需要一个最简单的示例程序：

引用测试 bean 的类：

```java
package com.davidwang456.test;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import org.springframework.stereotype.Component;

@Component("testBean")
public class Test {

    private Map<String, String> map;
    private List<String> list;

    public Test() {
        map = new HashMap<String, String>();
        map.put("MapA", "This is A");
        map.put("MapB", "This is B");
        map.put("MapC", "This is C");

        list = new ArrayList<String>();
        list.add("List0");
        list.add("List1");
        list.add("List2");

    }

    public Map<String, String> getMap() {
        return map;
    }

    public void setMap(Map<String, String> map) {
        this.map = map;
    }

    public List<String> getList() {
        return list;
    }

    public void setList(List<String> list) {
        this.list = list;
    }

}
```

引用 bean 的类：

```java
package com.davidwang456.test;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

@Component("customerBean")
public class Customer {

    @Value("#{testBean.map['MapA']}")
    private String mapA;

    @Value("#{testBean.list[0]}")
    private String list;

    public String getMapA() {
        return mapA;
    }

    public void setMapA(String mapA) {
        this.mapA = mapA;
    }

    public String getList() {
        return list;
    }

    public void setList(String list) {
        this.list = list;
    }

    @Override
    public String toString() {
        return "Customer [mapA=" + mapA + ", list=" + list + "]";
    }

}
```

配置文件，在 classpath 路线下：

```xml
<?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"
    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.davidwang456.test" />
</beans>
```

测试类：

```java
package com.davidwang456.test;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class AppMain {

    @SuppressWarnings("resource")
    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext("springbeans.xml");

        Customer obj = (Customer) context.getBean("customerBean");
        System.out.println(obj);
    }
}
```

测试结果：

> Customer [mapA=This is A, list=List0]

