**How to create a proxy call for an Employee object?**

Ex: Spring AOP works on the concepts of proxy. Bean is managed by IOC container. Whenever I call behaviour on a Bean, AOP aspect point cut (@Before, @After) is executed, then the bean behaviour is called.

The same can be done for any object… Ex: RMI, RPC.

Below is the example code to do the same.

Basically this was done through Proxy object on Reflection, Invocation Handler.

**package** com.sample.basics;

**import** java.lang.reflect.InvocationHandler;

**import** java.lang.reflect.Method;

**import** java.lang.reflect.Proxy;

**interface** Person {

**void** getName();

}

**class** Employee **implements** Person {

String name;

Employee(String name) {

**this**.name = name;

}

**public** **void** getName() {

System.***out***.println("Employee is called. The name is " + name);

}

}

**class** EmployeeProxy **implements** InvocationHandler {

Object obj;

**public** EmployeeProxy(Object o) {

obj = o;

}

**public** Object invoke(Object proxy, Method m, Object[] args)

**throws** Throwable {

Object result = **null**;

**try** {

System.***out***

.println("Employee Proxy is called before Employee invocation");

result = m.invoke(obj, args);

} **finally** {

System.***out***.println("Employee Proxy is called after Employee invocation");

}

**return** result;

}

**public** **static** Person getProxy(Person person) {

**return** (Person) Proxy.*newProxyInstance*(

Person.**class**.getClassLoader(),

**new** Class[] { Person.**class** }, **new** EmployeeProxy(

person));

}

}

**public** **class** ProxyObjectExample {

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

Person employee = EmployeeProxy.*getProxy*(**new** Employee("Prabhu"));

employee.getName();

}

}