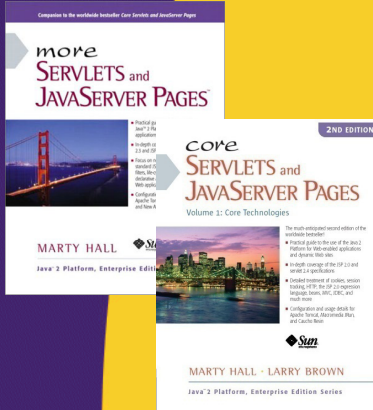




# Spring AOP Part 1

Originals of Slides and Source Code for Examples:  
<http://courses.coreservlets.com/Course-Materials/spring.html>

**Customized Java EE Training:** <http://courses.coreservlets.com/>  
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.



**For live Spring & Hibernate training, see  
courses at <http://courses.coreservlets.com/>.**



**Taught by the experts that brought you this tutorial.  
Available at public venues, or customized versions  
can be held on-site at your organization.**

- Courses developed and taught by Marty Hall
  - Java 5, Java 6, intermediate/beginning servlets/JSP, advanced servlets/JSP, Struts, JSF, Ajax, GWT, custom mix of topics
- Courses developed and taught by coreservlets.com experts (edited by Marty)
  - Spring, Hibernate/JPA, EJB3, Ruby/Rails

Contact [hall@coreservlets.com](mailto:hall@coreservlets.com) for details

# Topics in This Section

- **Aspect-oriented programming concepts**
- **Introduction**
- **Integration with Spring IoC**
- **Program integration using pointcuts**

4

Java EE training: <http://courses.coreservlets.com>

© 2008 coreservlets.com



# Aspect-Oriented Programming

**Customized Java EE Training: <http://courses.coreservlets.com/>**  
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

# Aspect-Oriented Programming

- **Addresses broad application responsibilities**
  - Logging
  - Transaction management
  - Error management
- **Modularizes responsibility**
  - e.g., logging
  - The encapsulation of a responsibility is an **advice** element

6

Java EE training: <http://courses.coreservlets.com>

# Aspect-Oriented Programming

- **Identifies program facets**
  - Uses mechanisms beyond traditional OO models
    - e.g., All POJO getter methods from `com.company.persistence` package class members
  - Target execution point is the **join point**
    - e.g., constructor or method invocation
  - Program component linking the execution target is a **pointcut**
- **Applies advisor to program facets**
  - An **aspect** is a single unit associating and applying an **advice** to **pointcuts**

7

Java EE training: <http://courses.coreservlets.com>

# Spring AOP

- **Programming model**
  - Method-only interceptors
- **Advisor**
  - Captures advice behavior using plain POJOs
  - AspectJ APIs
    - Annotations and library APIs
    - Advice beans are well-formed
      - Conform to method conventions but not special interfaces
  - AOP Alliance APIs
    - Advice beans implement interfaces
  - Spring AOP XML schema advice types

8

Java EE training: <http://courses.coreservlets.com>

# Spring AOP Continued

- **Pointcut**
  - AspectJ API annotations mapped to bean methods
  - Spring AOP XML schema pointcut declarations
- **Aspect**
  - AspectJ advice with advice and pointcut definitions using AspectJ annotations
  - Spring AOP XML schema aspect declarations

9

Java EE training: <http://courses.coreservlets.com>



# General Spring IoC

**Customized Java EE Training:** <http://courses.coreservlets.com/>  
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

## Spring IoC Process

- **Develop POJO library**
  - `coreservlets.Customer`, `coreservlets.CustomerQuery`,  
`coreservlets.MockCustomerQuery`
- **Register Spring JARs**
  - `spring-core.jar`, `spring-context.jar`, `spring-beans.jar`, `commons-logging.jar`
- **Create the bean definitions file**
  - `classpath:/coreservletsContext.xml`
- **Register beans**
  - `<bean id="customerQuery"`  
`class="coreservlets.MockCustomerQuery" />`
- **Inject dependencies**
  - `<bean><constructor-arg /></bean>`
- **Initialize the container**
  - `BeanFactory beanFactory = new ClassPathXmlApplicationContext(  
new String[]{"classpath:/ coreservletsContext.xml"});`
- **Access and use beans**
  - `CustomerQuery customerQuery =  
(CustomerQuery) beanFactory.getBean("customerQuery");`

# Develop POJO Library

```
public class Customer {  
  
    private String id;  
    private String name;  
  
    public Customer(String id, String name){  
        this.id = id;  
        this.name = name;  
    }  
    public String getId() {  
        return id;  
    }  
    public String getName() {  
        return name;  
    }  
}
```

Java EE training: <http://courses.coreservlets.com>

# Develop POJO Library

```
public interface CustomerQuery {  
  
    public Customer getCustomerByName(String name);  
  
}
```

Java EE training: <http://courses.coreservlets.com>

# Develop POJO Library

```
public class MockCustomerQuery implements CustomerQuery {  
  
    private List<Customer> customers;  
  
    public MockCustomerQuery(List<Customer>customers) {  
        this.customers = customers;  
    }  
  
    public Customer getCustomerByName(String name) {  
        for(Customer c : customers){  
            if(c.getName().equals(name)){  
                return c;  
            }  
        }  
        return null;  
    }  
}
```

Java EE training: <http://courses.coreservlets.com>

# Create Bean Definitions

- **Location**
  - `classpath:/coreservletsContext.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"  
    xsi:schemaLocation="http://www.springframework.org/schema/beans  
        http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">  
  
</beans>
```

Java EE training: <http://courses.coreservlets.com>

# Register Beans and Interdependencies

