|  |
| --- |
| Spring |
| <bean id=*"T1000"* class=*"ru.javabegin.training.spring.impls.robot.ModelT1000"*> |
| <constructor-arg ref=*"SonyHand"* /> |
| <property name=*"hand"* ref=*"SonyHand"*/>  <property name=*"year"* value=*"120"*></property>  <bean id="theTargetBean" class="..."/>  <bean id="theClientBean" class="...">  <property name="targetName">  <idref bean="theTargetBean" />  </property>  </bean> |
| <constructor-arg value=*"siver"* /> |
| <constructor-arg value=*"siver"* index=*"0"* /> |
| <constructor-arg value=*"true"* index=*"2"* type=*"boolean"* /> |
| P:Namespace  xmlns:p=[*http://www.springframework.org/schema/p*](http://www.springframework.org/schema/p)  <bean id=*"T1000"* class=*"ru.javabegin.training.spring.impls.robot.ModelT1000"* p:hand-ref=*"SonyHand"*>  <!-- <property name="hand" ref="ToshibaHand"></property> -->  </bean>  <bean id=*"SonyHand"* class=*"ru.javabegin.training.spring.impls.sony.SonyHand"* />  public void setHand(Hand hand) {  this.hand = hand;  } |
| ALIAS  <bean id=*"SonyHand"* class=*"ru.javabegin.training.spring.impls.sony.SonyHand"* />  <alias name=*"SonyHand"* alias=*"SonyHand2"*></alias>  <constructor-arg ref=*"SonyHand2"* index=*"1"*/> |
| Factory Method to instance Object |
|  |
| FactoryMethod + Constructer  <bean id="exampleBean" class="examples.ExampleBean"  factory-method="createInstance">  <constructor-arg ref="anotherExampleBean"/>  <constructor-arg ref="yetAnotherBean"/>  <constructor-arg value="1"/>  </bean>  <bean id="anotherExampleBean" class="examples.AnotherBean"/>  <bean id="yetAnotherBean" class="examples.YetAnotherBean"/>  public class ExampleBean {  *// a private constructor*  private ExampleBean(...) {  ...  }    *// a static factory method; the arguments to this method can be*  *// considered the dependencies of the bean that is returned,*  *// regardless of how those arguments are actually used.*  public static ExampleBean createInstance (  AnotherBean anotherBean, YetAnotherBean yetAnotherBean, int i) {  ExampleBean eb = new ExampleBean (...);  *// some other operations...*  return eb;  }  } |
| C:namespace |
| Man kann den Namen für den Construcor parameter selbst bestimmen  <bean id="exampleBean" class="examples.ExampleBean">  <constructor-arg name="years1" value="7500000"/>  <constructor-arg name="ultimateanswer" value="42"/>  </bean>  package examples;  public class ExampleBean {  *// Fields omitted*  @ConstructorProperties({"years1", "ultimateAnswer"})  public ExampleBean(int years, String ultimateAnswer) {  this.years = years;  this.ultimateAnswer = ultimateAnswer;  }  } |
| java.util.Properties  <bean id="mappings"  class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">  <*!-- typed as a java.util.Properties --*>  <property name="properties">  <value>  jdbc.driver.className=com.mysql.jdbc.Driver  jdbc.url=jdbc:mysql://localhost:3306/mydb  </value>  </property>  </bean> |
| LIST, MAP, PROP..  <bean id="moreComplexObject" class="example.ComplexObject">  <*!-- results in a setAdminEmails(java.util.Properties) call --*>  <property name="adminEmails">  <props>  <prop key="administrator">administrator@example.org</prop>  <prop key="support">support@example.org</prop>  <prop key="development">development@example.org</prop>  </props>  </property>  <*!-- results in a setSomeList(java.util.List) call --*>  <property name="someList">  <list>  <value>a list element followed by a reference</value>  <ref bean="myDataSource" />  </list>  </property>  <*!-- results in a setSomeMap(java.util.Map) call --*>  <property name="someMap">  <map>  <entry key="an entry" value="just some string"/>  <entry key ="a ref" value-ref="myDataSource"/>  </map>  </property>  <*!-- results in a setSomeSet(java.util.Set) call --*>  <property name="someSet">  <set>  <value>just some string</value>  <ref bean="myDataSource" />  </set>  </property>  </bean> |
