Java Dynamic Proxies

Fabienne Boyer
Laboratoire LIG (UGA)

http://sardes.imag.fr/~boyer
Fabienne.Boyer@imag.fr

Adressed Issue

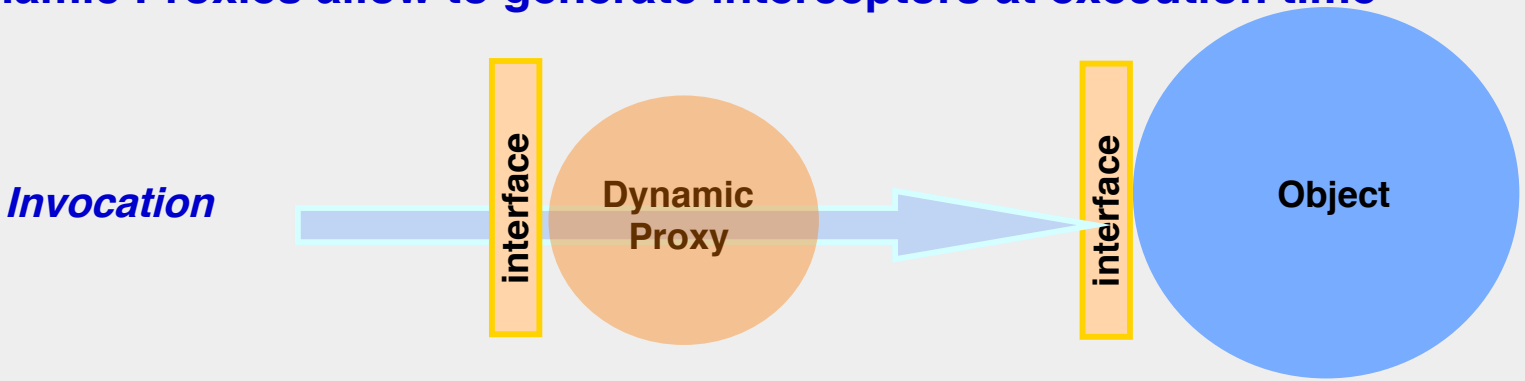
Interception

- Intercept the invocations to a given object
- Add instructions before / after / in place of the invocation (test, debug, transaction management)

Approaches

- Source code manipulation, at compile time (or before)
- Bytecode manipulation, at load time (or before)
- Object manipulation, at execution time

→ Dynamic Proxies allow to generate interceptors at execution time



Principles

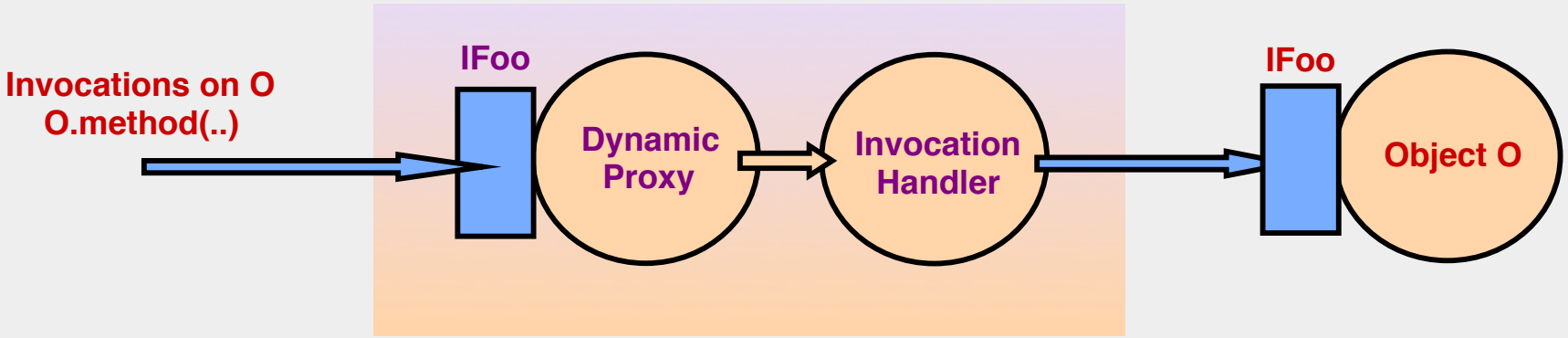
- Proxies are instances of classes implementing some target objects interfaces
  - Classes generation performed through java.lang.reflect.Proxy.getProxyClass ()
  - Instances generation performed through java.lang.reflect.Proxy.newProxyInstance()
- Proxies are in fact composed of two parts
  - An interceptor, called the DynamicProxy, in charge of intercepting the actual invocations
  - An invocation handler, in charge of performing the actual invocations