SpEL list 和 Map 引用背后原理探秘

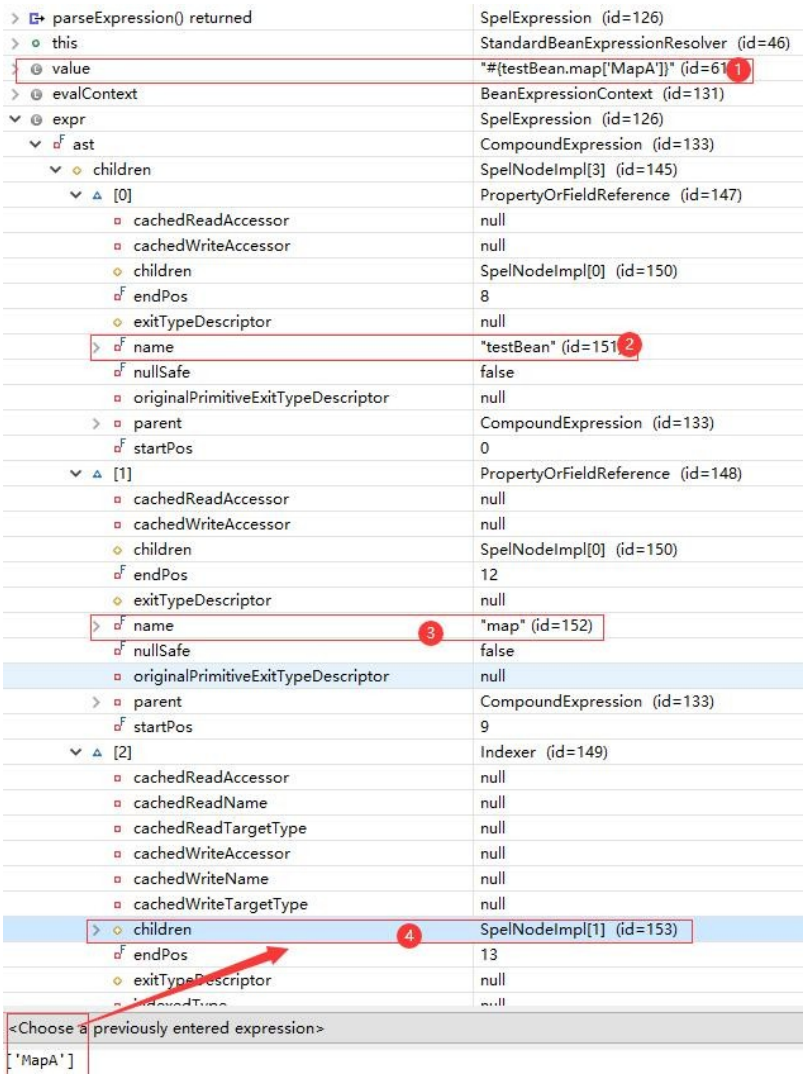
通过洋像葱剥皮一样一层层的深入 debug，重要找到核心层。

主链路：生成 AST 树

在 StandardBeanExpressionResolver.java#evaluate()，为什么呢？在这个方法里面将表达式转化为 AST：

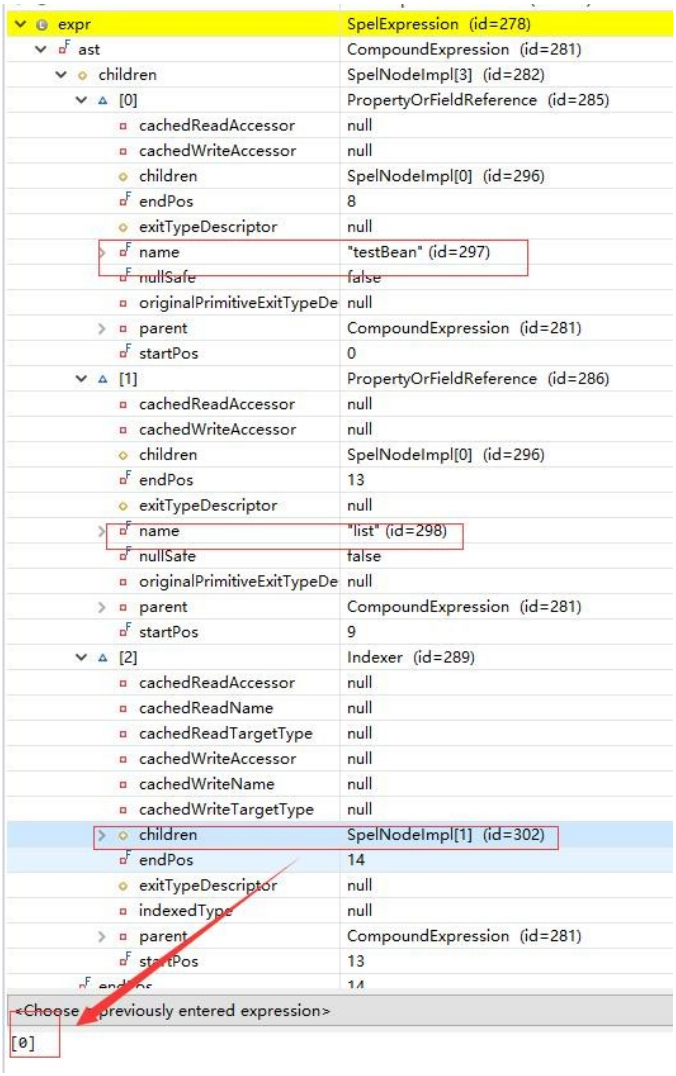
```
@Override
@Nullable
public Object evaluate(@Nullable String value, BeanExpressionContext evalContext) throws BeansException {
    if (!StringUtils.hasLength(value)) {
        return value;
    }
    try {
        Expression expr = this.expressionCache.get(value);
        if (expr == null) {
            expr = this.expressionParser.parseExpression(value, this.beanExpressionParserContext);
            this.expressionCache.put(value, expr);
        }
        StandardEvaluationContext sec = this.evaluationCache.get(evalContext);
        if (sec == null) {
            sec = new StandardEvaluationContext(evalContext);
            sec.addPropertyAccessor(new BeanExpressionContextAccessor());
            sec.addPropertyAccessor(new BeanFactoryAccessor());
            sec.addPropertyAccessor(new MapAccessor());
            sec.addPropertyAccessor(new EnvironmentAccessor());
            sec.setBeanResolver(new BeanFactoryResolver(evalContext.getBeanFactory()));
            sec.setTypeLocator(new StandardTypeLocator(evalContext.getBeanFactory().getBeanClassLoader()));
            ConversionService conversionService = evalContext.getBeanFactory().getConversionService();
            if (conversionService != null) {
                sec.setTypeConverter(new StandardTypeConverter(conversionService));
            }
            customizeEvaluationContext(sec);
            this.evaluationCache.put(evalContext, sec);
        }
        return expr.getValue(sec);
    }
    catch (Throwable ex) {
        throw new BeanExpressionException("Expression parsing failed", ex);
    }
}
```

