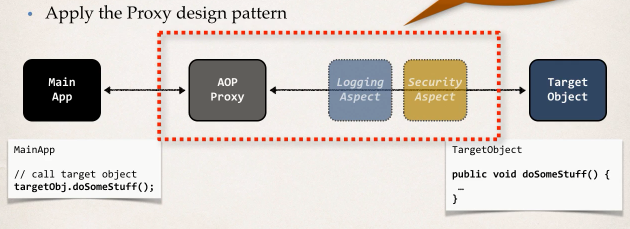
**AOP Solution and AOP Use Cases**

**AOP Solution**:

Now behand scenes, one possible solution with AOP is to apply the proxy design pattern.



So, we have our main application on the left, it's going to call the target object on the right. So, our main app will say target object to do some stuff. Our target object has that method: doSomeStuff().

The main application has no idea about AOP. The main application has no idea about Aspect, or any proxies. They are simple going to make a method call.

**Benefits of AOP**:

1. **Code for Aspect is defined in a single class**
   1. Much better than being scattered everywhere
   2. Promotes code reuse and easier to change
2. **Business code in your application is cleaner**
   1. Only applies to business functionality: addAccount()
   2. Reduces code complexity
3. **Configurable**
   1. Based on configuration, apply Aspects selectively to different parts of app
   2. No need to make changes to main application code … very important!

**Additional AOP Use Cases**:

1. **Most common**
   1. Logging, security, transactions
2. **Audit logging**
   1. who, what, when, where for a given method that's called?
3. **Exception handling**
   1. Log exception and notify DevOps team via SMS/email
4. **API Management**
   1. How many times has a method been called by user?
   2. Analytics: what are peak times? what is average load? who is top user?

**AOP: Advantages and Disadvantages**:

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| * Reusable modules * Resolve code tangling * Resolve code scatter * Apply selectively based on configuration | * Too many aspects and app flow are hard to follow * Minor performance cost for aspect execution (run-time weaving) |

**Note**:

Weaving is the process of linking aspects with other outsider application types or objects to create an advised object. This can be done at compile time (using the AspectJ compiler, for example), load time, or at runtime. Spring AOP, like other pure Java AOP frameworks, performs weaving at runtime only.

2. AOP Solution and AOP Use Cases