Principles

- Create a dynamic proxy class + instance for a target object

```
Proxy.newProxyInstance(classLoader, interfaces, invocationHdler)
```

- Perform invocations on the target object as usual
- Invocations are managed through an InvocationHandler



Dynamic Proxy

- A dynamic proxy is associated to
  - A class loader
  - An interface
  - An invocation handler
- Create a dynamic proxy
  - Through the static method newProxyInstance on the class Proxy

```
InvocationHandler invocationHandler =  new IFooInvocationHandler();
IFoo proxy = Proxy.newProxyInstance(
                    IFoo.class.getClassLoader(),
                    new Class[] {IFoo.class},
                    invocationHdler);
```

Invocation Handler

- **Define an Invocation Handler class for a given interface (e.g., IFoo)**
  - **Should implement InvocationHandler**

```
public interface InvocationHandler(
    Object invoke(Object proxy, Method method, Object[] args)
    throws Throwable;
 }
```

```
public class IFooInvocationHandler implements InvocationHandler {
   public Object invoke(Object proxy, Method method, Object[] args)
   throws Throwable{
   ...
}
```

Dynamic Proxy

Called method

Wrapped args

From http://java.sun.com/j2se/1.4/docs

About proxies serialization

- Proxy instances can be serialized
- If the associated InvocationHandler is not serializable, an exception is thrown

Example of a dynamic proxy used for debug

```
public class DebugProxy implements InvocationHandler {
 private Object obj;
 private DebugProxy(Object obj){this.obj= obj;}

 public static Object newInstance(Object obj) {
   return java.lang.reflect.Proxy.newProxyInstance(
        obj.getClass().getClassLoader(),
        obj.getClass().getInterfaces(),
        new DebugProxy(obj));
 }

 public Object invoke(Object proxy, Method m, Object args[]) {
   try {…
        Object result = m.invoke(obj, args);
        …
   } catch (Exception e) {…}
   return result;
 }
}
```

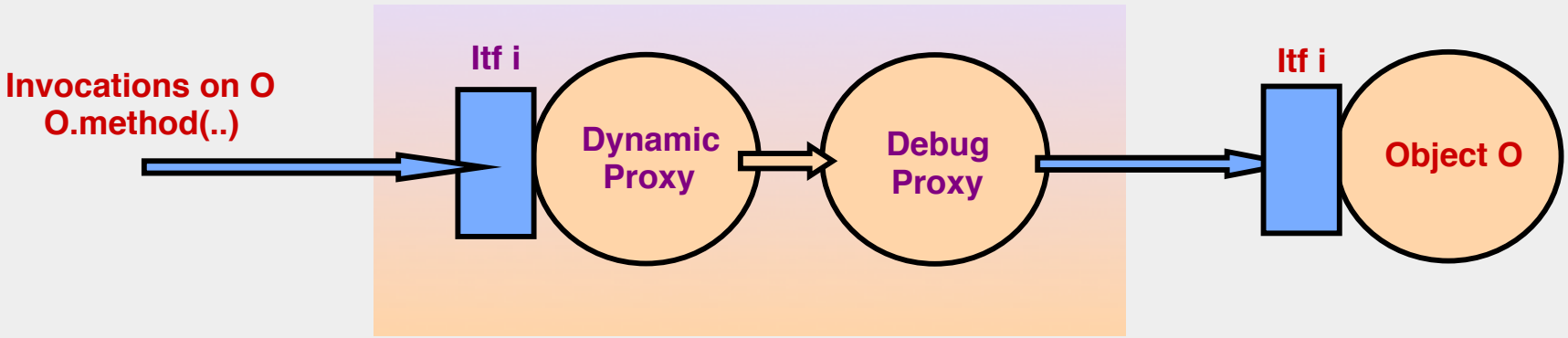
Target object

Proxy interfaces

Create Invocation Hdler

From http://java.sun.com/j2se/1.4/docs

Example of a dynamic proxy used for debug



Example of a dynamic proxy used for debug

```
public interface IFoo(
    Object bar(Object obj) throws BazException;
}

public class Foo implements IFoo {
     Object bar(Object obj) throws BazException {
          …
   }
}

// Construct a DebugProxy for an instance of Foo
IFoo foo = (IFoo) DebugProxy.newInstance(new Foo());
// Call one of the Foo instance methods
foo.bar(null);
```

From http://java.sun.com/j2se/1.4/docs

Example of a dynamic proxy used for Javanaise