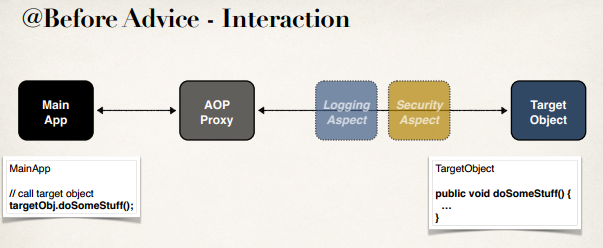
**AOP @Before Advice Type**

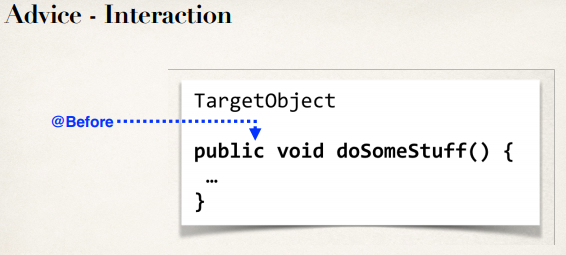
**Advice types**:

* **Before advice**: run before the method
* **After returning advice**: run after the method (success execution)
* **After throwing advice**: run after method (if exception thrown)
* **After finally advice**: run after the method (finally)
* **Around advice**: run before and after method

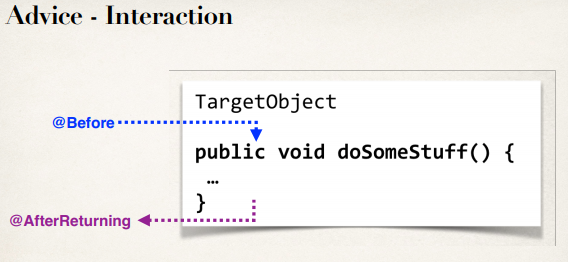
**@Before Advice – Interaction**:



**Advice – Interaction**:



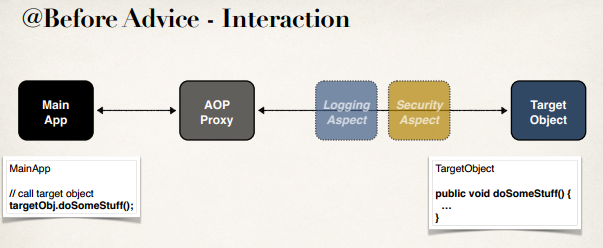
**Advice – Interaction – After Returning**:



**@Before Advice - Use Cases**:

* **Most common**
  + logging, security, transactions
* **Audit logging**
  + who, what, when, where?
* **API Management**
  + how many times has a method been called user?
  + analytics: what are peak times? what is average load? who is top user?

**AOP Example – Overview (Step-by-Step)**:



**Adding AspectJ JAR File**:

* Need to download AspectJ JAR file
* Even though we are using Spring AOP … still need AspectJ JAR file
* Why?
  + Spring AOP uses some of the AspectJ annotations
  + Spring AOP uses some of the AspectJ classes

**Development Process - @Before (Step-by-Step)**:

1. **Create target object: AccountDAO**
2. **Create Spring Java Config class**
3. **Create main app**
4. **Create an Aspect with @Before advice**

**Step 1: Create Target Object: AccountDAO**:

@Component

**public** **class** AccountDAO {

**public** **void** addAccount() {

System.***out***.println(getClass().getName() + ": DOING MY DB WORK:

ADDING AN ACCOUNT");

}

}

**2) Create Spring Java Config class**:

@Configuration

@EnableAspectJAutoProxy

@ComponentScan("com.ruhul.aopdemo")

**public** **class** DemoConfig {

}

@Configuration => Spring Pure Java Configuration

@EnableAspectJAutoProxy => Spring AOP Proxy Support

@ComponentScan("com.ruhul.aopdemo")=> Component scan for components and aspects (Resource Package)

**3) Create main app**:

**public** **class** MainDemoApp {

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

// read spring config java class

AnnotationConfigApplicationContext context =

**new** AnnotationConfigApplicationContext(DemoConfig.**class**);

// get the bean from spring container

AccountDAO theAccountDAO = context.getBean("accountDAO", AccountDAO.**class**);

// call the business method

theAccountDAO.addAccount();

// do it again

System.***out***.println("\n Let's call the method 'addAccount()' again");

// call the business method

theAccountDAO.addAccount();

// close the context

context.close();

}

}

**4) Create an Aspect with @Before advice**

@Aspect

@Component

**public** **class** MyDemoLoggingAspect {

// this is where we add all of our related advice for logging

// let's start with an @Before advice

@Before("execution(public void addAccount())")

**public** **void** beforeAddAccountAdvice() {

System.***out***.println("\n====>>> Executing @Before advice on addAccount()");

}

}

@Aspect => Special Aspect Class

@Before => Point-cut expression => Run this code BEFORE – target object method: "public void addAccount()"

AOP @Before Advice Type