```
<beans>
  <bean id="customerQuery"
    class="coreservlets.MockCustomerQuery">
    <constructor-arg>
      <list>
        <bean class="coreservlets.Customer">
          <property name="id" value="jjoe" />
          <property name="name" value="Java Joe" />
        </bean>
        <bean class="coreservlets.Customer">
          <property name="id" value="jjohn" />
          <property name="name" value="Java John" />
        </bean>
      </list>
    </constructor-arg>
  </bean>
</beans>
```

Java EE training: <http://courses.coreservlets.com>

# Access and Use Beans

```
import org.springframework.context.support.*;
public class Main {
  public static void main(String[] args) {

    BeanFactory beanFactory =
      new ClassPathXmlApplicationContext(
        "/coreservletsContext.xml");

    CustomerQuery query =
      (CustomerQuery) beanFactory.getBean("customerQuery");

    Customer customer = query.getCustomerByName("Java Joe");

    System.out.println(customer);
  }
}
```

Standard output

```
Customer id=jjoe, name=Java Joe
```





# Spring AOP Hello World

**Customized Java EE Training:** <http://courses.coreservlets.com/>  
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

## Spring AOP JARs

- **spring-aop.jar**
- **aopalliance.jar**
- **aspectjweaver.jar**
- **cglib.jar**

# Spring AOP Hello World Process

- **Identify concern**
- **Select join points**
  - All methods exposed by `coreservlets.CustomerQuery`
- **Create advice**
  - e.g., implement `org.aopalliance.intercept.MethodInterceptor`
- **Create AOP definitions file**
  - Create file `classpath:/coreservletsAopContext.xml`
  - Register `spring-aop` NS
- **Register advice beans**
  - `<bean/>`
- **Create pointcut definitions**
  - Identify join points
  - `<aop:config><aop:pointcut/></aop:config>`
- **Define aspects**
  - `<aop:config><aop:advisor /></aop:config>`

20

Java EE training: <http://courses.coreservlets.com>

## Select Join Points

- **All persistence methods**
  - `coreservlets.CustomerQuery` method implementations

```
public interface CustomerQuery {  
  
    public Customer getCustomerByName(String name);  
  
}
```

21

Java EE training: <http://courses.coreservlets.com>

# Select Join Points

```
public class MockCustomerQuery implements CustomerQuery {  
  
    private List<Customer> customers;  
  
    public MockCustomerQuery(List<Customer>customers) {  
        this.customers = customers;  
    }  
  
    public Customer getCustomerByName(String name) {  
        for(Customer c : customers){  
            if(c.getName().equals(name)){  
                return c;  
            }  
        }  
        return null;  
    }  
}
```

22

Java EE training: <http://courses.coreservlets.com>

# Create Advice

```
import org.aopalliance.intercept.MethodInterceptor;  
import org.aopalliance.intercept.MethodInvocation;  
  
import coreservlets.Customer;  
  
public class HelloWorldAdvice implements MethodInterceptor {  
  
    public Object invoke(MethodInvocation invocation)  
        throws Throwable {  
  
        return new Customer(){  
            public String toString(){  
                return "Hello World!";  
            }  
        };  
    }  
}
```

23

Java EE training: <http://courses.coreservlets.com>

# Create AOP Definitions File

- Location

- classpath:/coreservletsAopContext.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:aop="http://www.springframework.org/schema/aop"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-2.5.xsd">

</beans>
```

Java EE training: <http://courses.coreservlets.com>

# Register Advice Beans

```
<?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:aop="http://www.springframework.org/schema/aop"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-2.5.xsd">

  <bean id="helloWorldAdvice"
    class="coreservlets.aop.helloworld.HelloWorldAdvice" />

</beans>
```

# Create Pointcut Definitions

```
<?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:aop="http://www.springframework.org/schema/aop"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-2.5.xsd">

  <bean id="helloWorldAdvice"
    class="coreservlets.aop.helloworld.HelloWorldAdvice" />

  <aop:config>
    <aop:pointcut id="customerQueryPointcut"
      expression="target (coreservlets.CustomerQuery)" />
  </aop:config>
</beans>
```

26

Java EE training: <http://courses.coreservlets.com>

# Define Aspects

```
<?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:aop="http://www.springframework.org/schema/aop"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-2.5.xsd">

  <bean id="helloWorldAdvice"
    class="coreservlets.aop.helloworld.HelloWorldAdvice" />

  <aop:config>
    <aop:pointcut id="customerQueryPointcut"
      expression="target (coreservlets.CustomerQuery)" />
    <aop:advisor advice-ref="helloWorldAdvice"
      pointcut-ref="customerQueryPointcut" />
  </aop:config>
</beans>
```

27

Java EE training: <http://courses.coreservlets.com>

# Domain Beans

```
<beans>
  <bean id="customerQuery"
    class="coreservlets.MockCustomerQuery">
    <constructor-arg>
      <list>
        <bean class="coreservlets.Customer">
          <property name="id" value="jjoe" />
          <property name="name" value="Java Joe" />
        </bean>
        <bean class="coreservlets.Customer">
          <property name="id" value="jjohn" />
          <property name="name" value="Java John" />
        </bean>
      </list>
    </constructor-arg>
  </bean>
</beans>
```

Java EE training: <http://courses.coreservlets.com>

# Initialize Spring IoC Container

```
import org.springframework.context.support.*;

public class Main {
  public static void main(String[] args) {

    BeanFactory beanFactory =
      new ClassPathXmlApplicationContext(
        "/coreservletsContext.xml",
        "/coreservletsAopContext.xml");
    ...
  }
}
```

Java EE training: <http://courses.coreservlets.com>

# Access and Use Beans Without Spring AOP

```
import org.springframework.context.support.*;
public class Main {
    public static void main(String[]args) {
        BeanFactory beanFactory =
            new ClassPathXmlApplicationContext(new String[]{
                "/coreservletsContext.xml"});
        CustomerQuery query =
            (CustomerQuery) beanFactory.getBean("customerQuery");

        Customer customer = query.getCustomerByName("Java Joe");

        System.out.println(customer);
    }
}
```

