# Java Dynamic Proxy

动态代理是使用Java反射技术在运行时创建一个实现给定接口的新类及其示例，代理的是接口。Java原生支持动态代理，其实现类为Proxy，其提供下面的方法来构建对象：

*Proxy.newProxyInstance(ClassLoader ,Class<?>[] interfaces, Invocationhandler).*

* classLoader，类的加载器，使用null表示使用默认的加载器
* interface，需要代理的接口组
* invocationHandler，调用处理器，使用新建的Proxy对象调用方法的时候，都会调用该接口中的invoke方法

下面介绍其示例，对ArrayList对象提供Proxy代理，

1. 定义Handler，MyInvocationHandler

*package com.fys.learnjava.proxy;*

*import java.lang.reflect.InvocationHandler;*

*import java.lang.reflect.Method;*

*public class MyInvocationHandler implements InvocationHandler {*

*Object target;*

*public MyInvocationHandler(Object obj) {*

*target = obj;*

*}*

*public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {*

*System.out.println("Method Name:"+method.getName());*

*if(method.getName().equals("add")) { //add 方法中如果参数为apple，则返回false*

*if (args[0].equals("apple")) { //调用失败*

*return false;*

*}*

*}*

*return method.invoke(target,args);*

*}*

*}*

1. 测试程序，为ArrayList创建代理类

*import java.lang.reflect.Proxy;*

*import java.util.ArrayList;*

*import java.util.List;*

*public class ProxyMotion {*

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

*ArrayList<String> content = new ArrayList<String>();*

*MyInvocationHandler handler = new MyInvocationHandler(content);*

*//创建代理类*

*Object proxy = Proxy.newProxyInstance(null, new Class[]{List.class},handler);*

*if(proxy instanceof List) {*

*System.out.println("Proxy is List");*

*List<String> mlist = (List<String>) proxy;*

*mlist.add("One");*

*mlist.add("Two");*

*mlist.add("Apple");*

*mlist.add("Three");*

*}*

*System.out.println("Proxy:"+ proxy.toString());*

*System.out.println("Content:"+ content.toString());*

*}*

*}*

1. 输出如下，使用InvocationHandler过滤掉Apple字符串

*Proxy is List*

*Method Name:add*

*Method Name:add*

*Method Name:add*

*Method Name:add*

*Method Name:toString*

*Proxy:[One, Two, Three]*

*Content:[One, Two, Three]*