Map 的表达式“#{testBean.map['MapA']}”解析成 AST 的结构如下图所示：



它分成三层，从最顶层的 testBean，到中间层 map，到最底层的 SpelNodeImpl 实现。

同样 List 的表达式“#{testBean.list[0]}”生成的 AST 结构如下图所示：



同样，它也分成三层，从最顶层的 testBean，到中间层 list，到最底层的 SpelNodeImpl 实现。到了这里，其实还可以接着深入进去，

为了方便，可以通过程序抛出异常的方式来查看完整的调用链路。

完整链路：调用链

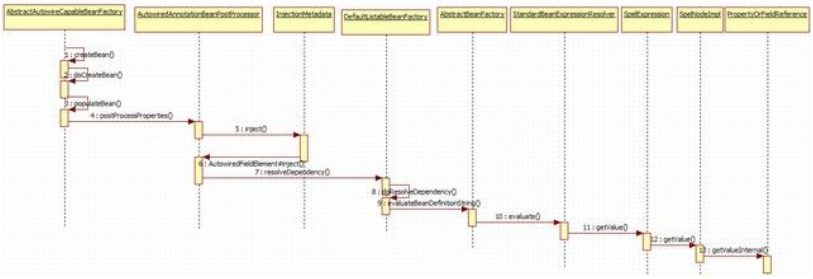
在 PropertyOrFieldReference 的调用链上抛出异常，抛出异常可以打印出到这里的整个调用链。

```
@Override
public TypedValue getValueInternal(ExpressionState state) throws EvaluationException {
    TypedValue tv = getValueInternal(state.getActiveContextObject(), state.getEvaluationContext(),
            state.getConfiguration().isAutoGrowNullReferences());
    PropertyAccessor accessorToUse = this.cachedReadAccessor;
    if (accessorToUse instanceof CompilablePropertyAccessor) {
        CompilablePropertyAccessor accessor = (CompilablePropertyAccessor) accessorToUse;
        setExitTypeDescriptor(CodeFlow.toDescriptor(accessor.getPropertyType()));
    }
    throw new NullPointerException();
    //return tv;
}
```

结果如下：

> 15:02:53.370 [main] WARN o.s.c.s.ClassPathXmlApplicationContext - Exception encountered during context initialization - cancelling refresh attempt: org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name ‘customerBean’: Unsatisfied dependency expressed through field ‘mapA’; nested exception is org.springframework.beans.factory.BeanExpressionException: Expression parsing failed; nested exception is java.lang.NullPointerException
>
> …
>
> …
>
> at com.davidwang456.test.AppMain.main(AppMain.java:11)
>
> …
>
> … 19 more

整理出时序图如下：



其中，StandardBeanExpressionResolver 包装了表达式解析器 ExpressionParser，它的最终实现是**语法分析器 SpelExpressionParser**，词法分析器 **SpelExpressionParser**将可分析的词组成 AST 。然后顺序执行。

总结

Spring3.x 引入的 SpEL 可谓非常的惊艳，它的实现非常的复杂，但它的使用却异常的简单和灵活。它给 Spring 外部化配置注入了更多的活力，它让我们在运行时赋值、改变值都轻松的成为了可能。

}

33 SpEL String应用示例及背后原理探究

35 SpEL 操作符应用示例及背后原理探究