Omitted  
classpath:/coreservletsAopContext.xml

Standard output

```
Customer id=jjoe, name=Java Joe
```

# Access and Use Beans With Spring AOP

```
import org.springframework.context.support.*;
public class Main {
    public static void main(String[]args) {
        BeanFactory beanFactory =
            new ClassPathXmlApplicationContext(new String[]{
                "/coreservletsContext.xml",
                "/coreservletsAopContext.xml"});

        CustomerQuery query =
            (CustomerQuery) beanFactory.getBean("customerQuery");

        Customer customer = query.getCustomerByName("Java Joe");

        System.out.println(customer);
    }
}
```

Output overridden  
by bean advisor

Standard output

```
Hello World!
```



# Integrated Process Spring IoC and AOP

Customized Java EE Training: <http://courses.coreservlets.com/>  
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

## Integrated Process Spring IoC and AOP, Step 1

- **Develop POJO library**
  - e.g., `coreservlets.Customer`,  
`coreservlets.CustomerQuery`,  
`coreservlets.MockCustomerQuery`
- **Register Spring IoC and AOP JARs**
  - `spring-core.jar`, `spring-context.jar`, `spring-beans.jar`, `spring-aop.jar`,  
`aopalliance.jar`, `aspectjweaver.jar`, `cglib.jar`, `commons-logging.jar`
- **Create the bean definitions file**
  - e.g., `classpath:/coreservletsContext.xml`
- **Register beans**
  - e.g., `<bean id="customerQuery"`  
`class="coreservlets.MockCustomerQuery" />`
- **Inject dependencies**
  - e.g., `<bean><constructor-arg/></bean>`  
– e.g., `<bean><property/></bean>`



## Integrated Process Spring IoC and AOP, Step 2

- **Identify concern**
- **Select join points**
  - All methods exposed by `coreservlets.CustomerQuery`
- **Create advice**
  - e.g., implement `org.aopalliance.intercept.MethodInterceptor`
- **Create AOP definitions file**
  - Create file `classpath:/coreservletsAopContext.xml`
  - Register `spring-aop` NS
- **Register advice beans**
  - `<bean/>`
- **Create pointcut definitions**
  - Identify join points
  - `<aop:config><aop:pointcut/></aop:config>`
- **Define aspects**
  - `<aop:config><aop:advisor /></aop:config>`

34

Java EE training: <http://courses.coreservlets.com>

## Integrated Process Spring IoC and AOP, Step 3

- **Initialize the container**
  - Specify both domain bean and Spring AOP definitions paths
  - e.g., `BeanFactory beanFactory = new ClassPathXmlApplicationContext(new String[]{ "classpath:/coreservletsContext.xml", "classpath:/coreservletsAopContext.xml" });`
- **Access and use beans**
  - e.g., `CustomerQuery customerQuery = (CustomerQuery) beanFactory.getBean("customerQuery");`

35

Java EE training: <http://courses.coreservlets.com>



# Addressing the Application

**Customized Java EE Training:** <http://courses.coreservlets.com/>  
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

## Pointcuts

- **Join point**
  - Identifies a program execution point
- **Pointcut**
  - Identifies a set of join points such as all persistence methods
  - Defined by using general specifications about a class system
    - Enclosing class
      - Supertype
      - Modifier
      - Name
      - Package namespace
    - Method
      - Name
      - Return type
      - Argument types

# Pointcut Definition (PCD)

- **Keywords**

- **execution**

- Method signature matching expression
    - Matching includes supertypes

- **within**

- Package and class name matching expression
    - Matching does not include supertypes

- **this**

- Class or supertype matching expression
    - Matches on the proxy bean

- **target**

- Class or supertype matching expression
    - Matches on the actual bean

# Pointcut Definition (PCD)

- **Keywords continued**

- **args**

- Method parameter matching expression
    - Includes runtime supertypes
    - Refines when combined with another PCD; e.g. **target**
    - Also maps join point parameters to advice bean parameters

- **beans**

- Bean identifier pattern matching expression

- **Operators**

- Combines expression combinations using logical operators: **AND**, **OR**, **NOT**

- **Context**

- **@org.aspectj.lang.annotation.Pointcut** content
  - Spring AOP XML **expression** attribute value
    - **<aop:config><aop:pointcut /></aop:config>**



# Execution PCD

Customized Java EE Training: <http://courses.coreservlets.com/>  
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

## Execution PCD Details

- **PCD Expression**
  - `execution(<access modifier>*`  
          `<return type>`  
          `<package/class name>*.<method name>`  
          `(<argument expression>)`
- **Expression elements**
  - Access modifier\*
  - Method return type pattern
  - Enclosing package and class name pattern \*
  - Method name pattern
  - Method argument **expression** pattern
- **Expression example**
  - `execution(java.lang.String get*())`

**\*optional**

# Execution PCD Guidelines

- **Required/optional**
  - Optional elements can be fully omitted
  - Required elements must be represented explicitly or using a pattern; e.g., `execution(* *(..))`
- **Pattern**
  - All expression elements can be replaced fully or partially using a pattern
    - Root expression element such as enclosing package/class name
      - `coreservlets.CustomerQuery`
      - `coreservlets.Customer`
      - `coreservlets.Customer*`
    - Argument sub-expression
      - `(*)`
      - `(coreservlets.Customer*)`

42

Java EE training: <http://courses.coreservlets.com>

# Execution PCD Guidelines

- **Argument sub-expression options**
  - Zero, one, or more arguments
    - `()`
    - `(java.lang.String)`
    - `(java.lang.String, java.lang.String)`
    - `(java.lang.String, java.lang.String, *)`
  - Any argument set
    - `(..)`

