Spring

- Spring is a lightweight framework
- It is framework of frameworks because it provides support to various frameworks.
- The framework comprises several modules such as IOC, AOP, Context, ORM, WEB MVC etc

Advantages of Spring:

- Provides predefined templates
- Ensures Loose coupling
- Light weight
- Easy to Test
- Fast & quicker coding on transactions

IOC Container

- Responsible to instantiate, configure and assemble the objects
- Gets information from the XML file and processes as needed
- Main tasks:
  - to instantiate the application class
  - to configure the object
  - to assemble the dependencies between the objects

How to create Spring Application?

Download Spring Jar files & include in the Java Program

Example 1: (basic Spring program)

Student.java

```
package e1_SpringBasics;

public class Student {

    private String name;
```

```
    public String getName(){
        return name;
    }

    public void setName(String name){
        this.name = name;
    }

    public void displayInfo(){
        System.out.println("Hello " + name);
    }
}
```

ApplicationContext.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/schem
a/context"


xsi:schemaLocation="http://www.springframework.org/
schema/beans

http://www.springframework.org/schema/beans/spring-
beans-3.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/sprin
g-context.xsd">
```

```
    <bean id="studentbean"
class="e1_SpringBasics.Student">
        <property name="name" value=
"John"></property>
    </bean>
</beans>
```

Test.java

```
package e1_SpringBasics;

import
org.springframework.beans.factory.BeanFactory;
import
org.springframework.beans.factory.xml.XmlBeanFactor
y;
import
org.springframework.core.io.ClassPathResource;
import org.springframework.core.io.Resource;

public class Test {

    public static void main(String[] args) {

        Resource resource = new
ClassPathResource("applicationContext.xml");
        BeanFactory factory = new
XmlBeanFactory(resource);

        Student student =
(Student)factory.getBean("studentbean");
        student.displayInfo();

    }

}
```

Dependency Injection

Dependency Injection (DI) is a design pattern that removes the dependency from the programming code so that it can be easy to manage and test the application. Dependency Injection makes our programming code loosely coupled.
we provide the information from the external source such as XML file

Two ways to perform Dependency Injection in Spring framework

- By Constructor
- By Setter method

Constructor Injection:

We can inject the dependency by constructor. The **<constructor-arg>** subelement of **<bean>** is used for constructor injection. Here we are going to inject

1. primitive and String-based values
2. Dependent object (contained object)
3. Collection values etc.

Example 1 (Primitive & String Values)

Student.java

```
package e2_DI_ConstructorInjection;

public class Student {

    private int id;
    private String name;

    public Student(){
        System.out.println("No Data");
    }

    public Student(int id){
        this.id = id;
    }

    public Student(String name){
```

```
        this.name = name;
    }

    public Student(int id, String name){
        this.id = id;
        this.name = name;
    }

    public void displayInfo(){
        System.out.println(id + " " + name);
    }
}
```

ApplicationContext.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/schem
a/context"


xsi:schemaLocation="http://www.springframework.org/
schema/beans

http://www.springframework.org/schema/beans/spring-
beans-3.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/sprin
g-context.xsd">
```

```
    <bean id="studentbean"
class="e2_DI_ConstructorInjection.Student">
        <constructor-arg value="101"
type="int"></constructor-arg>
        <constructor-arg
value="Jack"></constructor-arg>
    </bean>
</beans>
```

Test.java

```
package e2_DI_ConstructorInjection;

import
org.springframework.beans.factory.BeanFactory;
import
org.springframework.beans.factory.xml.XmlBeanFactor
y;
import
org.springframework.core.io.ClassPathResource;
import org.springframework.core.io.Resource;

public class Test {

    public static void main(String[] args) {

        Resource resource = new
ClassPathResource("applicationContext.xml");
        BeanFactory factory = new
XmlBeanFactory(resource);

        Student student =
(Student)factory.getBean("studentbean");
        student.displayInfo();

    }
```

```
}
```

Example 2 (Dependent Object)

Education.java

```
package e2_DI_ConstructorInjection;

public class Education {

    private String degree;
    private String course;
    private String subject;

    public Education(String degree, String course,
String subject){
        super();
        this.degree = degree;
        this.course = course;
        this.subject = subject;
    }

    public String show(){
        return degree + " " + course + " " +
subject;
    }
}
```