| The following example demonstrates collection merging:  <beans>  <bean id="parent" abstract="true" class="example.ComplexObject">  <property name="adminEmails">  <props>  <prop key="administrator">administrator@example.com</prop>  <prop key="support">support@example.com</prop>  </props>  </property>  </bean>  <bean id="child" parent="parent">  <property name="adminEmails">  <*!-- the merge is specified on the \*child\* collection definition --*>  <props merge="true">  <prop key="sales">sales@example.com</prop>  <prop key="support">support@example.co.uk</prop>  </props>  </property>  </bean>  <beans> |
| Depends-On  <bean id="beanOne" class="ExampleBean" depends-on="manager,accountDao">  <property name="manager" ref="manager" />  </bean>  <bean id="manager" class="ManagerBean" />  <bean id="accountDao" class="x.y.jdbc.JdbcAccountDao" /> |
| Value  <property name="myProperty">  <value>hello</value>  </property>  <constructor-arg>  <value>hello</value>  </constructor-arg>  <entry key="myKey">  <value>hello</value>  </entry>  are equivalent to:  <property name="myProperty" value="hello"/>  <constructor-arg value="hello"/>  <entry key="myKey" value="hello"/> |
| Ref  <property name="myProperty">  <ref bean="myBean">  </property>  <constructor-arg>  <ref bean="myBean">  </constructor-arg>  ... are equivalent to:  <property name="myProperty" ref="myBean"/>  <constructor-arg ref="myBean"/> |
| Lazily-instantiated beans  <bean id="lazy" class="com.foo.ExpensiveToCreateBean" lazy-init="true"/>  <bean name="not.lazy" class="com.foo.AnotherBean"/>  <beans default-lazy-init="true">  <*!-- no beans will be pre-instantiated... --*>  </beans> |
| Scope  <bean id=*"t1000\_v3"* class=*"ru.javabegin.training.spring.impls.robot.ModelT1000"* scope=*"prototype"*> </bean> |
| Init Destroy-Method  <bean id=*"T1000"* class=*"ru.javabegin.training.spring.impls.robot.ModelT1000"*  scope=*"prototype"* init-method=*"initObject"* destroy-method=*"destroyObject"*>  </bean>  oder  default-init-method=*"initObject"*  oder  inerface InitializingBean, DisposableBean RobotSpring\_10a  <bean class=*"ru.javabegin.training.spring.main.PostProzessor"* ></bean>  @Override  public Object postProcessBeforeInitialization(Object bean, String beanName) throws BeansException {  System.*out*.println(bean.getClass().getSimpleName());  return bean;  }  oder  public class PostProzessor implements BeanPostProcessor{}  <bean class=*"ru.javabegin.training.spring.main.PostProzessor"*></bean> |
| Import  <import resource=*"sony.xml"*/>  <import resource=*"toshiba.xml"*/> |
| DependsOn  depends-on=*"database"* |
| Parent  <bean id=*"BaseModel"* abstract=*"true"*>  <constructor-arg ref=*"SonyHand"* index=*"0"* />  <constructor-arg ref=*"ToshibaHead"* index=*"1"* />  <constructor-arg ref=*"SonyLeg"* index=*"2"* />  </bean>  <bean id=*"T1000"* class=*"ru.javabegin.training.spring.impls.robot.ModelT1000"*  parent=*"BaseModel"*>  <constructor-arg value=*"true"* index=*"3"* type=*"boolean"* />  <constructor-arg value=*"silver"* index=*"4"* type=*"String"* />  <constructor-arg value=*"12"* index=*"5"* type=*"int"* />  </bean> |
| Replaced-Method  public class ReplacedM implements MethodReplacer {  @Override  public Object reimplement(Object arg0, Method arg1, Object[] arg2) throws Throwable {    System.*out*.println(arg2);  return "hallo";  }  }  <bean id=*"T1000"* class=*"ru.javabegin.training.spring.impls.robot.ModelT1000"*  autowire=*"byType"* scope=*"prototype"*>  <replaced-method name=*"dance"* replacer=*"replacementTarget"*/>  </bean>  <bean id=*"replacementTarget"* class=*"ru.javabegin.training.spring.main.ReplacedM"*>  </bean> |
| Lookup-Method  public interface IRobotConveyer { //kann auch class oder abstract class sein    public Robot getRobot();  }  <bean id=*"RobotConveyer"* class=*"ru.javabegin.training.spring.main.IRobotConveyer"*>  <lookup-method name=*"getRobot"* bean=*"T1000"* />  </bean> |
|  |