43

Java EE training: <http://courses.coreservlets.com>

# Integrated Process Spring IoC and AOP, Step 1

- **Develop POJO library**
  - e.g., `coreservlets.Customer`,  
`coreservlets.CustomerQuery`,  
`coreservlets.MockCustomerQuery`
- **Register Spring IoC and AOP JARs**
  - `spring-core.jar`, `spring-context.jar`, `spring-beans.jar`, `spring-aop.jar`,  
`aopalliance.jar`, `aspectjweaver.jar`, `cglib.jar`, `commons-logging.jar`
- **Create the bean definitions file**
  - e.g., `classpath:/coreservletsContext.xml`
- **Register beans**
  - e.g., `<bean id="customerQuery" class="coreservlets.MockCustomerQuery" />`
- **Inject dependencies**
  - e.g., `<bean><constructor-arg/></bean>`
  - e.g., `<bean><property/></bean>`

Java EE training: <http://courses.coreservlets.com>

# Integrated Process Spring IoC and AOP, Step 2

- **Identify concern**
- **Select join points**
  - All methods exposed by `coreservlets.CustomerQuery`
- **Create advice**
  - e.g., implement `org.aopalliance.intercept.MethodInterceptor`
- **Create AOP definitions file**
  - Create file `classpath:/coreservletsAopContext.xml`
  - Register `spring-aop` NS
- **Register advice beans**
  - `<bean/>`
- **Create pointcut definitions**
  - Identify join points
  - `<aop:config><aop:pointcut expression=""/></aop:config>`
- **Define aspects**
  - `<aop:config><aop:advisor /></aop:config>`

# Integrated Process Spring IoC and AOP, Step 3

- **Initialize the container**

- Specify both domain bean and Spring AOP definitions paths
- e.g., `BeanFactory beanFactory = new ClassPathXmlApplicationContext(new String[] { "classpath:/coreservletsContext.xml", "classpath:/coreservletsAopContext.xml" });`

- **Access and use beans**

- e.g., `CustomerQuery customerQuery = (CustomerQuery) beanFactory.getBean("customerQuery");`

46

Java EE training: <http://courses.coreservlets.com>

# Execution PCD Process Step 1 Highlights

- **classpath:/coreservletsContext.xml**

```
<beans>
  <bean id="customerQuery" class="coreservlets.MockCustomerQuery">
    <constructor-arg>
      <list>
        <bean class="coreservlets.Customer">
          <property name="id" value="jjoe" />
          <property name="name" value="Java Joe" />
        </bean>
        <bean class="coreservlets.Customer">
          <property name="id" value="jjohn" />
          <property name="name" value="Java John" />
        </bean>
      </list>
    </constructor-arg>
  </bean>
</beans>
```

Java EE training: <http://courses.coreservlets.com>

# Select Join Points

- All persistence methods
  - `coreservlets.CustomerQuery` method implementations

```
public interface CustomerQuery {  
  
    public Customer getCustomerByName(String name);  
  
}
```

# Select Join Points

```
public class MockCustomerQuery implements CustomerQuery {  
  
    private List<Customer> customers;  
  
    public MockCustomerQuery(List<Customer>customers) {  
        this.customers = customers;  
    }  
  
    public Customer getCustomerByName(String name) {  
        for(Customer c : customers){  
            if(c.getName().equals(name)){  
                return c;  
            }  
        }  
        return null;  
    }  
}
```



## Create Advice

```
import org.aopalliance.intercept.*

public class SummarizingMethodAdvice
    implements MethodInterceptor {

    public Object invoke(MethodInvocation inv)
        throws Throwable {
        ...
        return inv.proceed();
        ...
    }
}
```

50

Java EE training: <http://courses.coreservlets.com>

## Create Advice Continued

```
public Object invoke(MethodInvocation inv) throws Throwable {
    String buffer = inv.getMethod().toGenericString()
        + ". args=" + Arrays.toString(inv.getArguments());
    try{
        Object returnValue = inv.proceed();
        buffer += ". exit=return[" + returnValue + "];
        return returnValue;
    }
    catch(Throwable t){
        buffer += " - exit=error[" + t + "];
        throw t;
    }
    finally{
        Logger log = Logger.getLogger(inv.getThis().getClass());
        log.debug(">>>" + buffer);
    }
}
```

51

Java EE training: <http://courses.coreservlets.com>

# Create AOP Definitions

- Location

- `classpath:/coreservletsAopContext.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:aop="http://www.springframework.org/schema/aop"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-2.5.xsd">

</beans>
```

Java EE training: <http://courses.coreservlets.com>

# Register Advice Beans

```
<?xml version="1.0" encoding="UTF-8"?>
<beans>

  <bean id="summarizingMethodAdvice"
    class="coreservlets.SummarizingMethodAdvice" />

</beans>
```

# Define Pointcuts

```
<?xml version="1.0" encoding="UTF-8"?>
<beans>

  <bean id="summarizingMethodAdvice"
        class="coreservlets.SummarizingMethodAdvice" />

