digitalocean.com

Proxy Design Pattern

Pankaj

3–4 minutes

Proxy Design pattern is one of the Structural design pattern and in my opinion one of the simplest pattern to understand.

Proxy Design Pattern

Proxy design pattern intent according to GoF is: **Provide a surrogate or placeholder for another object to control access to it.** The definition itself is very clear and proxy design pattern is used when we want to provide controlled access of a functionality. Let's say we have a class that can run some command on the system. Now if we are using it, its fine but if we want to give this program to a client application, it can have severe issues because client program can issue command to delete some system files or change some settings that you don't want. Here a proxy class can be created to provide controlled access of the program.

Proxy Design Pattern - Main Class

Since we code Java in terms of interfaces, here is our interface and its implementation class. `CommandExecutor.java`

```
package com.journaldev.design.proxy;

public interface CommandExecutor {

	public void runCommand(String cmd) throws
Exception;
}
```

CommandExecutorImpl.java

```
package com.journaldev.design.proxy;

import java.io.IOException;

public class CommandExecutorImpl implements
CommandExecutor {

	@Override
	public void runCommand(String cmd) throws
IOException {
		//some heavy implementation
		Runtime.getRuntime().exec(cmd);
		System.out.println("'" + cmd + "'
command executed.");
	}

}
```

Proxy Design Pattern - Proxy Class

Now we want to provide only admin users to have full access of above class, if the user is not admin then only limited commands will be allowed. Here is our very simple proxy class implementation.
CommandExecutorProxy.java

```
package com.journaldev.design.proxy;

public class CommandExecutorProxy implements
CommandExecutor {

	private boolean isAdmin;
	private CommandExecutor executor;

	public CommandExecutorProxy(String user,
String pwd){
		if("Pankaj".equals(user) &&
"J@urnalD$v".equals(pwd)) isAdmin=true;
		executor = new CommandExecutorImpl();
	}

	@Override
	public void runCommand(String cmd) throws
Exception {
		if(isAdmin){
			executor.runCommand(cmd);
		}else{

if(cmd.trim().startsWith("rm")){
				throw new
```

```
Exception("rm command is not allowed for non-admin
users.");
                        }else{

executor.runCommand(cmd);
                        }
                }
        }

}
```

Proxy Design Pattern Client Program

ProxyPatternTest.java

```
package com.journaldev.design.test;

import com.journaldev.design.proxy.CommandExecutor;
import
com.journaldev.design.proxy.CommandExecutorProxy;

public class ProxyPatternTest {

        public static void main(String[] args){
                CommandExecutor executor = new
CommandExecutorProxy("Pankaj", "wrong_pwd");
                try {
                        executor.runCommand("ls
-ltr");
                        executor.runCommand(" rm -rf
abc.pdf");
                } catch (Exception e) {
                        System.out.println("Exception
Message::"+e.getMessage());
                }

        }

}
```

Output of above proxy design pattern example program is:

```
'ls -ltr' command executed.
Exception Message::rm command is not allowed for non-
admin users.
```

Proxy design pattern common uses are to control access or to provide a wrapper implementation for better performance. Java RMI package uses proxy pattern. That's all for proxy design pattern in java.