Student.java

```
package e2_DI_ConstructorInjection;

public class Student {

    private int id;
    private String name;
    private Education education;

    public Student(){
```

```
            System.out.println("No Data");
        }

        public Student(int id, String name, Education
education){
            this.id = id;
            this.name = name;
            this.education = education;
        }

        public void displayInfo(){
            System.out.println(id + " " + name);
            System.out.println(education.show());
        }
}
```

ApplicationContext.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/schem
a/context"


xsi:schemaLocation="http://www.springframework.org/
schema/beans

http://www.springframework.org/schema/beans/spring-
beans-3.0.xsd

http://www.springframework.org/schema/context
```

```
http://www.springframework.org/schema/context/spring-context.xsd">

    <bean id="e1" class="e2_DI_ConstructorInjection.Education">
        <constructor-arg value="UG"></constructor-arg>
        <constructor-arg value="BE"></constructor-arg>
        <constructor-arg value="Mech"></constructor-arg>
    </bean>

    <bean id="studentbean" class="e2_DI_ConstructorInjection.Student">
        <constructor-arg value="101" type="int"></constructor-arg>
        <constructor-arg value="Jack"></constructor-arg>
        <constructor-arg>
            <ref bean = "e1"/>
        </constructor-arg>
    </bean>
</beans>
```

Test.java

```
package e2_DI_ConstructorInjection;

import org.springframework.beans.factory.BeanFactory;
import org.springframework.beans.factory.xml.XmlBeanFactory;
import org.springframework.core.io.ClassPathResource;
```

```
import org.springframework.core.io.Resource;

public class Test {

    public static void main(String[] args) {

        Resource resource = new
ClassPathResource("applicationContext.xml");
        BeanFactory factory = new
XmlBeanFactory(resource);

        Student student =
(Student)factory.getBean("studentbean");
        student.displayInfo();

    }

}
```

Example 3 (Collections)

Student.java

```
package e2_DI_ConstructorInjection;

import java.util.Iterator;
import java.util.List;

public class Student {

    private int id;
    private String name;
    private List<String> allied;

    public Student(){
        System.out.println("No Data");
    }
```

```
    public Student(int id, String name,
List<String> allied){
        this.id = id;
        this.name = name;
        this.allied = allied;
    }

    public void displayInfo(){
        System.out.println(id + " " + name);
        System.out.println("Allied subjects are
:");
        Iterator<String> itr = allied.iterator();
        while(itr.hasNext()){
            System.out.println(itr.next());
        }
    }
}
```

ApplicationContext.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/schem
a/context"


xsi:schemaLocation="http://www.springframework.org/
schema/beans

http://www.springframework.org/schema/beans/spring-
beans-3.0.xsd
```

```
http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context.xsd">

    <bean id="studentbean"
class="e2_DI_ConstructorInjection.Student">
        <constructor-arg value="101"
type="int"></constructor-arg>
        <constructor-arg
value="Jack"></constructor-arg>
        <constructor-arg>
            <list>
                <value>Commerce</value>
                <value>Economics</value>
                <value>History</value>
            </list>
        </constructor-arg>
    </bean>
</beans>
```

Test.java

```
package e2_DI_ConstructorInjection;

import
org.springframework.beans.factory.BeanFactory;
import
org.springframework.beans.factory.xml.XmlBeanFactor
y;
import
org.springframework.core.io.ClassPathResource;
import org.springframework.core.io.Resource;

public class Test {
```

```
    public static void main(String[] args) {

        Resource resource = new
ClassPathResource("applicationContext.xml");
        BeanFactory factory = new
XmlBeanFactory(resource);

        Student student =
(Student)factory.getBean("studentbean");
        student.displayInfo();

    }

}
```

Setter Injection:

We can inject the dependency by setter method also. The **<property>** subelement of **<bean>** is used for setter injection. Here we are going to inject

1. primitive and String-based values
2. Dependent object (contained object)
3. Collection values etc.