  <aop:config>
    <aop:pointcut id="anyMethodPcd"
                  expression="execution(* *(..))" />
  </aop:config>
</beans>
```

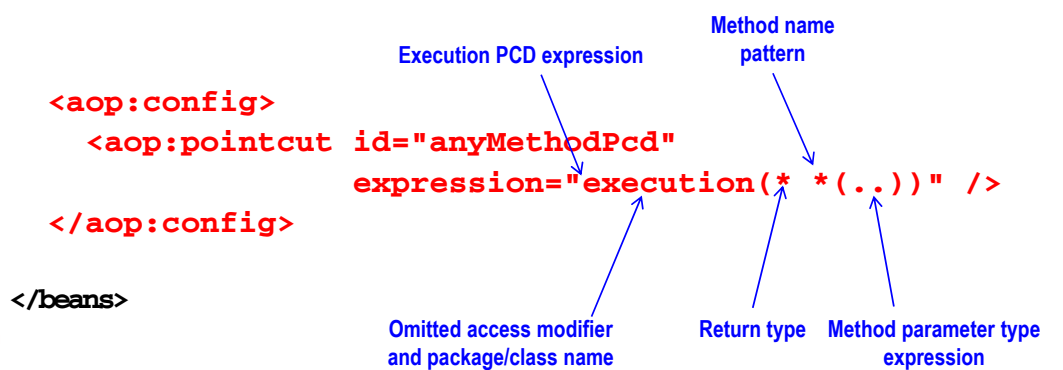
Execution PCD expression

Method name pattern

Omitted access modifier and package/class name

Return type

Method parameter type expression



54

Java EE training: <http://courses.coreservlets.com>

# Define Aspects

```
<?xml version="1.0" encoding="UTF-8"?>
<beans>

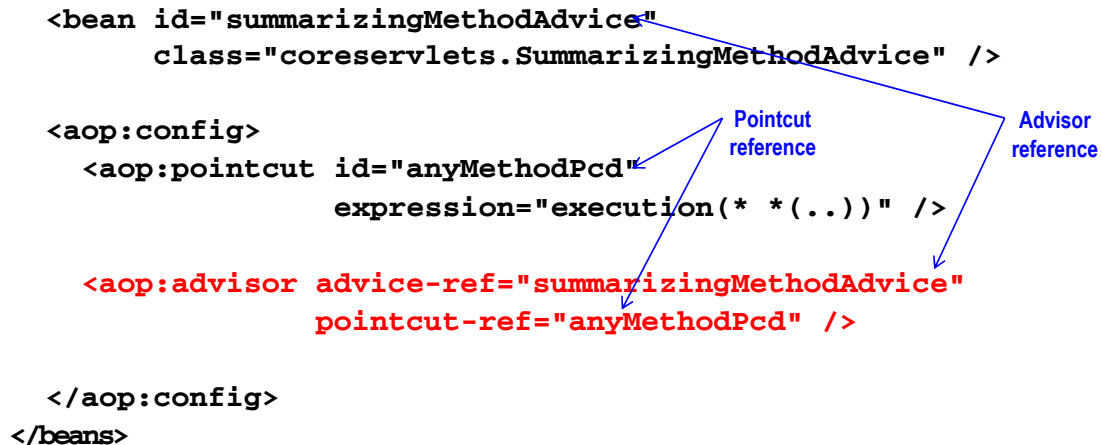
  <bean id="summarizingMethodAdvice"
        class="coreservlets.SummarizingMethodAdvice" />

  <aop:config>
    <aop:pointcut id="anyMethodPcd"
                  expression="execution(* *(..))" />

    <aop:advisor advice-ref="summarizingMethodAdvice"
                 pointcut-ref="anyMethodPcd" />
  </aop:config>
</beans>
```

Pointcut reference

Advisor reference



55

Java EE training: <http://courses.coreservlets.com>

# Initialize Spring IoC Container

```
import org.springframework.context.support.*;

public class Main {
    public static void main(String[] args) {

        BeanFactory beanFactory =
            new ClassPathXmlApplicationContext(
                "/coreservletsContext.xml",
                "/coreservletsAopContext.xml");
        ...
    }
}
```

Java EE training: <http://courses.coreservlets.com>

# Access and Use Beans Without Spring AOP

```
import org.springframework.context.support.*;
public class Main {
    public static void main(String[] args) {
        BeanFactory beanFactory =
            new ClassPathXmlApplicationContext(new String[]{
                "/coreservletsContext.xml"});
        CustomerQuery query =
            (CustomerQuery) beanFactory.getBean("customerQuery");
        Customer customer = query.getCustomerByName("Java Joe");
    }
}
```

Omitted [classpath:/coreservletsAopContext.xml](#)

Standard output



# Access and Use Beans With Spring AOP

```
import org.springframework.context.support.*;
public class Main {
    public static void main(String[] args) {
        BeanFactory beanFactory =
            new ClassPathXmlApplicationContext(new String[]{
                "/coreservletsContext.xml",
                "/coreservletsAopContext.xml"});
        CustomerQuery query =
            (CustomerQuery) beanFactory.getBean("customerQuery");
        Customer customer = query.getCustomerByName("Java Joe");
    }
}
Standard output
>>> public java.lang.String coreservlets.Customer.getName().
args=[]. exit=return[Java Joe]

>>> public java.lang.String coreservlets.Customer.toString().
args=[]. exit=return[Customer id=jjoe, name=Java Joe]

>>> public abstract coreservlets.Customer
coreservlets.CustomerQuery.getCustomerByName(java.lang.String).
args=[Java Joe]. exit=return[Customer id=jjoe, name=Java Joe]
```

## Execution PCD Match Origins

```
>>> public java.lang.String coreservlets.Customer.getName().
args=[]. exit=return[Java Joe]

>>> public java.lang.String coreservlets.Customer.toString().
args=[]. exit=return[Customer id=jjoe, name=Java Joe]

>>> public abstract coreservlets.Customer
coreservlets.CustomerQuery.getCustomerByName(java.lang.String).
args=[Java Joe]. exit=return[Customer id=jjoe, name=Java Joe]

public class MockCustomerQuery implements CustomerQuery {
    ...
    public Customer getCustomerByName(String name) {
        for(Customer customer : customers){
            if(customer.getName().equals(name)){
                return customer;
            }
        }
        return null;
    }
}
execution(* *(..))
```


# Execution PCD Match Origins

```
>>> public java.lang.String coreservlets.Customer.getName().
args=[]. exit=return[Java Joe]

>>> public java.lang.String coreservlets.Customer.toString().
args=[]. exit=return[Customer id=jjoe, name=Java Joe]

>>> public abstract coreservlets.Customer
coreservlets.CustomerQuery.getCustomerByName(java.lang.String).
args=[Java Joe]. exit=return[Customer id=jjoe, name=Java Joe]
```

```
public class SummarizingMethodAdvice
public Object invoke(MethodInvocation inv)
throws Throwable {
    ...
    Object returnValue = inv.proceed();
    buffer += ". exit=return[" + returnValue + "];
    ...
}
```



60

Java EE training: <http://courses.coreservlets.com>

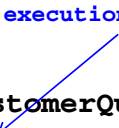
# Execution PCD Match Origins

```
>>> public java.lang.String coreservlets.Customer.getName().
args=[]. exit=return[Java Joe]

>>> public java.lang.String coreservlets.Customer.toString().
args=[]. exit=return[Customer id=jjoe, name=Java Joe]

>>> public abstract coreservlets.Customer
coreservlets.CustomerQuery.getCustomerByName(java.lang.String).
args=[Java Joe]. exit=return[Customer id=jjoe, name=Java Joe]
```

```
public class Main {
public static void main(String[]args) {
    BeanFactory beanFactory =
        new ClassPathXmlApplicationContext(new String[]{
            "/coreservletsContext.xml",
            "/coreservletsAopContext.xml"});
    CustomerQuery query =
        (CustomerQuery) beanFactory.getBean("customerQuery");
    Customer customer = query.getCustomerByName("Java Joe");
}
```



61

Java EE training: <http://courses.coreservlets.com>

## More Execution PCD Examples

- **Drop access modifier**
  - `<aop:pointcut expression="execution(coreservlets.Customer coreservlets.CustomerQuery.getCustomerByName ( java.lang.String) )"/>`
- **Any access modifier and return type**
  - `<aop:pointcut expression="execution(* coreservlets.CustomerQuery.getCustomerByName ( java.lang.String) )"/>`
- **Any access modifier, return type and coreservlets package**
  - `<aop:pointcut expression="execution(* coreservlets.*.CustomerQuery.getCustomerByName ( java.lang.String) )"/>`

62

Java EE training: <http://courses.coreservlets.com>

## More Execution PCD Examples Continued

- **Any access modifier, return type, package, class, and method name**
  - `<aop:pointcut expression="execution(* *(java.lang.String) )"/>`
- **Any public bean property getter method**
  - `<aop:pointcut expression="execution(public * get*())"/>`
- **Any public bean property setter method**
  - `<aop:pointcut expression="execution(public void set*(*))"/>`

63

Java EE training: <http://courses.coreservlets.com>



# Within PCD

**Customized Java EE Training:** <http://courses.coreservlets.com/>  
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

## Within PCD Details

- **PCD Expression**
  - `within(<package/class name>)`
- **Expression elements**
  - Enclosing package and class name **pattern**
- **Expression example**
  - `within(coreservlets.MockCustomerQuery)`



## Within PCD Guidelines

- **Match target**
  - Matches on the instantiated/concrete bean type
  - Does not match on target bean supertypes
    - e.g., `coreservlets.CustomerQuery` will not match on an instantiation of `coreservlets.MockCustomerQuery`
  - Does not match on method signature elements
- **Pattern**
  - Expression supports partial or full replacement using wildcard(s)

66

Java EE training: <http://courses.coreservlets.com>

## Integrated Process Spring IoC and AOP, Step 1

- **Develop POJO library**
  - e.g., `coreservlets.Customer`,  
`coreservlets.CustomerQuery`,  
`coreservlets.MockCustomerQuery`
- **Register Spring IoC and AOP JARs**
  - `spring-core.jar`, `spring-context.jar`, `spring-beans.jar`, `spring-aop.jar`,  
`aopalliance.jar`, `aspectjweaver.jar`, `cglib.jar`, `commons-logging.jar`
- **Create the bean definitions file**
  - e.g., `classpath:/coreservletsContext.xml`
- **Register beans**
  - e.g., `<bean id="customerQuery" class="coreservlets.MockCustomerQuery" />`
- **Inject dependencies**
  - e.g., `<bean><constructor-arg/></bean>`
  - e.g., `<bean><property/></bean>`

Java EE training: <http://courses.coreservlets.com>

## Integrated Process Spring IoC and AOP, Step 2

- **Identify concern**
- **Select join points**
  - All methods exposed by `coreservlets.CustomerQuery`
- **Create advice**
  - e.g., implement `org.aopalliance.intercept.MethodInterceptor`
- **Create AOP definitions file**
  - Create file `classpath:/coreservletsAopContext.xml`
  - Register `spring-aop` NS
- **Register advice beans**
  - `<bean/>`
- **Create pointcut definitions**
  - Identify join points
  - `<aop:config><aop:pointcut expression=""/></aop:config>`
- **Define aspects**
  - `<aop:config><aop:advisor /></aop:config>`

68

Java EE training: <http://courses.coreservlets.com>

## Integrated Process Spring IoC and AOP, Step 3

- **Initialize the container**
  - Specify both domain bean and Spring AOP definitions paths
  - e.g., `BeanFactory beanFactory = new ClassPathXmlApplicationContext(new String[] { "classpath:/coreservletsContext.xml", "classpath:/coreservletsAopContext.xml" });`
- **Access and use beans**
  - e.g., `CustomerQuery customerQuery = (CustomerQuery) beanFactory.getBean("customerQuery");`

69

Java EE training: <http://courses.coreservlets.com>

# Execution PCD Process Step 1 Highlights

- `classpath:/coreservletsContext.xml`