Example 1 (Primitive Data)

Student.java

```
package e3_DI_SetterInjection;

public class Student {

    private int id;
    private String name;

    public int getId(){
        return id;
    }
    public void setId(int id){
        this.id = id;
```

```
    }

    public String getName(){
        return name;
    }
    public void setName(String name){
        this.name = name;
    }

    public void displayInfo(){
        System.out.println(id + " " + name);
    }
}
```

ApplicationContext.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/schem
a/context"


xsi:schemaLocation="http://www.springframework.org/
schema/beans

http://www.springframework.org/schema/beans/spring-
beans-3.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/sprin
g-context.xsd">
```

```
    <bean id="studentbean"
class="e3_DI_SetterInjection.Student">
        <property name="id">
            <value>201</value>
        </property>
        <property name="name">
            <value>Jim</value>
        </property>
    </bean>
</beans>
```

Test.java

```
package e3_DI_SetterInjection;

import
org.springframework.beans.factory.BeanFactory;
import
org.springframework.beans.factory.xml.XmlBeanFactor
y;
import
org.springframework.core.io.ClassPathResource;
import org.springframework.core.io.Resource;

public class Test {

    public static void main(String[] args) {

        Resource resource = new
ClassPathResource("applicationContext.xml");
        BeanFactory factory = new
XmlBeanFactory(resource);

        Student student =
(Student)factory.getBean("studentbean");
        student.displayInfo();
```

```
    }

}
```

Example 2 (Dependent Object)

Education.java

```
package e3_DI_SetterInjection;

public class Education {

    private String degree;
    private String course;
    private String subject;

    public String getDegree(){
        return degree;
    }
    public void setDegree(String degree){
        this.degree = degree;
    }

    public String getCourse(){
        return course;
    }
    public void setCourse(String course){
        this.course = course;
    }

    public String getSubject(){
        return subject;
    }
    public void setSubject(String subject){
        this.subject = subject;
    }

    public String show(){
```

```
        return degree + " " + course + " " +
subject;
    }
}
```

Student.java

```
package e3_DI_SetterInjection;

public class Student {

    private int id;
    private String name;
    private Education education;

    public int getId(){
        return id;
    }
    public void setId(int id){
        this.id = id;
    }

    public String getName(){
        return name;
    }
    public void setName(String name){
        this.name = name;
    }

    public Education getEducation(){
        return education;
    }
    public void setEducation(Education education){
        this.education = education;
    }

    public void displayInfo(){
```

```
        System.out.println(id + " " + name);
        System.out.println(education.show());
    }
}
```

ApplicationContext.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/schem
a/context"


xsi:schemaLocation="http://www.springframework.org/
schema/beans

http://www.springframework.org/schema/beans/spring-
beans-3.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/sprin
g-context.xsd">

    <bean id="e1"
class="e3_DI_SetterInjection.Education">
        <property name="degree"
value="PG"></property>
        <property name="course"
value="MTech"></property>
        <property name="subject"
value="Civil"></property>
```

```
    </bean>

    <bean id="studentbean"
class="e3_DI_SetterInjection.Student">
        <property name="id">
            <value>201</value>
        </property>
        <property name="name">
            <value>Jim</value>
        </property>
        <property name="education"
ref="e1"></property>
    </bean>
</beans>
```

Test.java

```
package e3_DI_SetterInjection;

import
org.springframework.beans.factory.BeanFactory;
import
org.springframework.beans.factory.xml.XmlBeanFactor
y;
import
org.springframework.core.io.ClassPathResource;
import org.springframework.core.io.Resource;

public class Test {

    public static void main(String[] args) {

        Resource resource = new
ClassPathResource("applicationContext.xml");
        BeanFactory factory = new
XmlBeanFactory(resource);
```

```
        Student student =
(Student)factory.getBean("studentbean");
        student.displayInfo();

    }

}
```

Example 3 (Collections)

Student.java

```
package e3_DI_SetterInjection;

import java.util.Iterator;
import java.util.List;

public class Student {

    private int id;
    private String name;
    private List<String> allied;

    public int getId(){
        return id;
    }
    public void setId(int id){
        this.id = id;
    }

    public String getName(){
        return name;
    }
    public void setName(String name){
        this.name = name;
    }
```

```
    public List<String> getAllied(){
        return allied;
    }
    public void setAllied(List<String> allied){
        this.allied = allied;
    }

    public void displayInfo(){
        System.out.println(id + " " + name);
        System.out.println("Allied subjects are
:");
        Iterator<String> itr = allied.iterator();
        while(itr.hasNext()){
            System.out.println(itr.next());
        }
    }
}
```