```
<beans>
  <bean id="customerQuery" class="coreservlets.MockCustomerQuery">
    <constructor-arg>
      <list>
        <bean class="coreservlets.Customer">
          <property name="id" value="jjoe" />
          <property name="name" value="Java Joe" />
        </bean>
        <bean class="coreservlets.Customer">
          <property name="id" value="jjohn" />
          <property name="name" value="Java John" />
        </bean>
      </list>
    </constructor-arg>
  </bean>
</beans>
```

Java EE training: <http://courses.coreservlets.com>

## Select Join Points

- **General coreservlets library methods**
  - `coreservlets` package members
    - All class member methods
      - `expression="within(coreservlets.*)"`

```
coreservlets.Customer
coreservlets.CustomerQuery
coreservlets.MockCustomerQuery
coreservlets.jdbe.JdbeCustomerQuery
```

## Create Advice

```
import org.aopalliance.intercept.*

public class SummarizingMethodAdvice
    implements MethodInterceptor {

    public Object invoke(MethodInvocation inv)
        throws Throwable {
        ...
        return inv.proceed();
        ...
    }
}
```

72

Java EE training: <http://courses.coreservlets.com>

## Create Advice Continued

```
public Object invoke(MethodInvocation inv) throws Throwable {
    String buffer = inv.getMethod().toGenericString()
        + ". args=" + Arrays.toString(inv.getArguments());
    try{
        Object returnValue = inv.proceed();
        buffer += ". exit=return[" + returnValue + "];
        return returnValue;
    }
    catch(Throwable t){
        buffer += " - exit=error[" + t + "];
        throw t;
    }
    finally{
        Logger log = Logger.getLogger(inv.getThis().getClass());
        log.debug(">>>" + buffer);
    }
}
```

73

Java EE training: <http://courses.coreservlets.com>

# Create AOP Definitions File

- Location

- `classpath:/coreservletsAopContext.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:aop="http://www.springframework.org/schema/aop"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-2.5.xsd">

</beans>
```

Java EE training: <http://courses.coreservlets.com>

# Register Advisor Beans

```
<?xml version="1.0" encoding="UTF-8"?>
<beans>

  <bean id="summarizingMethodAdvice"
    class="coreservlets.SummarizingMethodAdvice" />

</beans>
```

# Define Pointcuts

```
<?xml version="1.0" encoding="UTF-8"?>
<beans>

  <bean id="summarizingMethodAdvice"
        class="coreservlets.SummarizingMethodAdvice" />

  <aop:config>
    <aop:pointcut id="coreservletsPackagePcd"
                  expression="within(coreservlets.*)" />
  </aop:config>
</beans>
```

Execution PCD expression

Package namespace

PCD expression keyword

Class wildcard  
Does not match on sub-packages

76

Java EE training: <http://courses.coreservlets.com>

# Define Aspects

```
<?xml version="1.0" encoding="UTF-8"?>
<beans>

  <bean id="summarizingMethodAdvice"
        class="coreservlets.SummarizingMethodAdvice" />

  <aop:config>
    <aop:pointcut id="coreservletsPackagePcd"
                  expression="within(coreservlets.*)" />
    <aop:advisor advice-ref="summarizingMethodAdvice"
                 pointcut-ref="coreservletsPackagePcd" />
  </aop:config>
</beans>
```

Pointcut reference

Advisor reference

77

Java EE training: <http://courses.coreservlets.com>

# Initialize Spring IoC Container

```
import org.springframework.context.support.*;

public class Main {
    public static void main(String[]args) {

        BeanFactory beanFactory =
            new ClassPathXmlApplicationContext(
                "/coreservletsContext.xml",
                "/coreservletsAopContext.xml");
        ...
    }
}
```

Java EE training: <http://courses.coreservlets.com>

# Access and Use Beans

```
import org.springframework.context.support.*;
public class Main {
    public static void main(String[]args) {
        BeanFactory beanFactory =
            new ClassPathXmlApplicationContext(new String[]{
                "/coreservletsContext.xml",
                "/coreservletsAopContext.xml"});
        CustomerQuery query =
            (CustomerQuery) beanFactory.getBean("customerQuery");
        Customer customer = query.getCustomerByName("Java Joe");
    }
}
```

Standard output

```
>>> public java.lang.String coreservlets.Customer.getName().
args=[]. exit=return[Java Joe]

>>> public java.lang.String coreservlets.Customer.toString().
args=[]. exit=return[Customer id=jjoe, name=Java Joe]

>>> public abstract coreservlets.Customer
coreservlets.CustomerQuery.getCustomerByName(java.lang.String).
args=[Java Joe]. exit=return[Customer id=jjoe, name=Java Joe]
```

# More Within PCD Examples

- **Fully-qualified classname**
  - Configuration is bound to the implementation type

```
<aop:pointcut  
    expression="within(coreservlets.MockCustomerQuery)"/>
```
- **Interface-type**
  - Quietly fails to match

```
<aop:pointcut  
    expression="within(coreservlets.MockCustomerQuery)"/>
```
- **Interface-type**
  - Name must be within the coreservlets package and include the term `CustomerQuery`

```
<aop:pointcut  
    expression="within(coreservlets.*CustomerQuery)"/>
```

80

Java EE training: <http://courses.coreservlets.com>

# More Within PCD Examples

- **General package name pattern**
  - Matches only on the immediate members of the `coreservlets` package

```
<aop:pointcut  
    expression="within(coreservlets.*)"/>
```
- **All methods from all classes of all packages**
  - What's the point?

```
<aop:pointcut expression="within(*)"/>
```

81

Java EE training: <http://courses.coreservlets.com>





# Target PCD

**Customized Java EE Training:** <http://courses.coreservlets.com/>  
Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