ApplicationContext.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/schem
a/context"


xsi:schemaLocation="http://www.springframework.org/
schema/beans

http://www.springframework.org/schema/beans/spring-
beans-3.0.xsd
```

```
http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context.xsd">

    <bean id="studentbean"
class="e3_DI_SetterInjection.Student">
        <property name="id">
            <value>201</value>
        </property>
        <property name="name">
            <value>Jim</value>
        </property>
        <property name="allied">
            <list>
                <value>English</value>
                <value>Maths</value>
                <value>Sociology</value>
            </list>
        </property>
    </bean>
</beans>
```

Test.java

```
package e3_DI_SetterInjection;

import
org.springframework.beans.factory.BeanFactory;
import
org.springframework.beans.factory.xml.XmlBeanFactor
y;
import
org.springframework.core.io.ClassPathResource;
import org.springframework.core.io.Resource;
```

```
public class Test {

    public static void main(String[] args) {

        Resource resource = new
ClassPathResource("applicationContext.xml");
        BeanFactory factory = new
XmlBeanFactory(resource);

        Student student =
(Student)factory.getBean("studentbean");
        student.displayInfo();

    }

}
```

Differences between CI & SI:

	CI	SI
Partial Dependency	Not possible	Possible
Overriding	Not possible	SI overrides CI
Changes	Not possible	It can easily change value to properties

Autowiring:

- Inject the object dependency implicitly, means internally using setter or constructor injection
- It can't be used with primitive & string values
- Use "autowire" attribute of bean element to apply autowire with different modes

Modes:

1. byName – injects object dependency according to name of bean; internally uses setter method
2. byType – injects object dependency according to type; internally uses setter method
3. constructor – inject dependency by calling constructor
4. no – default mode; means no autowiring included

Example 1 (byName):

Course.java:

```
package e4_Autowiring;

public class Course {

    Course(){
        System.out.println("Student Course
Registered");
    }

    void displayInfo(){
        System.out.println("Course is BTech");
    }
}
```

Student.java:

```
package e4_Autowiring;

public class Student {

    Course course;

    Student(){
        System.out.println("Student Profile
Created");
    }

    public Course getCourse(){
        return course;
    }

    public void setCourse(Course course){
        this.course = course;
    }
```

```
    void displayInfo(){
        System.out.println("Student name is John");
    }

    void displayProfile(){
        displayInfo();
        course.displayInfo();
    }
}
```

ApplicationContext.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/schem
a/context"


xsi:schemaLocation="http://www.springframework.org/
schema/beans

http://www.springframework.org/schema/beans/spring-
beans-3.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/sprin
g-context.xsd">

    <bean id="course"
class="e4_Autowiring.Course"></bean>
```

```
    <bean id="student"
class="e4_Autowiring.Student"
autowire="byName"></bean>
</beans>
```

Test:

```
package e4_Autowiring;

import
org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApp
licationContext;

public class Test {

    public static void main(String[] args) {

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

        Student student =
context.getBean("student", Student.class);
        student.displayProfile();;

    }

}
```

Example 2 (byType):

Course.java:

```
package e4_Autowiring;

public class Course {
```

```
    Course(){
        System.out.println("Student Course
Registered");
    }

    void displayInfo(){
        System.out.println("Course is BTech");
    }
}
```

Student.java:

```
package e4_Autowiring;

public class Student {

    Course course;

    Student(){
        System.out.println("Student Profile
Created");
    }

    public Course getCourse(){
        return course;
    }

    public void setCourse(Course course){
        this.course = course;
    }

    void displayInfo(){
        System.out.println("Student name is John");
    }

    void displayProfile(){
```

```
        displayInfo();
        course.displayInfo();
    }
}
```

ApplicationContext.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/schem
a/context"


xsi:schemaLocation="http://www.springframework.org/
schema/beans

http://www.springframework.org/schema/beans/spring-
beans-3.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/sprin
g-context.xsd">

    <bean id="course1"
class="e4_Autowiring.Course"></bean>
    <bean id="student"
class="e4_Autowiring.Student"
autowire="byType"></bean>
</beans>
```

Test:

```
package e4_Autowiring;

import
org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApp
licationContext;

public class Test {

    public static void main(String[] args) {

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

        Student student =
context.getBean("student", Student.class);
        student.displayProfile();