## Target PCD Details

- **PCD Expression**
  - `target(<package/class name>)`
- **Expression elements**
  - Enclosing package and class name **pattern**
- **Expression example**
  - `target(coreservlets.CustomerQuery)`
- **Match target**
  - Matches on the instantiated/concrete bean type
  - Matches on target bean supertypes
  - Does not match on method signature elements
- **Pattern support**
  - NONE
  - **Wildcards will generate errors**

# Target PCD Guidelines

- **Assists**
  - POJOs having well-defined interface types
  - Many concrete POJOs implementing marker interfaces
- **Hinders**
  - Concrete types sharing no general interfaces

# Execution PCD Process Step 1 Highlights

- **classpath:/coreservletsContext.xml**

```
<beans>
  <bean id="customerQuery" class="coreservlets.MockCustomerQuery">
    <constructor-arg>
      <list>
        <bean class="coreservlets.Customer">
          <property name="id" value="jjoe" />
          <property name="name" value="Java Joe" />
        </bean>
        <bean class="coreservlets.Customer">
          <property name="id" value="jjohn" />
          <property name="name" value="Java John" />
        </bean>
      </list>
    </constructor-arg>
  </bean>
</beans>
```

## Select Join Points

- General `coreservlets.CustomerQuery` implementors and methods

```
package coreservlets;  
  
public interface CustomerQuery {  
  
    public Customer getCustomerByName(String name);  
  
}
```

## Create Advice

```
import org.aopalliance.intercept.*  
  
public class SummarizingMethodAdvice  
    implements MethodInterceptor {  
  
    public Object invoke(MethodInvocation inv)  
        throws Throwable {  
        ...  
        return inv.proceed();  
        ...  
    }  
}
```

# Create Advice Continued

```
public Object invoke(MethodInvocation inv) throws Throwable {
    String buffer = inv.getMethod().toGenericString()
        + ". args=" + Arrays.toString(inv.getArguments());
    try{
        Object returnValue = inv.proceed();
        buffer += ". exit=return[" + returnValue + "];
        return returnValue;
    }
    catch(Throwable t){
        buffer += " - exit=error[" + t + "];
        throw t;
    }
    finally{
        Logger log = Logger.getLogger(inv.getThis().getClass());
        log.debug(">>>" + buffer);
    }
}
```

88

Java EE training: <http://courses.coreservlets.com>

# Create AOP Definitions File

- Location

- classpath:/coreservletsAopContext.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:aop="http://www.springframework.org/schema/aop"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop-2.5.xsd">

</beans>
```

Java EE training: <http://courses.coreservlets.com>

# Define Pointcuts

```
<?xml version="1.0" encoding="UTF-8"?>
<beans>

  <bean id="summarizingMethodAdvice"
        class="coreservlets.SummarizingMethodAdvice" />

  <aop:config>

    <aop:pointcut id="customerQueryPcd"
                  expression="target(coreservlets.CustomerQuery)" />

  </aop:config>

</beans>
```

Execution PCD expression

PCD expression keyword

Fully-qualified interface type

90

Java EE training: <http://courses.coreservlets.com>

# Define Aspects

```
<?xml version="1.0" encoding="UTF-8"?>
<beans>

  <bean id="summarizingMethodAdvice"
        class="coreservlets.SummarizingMethodAdvice" />

  <aop:config>

    <aop:pointcut id="customerQueryPcd"
                  expression="target(coreservlets.CustomerQuery)" />

    <aop:advisor advice-ref="summarizingMethodAdvice"
                 pointcut-ref="customerQueryPcd" />

  </aop:config>

</beans>
```

Pointcut reference

Advisor reference

91

Java EE training: <http://courses.coreservlets.com>

# Initialize Spring IoC Container

```
import org.springframework.context.support.*;

public class Main {
    public static void main(String[]args) {

        BeanFactory beanFactory =
            new ClassPathXmlApplicationContext(
                "/coreservletsContext.xml",
                "/coreservletsAopContext.xml");
        ...
    }
}
```

Java EE training: <http://courses.coreservlets.com>

# Access and Use Beans

```
import org.springframework.context.support.*;
public class Main {
    public static void main(String[]args) {

        BeanFactory beanFactory =
            new ClassPathXmlApplicationContext(new String[]{
                "/coreservletsContext.xml",
                "/coreservletsAopContext.xml"});

        CustomerQuery query =
            (CustomerQuery) beanFactory.getBean("customerQuery");

        Customer customer = query.getCustomerByName("Java Joe");
    }
}
```

Standard output

```
>>> [MockCustomerQuery] public abstract coreservlets.Customer
coreservlets.CustomerQuery.getCustomerByName(java.lang.String).
args=[Java Joe]. exit=return[Customer id=jjoe, name=Java Joe]
```

# More Target PCD Examples

- **Concrete type**

- Configuration is bound to a particular concrete type

```
<aop:pointcut  
    expression="target (coreservlets.MockCustomerQuery) "/>
```

- **Supertype may be an interface or class**

```
<aop:pointcut  
    expression="target (java.lang.Object) "/>
```



## Wrap-up

# Summary

- **Manage AOP code and definitions separately**
- **Identify concerns and join points**
- **Create advice**
  - e.g., implement `org.aopalliance.intercept.MethodInterceptor`
- **Create AOP definitions file**
  - Create file `classpath:/coreservletsAopContext.xml`
  - Register `spring-aop` NS
- **Register advice beans**
  - `<bean/>`
- **Create pointcut definitions**
  - Identify join points
  - `<aop:config><aop:pointcut expression=""/></aop:config>`
- **Define aspects**
  - `<aop:config><aop:advisor /></aop:config>`
- **Merge AOP code and definitions with domain products at runtime**

96

Java EE training: <http://courses.coreservlets.com>

© 2008 coreservlets.com



## Questions?

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, Struts, JSF/MyFaces/Facelets, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.