    }

}
```

Exercise:

Do with autowire modes for default, no & constructor

Stereotype Annotations:

- Beans has to be defined so the container is aware & will do autowiring
- Container will inject dependency with beans
- Basic annotations include @Component, @Repository, @Service, @Controller, etc
- Stereotype annotations are intended to mark different layers in the multi-tier application
- Annotations mark the business, presentation, persistence, etc respectively for @Service, @Component, @Repository, etc

Spring Application work with Annotations:

- Spring will automatically import the beans into the container & inject dependencies
- The container has to invoke component-scanning mechanism
- Enable component scanning in the XML file (include “context:component-scan” tag with the base package)
- It will scan & configure the bean only when the container does DI

Note:

- ✓ Always use the annotations over concrete classes, not over interface
- ✓ Once you include stereotype annotations, you can directly use bean references defined inside classes
- ✓ There are few annotations used for methods, but it functions differently

Example 1:

Student class:

```
package e5_StereotypeAnno1;

import org.springframework.stereotype.Component;

@Component
public class Student {

    private int id=111;
    private String firstName = "Jim";
    private String lastName = "Courier";

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }

    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
```

```
    }

    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    @Override
    public String toString() {
        return "Student [id=" + id + ", firstName="
+ firstName + ", lastName=" + lastName + "]";
    }

}
```

ApplicationContext.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/schem
a/context"


xsi:schemaLocation="http://www.springframework.org/
schema/beans

http://www.springframework.org/schema/beans/spring-
beans-3.0.xsd

http://www.springframework.org/schema/context
```

```
http://www.springframework.org/schema/context/spring-context.xsd">

    <context:component-scan base-package="e5_StereotypeAnno1"></context:component-scan>
</beans>
```

Test:

```
package e5_StereotypeAnno1;

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

public class Test {

    public static void main(String[] args) {

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

        Student student = (Student)context.getBean("student");
        System.out.println(student);
    }

}
```

Example 2 (with all annotations):

StudentDTO class

```
package e6_StereotypeAnno2;

public class StudentDTO {

	private Integer id;
	private String firstName;
	private String lastName;

	public Integer getId() {
		return id;
	}
	public void setId(Integer id) {
		this.id = id;
	}

	public String getFirstName() {
		return firstName;
	}
	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}

	public String getLastName() {
		return lastName;
	}
	public void setLastName(String lastName) {
		this.lastName = lastName;
	}

	@Override
	public String toString() {
		return "StudentDTO [id=" + id + ",
firstName=" + firstName + ", lastName=" + lastName
+ "]";
```

```
    }

}
```

StudentDAO Interface

```
package e6_StereotypeAnno2;

public interface StudentDAO {

    public StudentDTO createNewStudent();

}
```

StudentDAOImpl class

```
package e6_StereotypeAnno2;

import org.springframework.stereotype.Repository;

@Repository("studentDao")
public class StudentDAOImpl implements StudentDAO{

    public StudentDTO createNewStudent(){

        StudentDTO s = new StudentDTO();
        s.setId(222);
        s.setFirstName("Stefan");
        s.setLastName("Edberg");
        return s;
    }
}
```

StudentManager Interface

```
package e6_StereotypeAnno2;
```

```
public interface StudentManager {

    public StudentDTO createNewStudent();
}
```

StudentManagerImpl class

```
package e6_StereotypeAnno2;

import
org.springframework.beans.factory.annotation.Autowi
red;
import org.springframework.stereotype.Service;

@Service("studentManager")
public class StudentManagerImpl implements
StudentManager{

    @Autowired
    StudentDAO dao;

    public StudentDTO createNewStudent(){
        return dao.createNewStudent();
    }
}
```

StudentController class

```
package e6_StereotypeAnno2;

import
org.springframework.beans.factory.annotation.Autowi
red;
import org.springframework.stereotype.Controller;

@Controller("studentController")
public class StudentController {
```

```
    @Autowired
    StudentManager manager;

    public StudentDTO createNewStudent(){
        return manager.createNewStudent();
    }
}
```

ApplicationContext.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/schem
a/context"


xsi:schemaLocation="http://www.springframework.org/
schema/beans

http://www.springframework.org/schema/beans/spring-
beans-3.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/sprin
g-context.xsd">

    <context:component-scan base-
package="e6_StereotypeAnno2"></context:component-
scan>
</beans>
```

Test class:

```
package e6_StereotypeAnno2;

import
org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApp
licationContext;

public class Test {

    public static void main(String[] args) {

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

        StudentController controller =
(StudentController)context.getBean("studentControll
er");

    System.out.println(controller.createNewStudent(
));
    }

}
```

Execution & Workflow of above example: