职责链模式
Chain of Responsibility

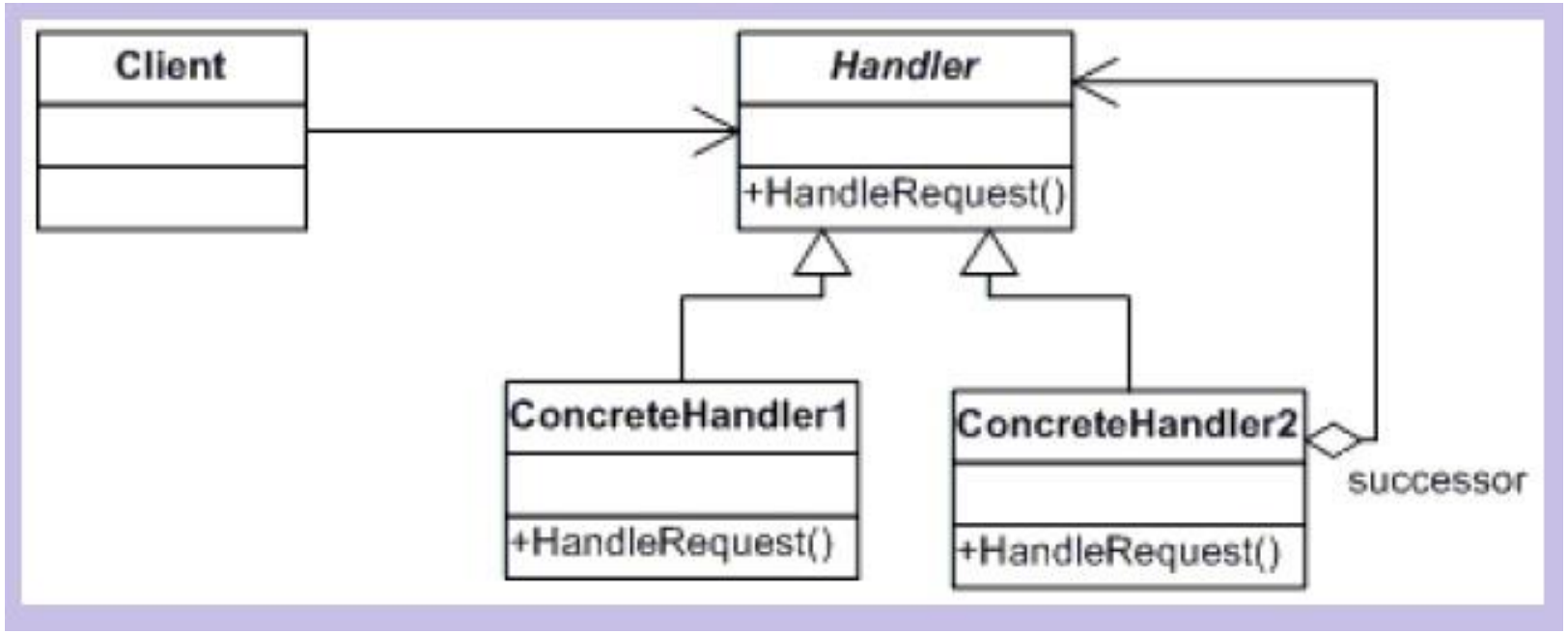
波波老师~研发总监/资深架构师

定义

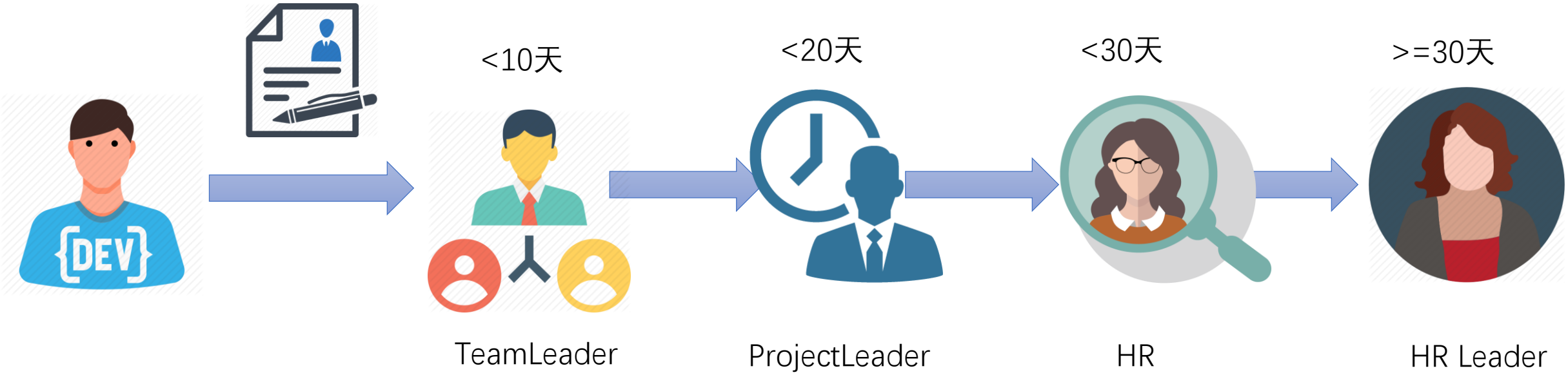
- 通过让多个对象依次处理请求的方式，将请求的接收者和发送者解耦
- 将接收对象组织成链状结构，将请求在链中依次传递，直到某个对象能够处理请求。

关系图

角色	职责
Handler	所有能处理请求的类要实现的公共接口或超类
ConcreteHandler	能够处理或者向下传递请求的类，实现Handler
Client	将ConcreteHandler组装成职责链； 向职责链发起请求



案例需求~请假审批流程



代码~Handler接口

```java
package io.spring2go.corespring;

// Handler
public interface ApproveHandler {

    public void setNextHandler(ApproveHandler nextHandler);

    public void approve(Leave leave);

}
```

代码~Leave

```java
// 请求对象
public class Leave {
    private int leaveId;
    private int numberOfDays;

    public Leave(int leaveId, int numberOfDays) {
        this.leaveId = leaveId;
        this.numberOfDays = numberOfDays;
    }

    // region getter/setter
}
```

代码~TeamLeader

```java
// ConcreteHandler
public class TeamLeader implements ApproveHandler {

    private ApproveHandler nextHandler;

    public final static int MAX_LEAVES_CAN_APPROVE = 10;

    public void setNextHandler(ApproveHandler nextHandler) {
        this.nextHandler = nextHandler;
    }

    public void approve(Leave leave) {
        if (leave.getNumberOfDays() < MAX_LEAVES_CAN_APPROVE) {
            String output = String.format(
                    "LeaveId: %d, Days: %d, Approver: %s",
                    leave.getLeaveId(),
                    leave.getNumberOfDays(),
                    "TeamLeader");
            System.out.println(output);
        } else {
            if (nextHandler != null) {
                nextHandler.approve(leave);
            }
        }
    }
}
```

代码~ProjectLeader

```
//ConcreteHandler
public class ProjectLeader implements ApproveHandler {

    private ApproveHandler nextHandler;

    public final static int MAX_LEAVES_CAN_APPROVE = 20;

    public void setNextHandler(ApproveHandler nextHandler) {
        this.nextHandler = nextHandler;
    }

    public void approve(Leave leave) {
        if (leave.getNumberOfDays() < MAX_LEAVES_CAN_APPROVE) {
            String output = String.format(
                    "LeaveId: %d, Days: %d, Approver: %s",
                    leave.getLeaveId(),
                    leave.getNumberOfDays(),
                    "ProjectLeader");
            System.out.println(output);
        } else {
            if (nextHandler != null) {
                nextHandler.approve(leave);
            }
        }
    }
}
```

代码~HR

```java
//ConcreteHandler
public class HR implements ApproveHandler {

    private ApproveHandler nextHandler;

    public final static int MAX_LEAVES_CAN_APPROVE = 30;

    public void setNextHandler(ApproveHandler nextHandler) {
        this.nextHandler = nextHandler;
    }

    public void approve(Leave leave) {
        if (leave.getNumberOfDays() < MAX_LEAVES_CAN_APPROVE) {
            String output = String.format(
                    "LeaveId: %d, Days: %d, Approver: %s",
                    leave.getLeaveId(),
                    leave.getNumberOfDays(),
                    "HR");
            System.out.println(output);
        } else {
            if (nextHandler != null) {
                nextHandler.approve(leave);
            } else {
                System.out.println("Leave application suspended, Please contact HR");
            }
        }
```

代码~Client

```java
// Client
public class ClientProgram {

    public static void main(String[] args) {
        TeamLeader tl = new TeamLeader();
        ProjectLeader pl = new ProjectLeader();
        HR hr = new HR();

        tl.setNextHandler(pl);
        pl.setNextHandler(hr);


        tl.approve(new Leave(1, 5));
        tl.approve(new Leave(2, 15));
        tl.approve(new Leave(3, 25));
        tl.approve(new Leave(4, 35));

    }

}
```

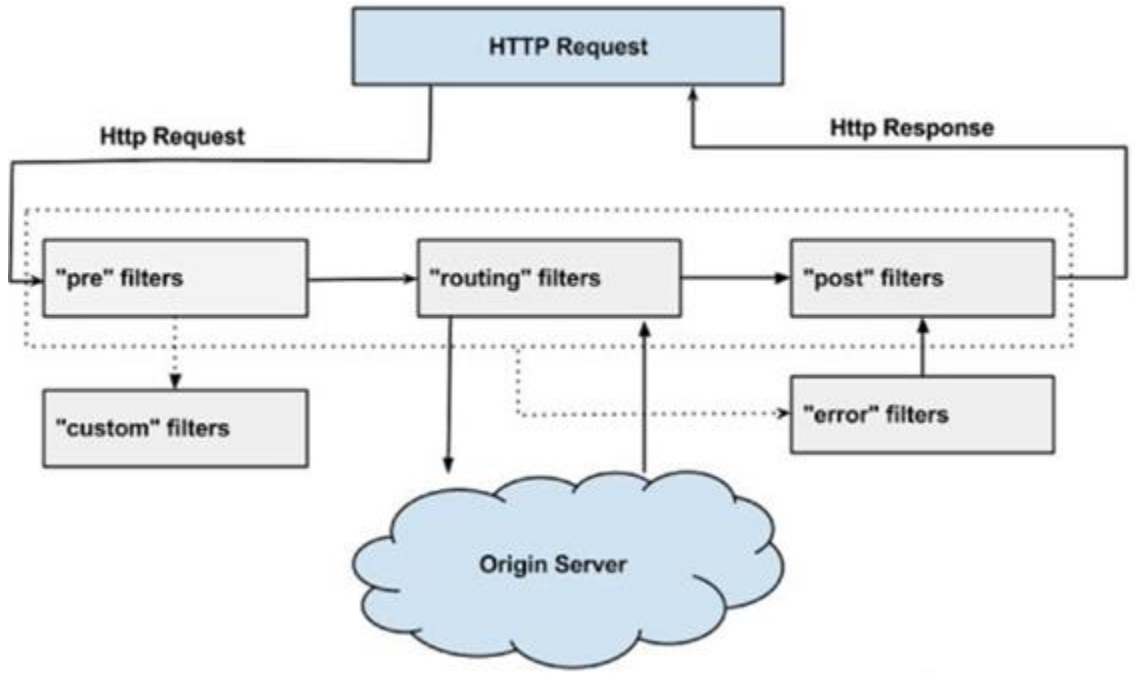
```
LeaveId: 1, Days: 5, Approver: TeamLeader
LeaveId: 2, Days: 15, Approver: ProjectLeader
LeaveId: 3, Days: 25, Approver: HR
Leave application suspended, Please contact HR
```

好处

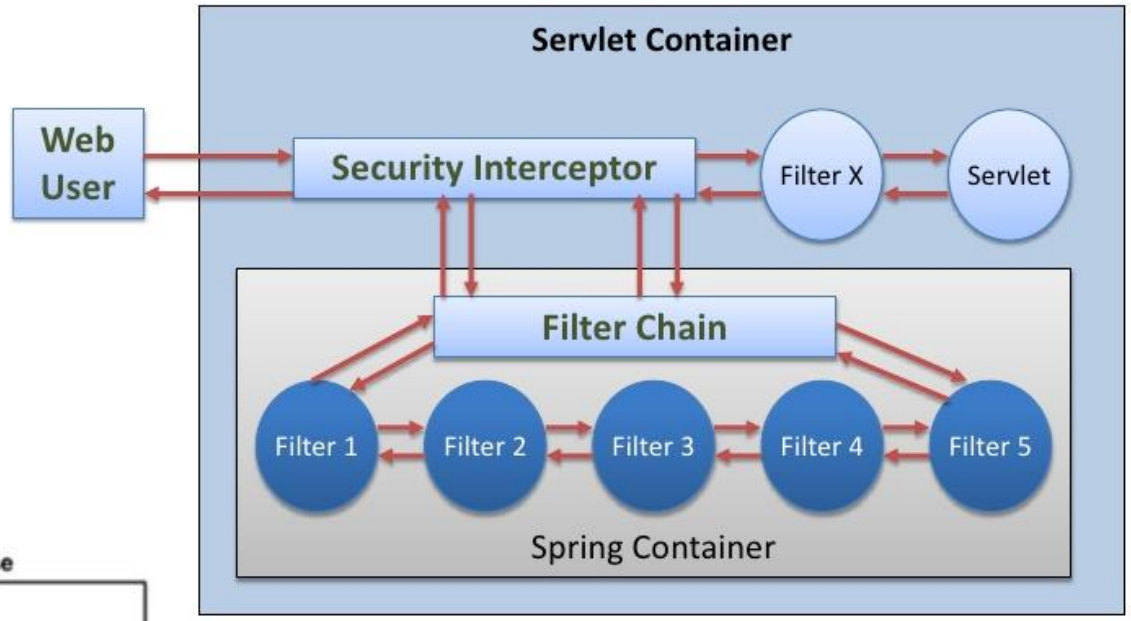
- 发送者和接收者对象解耦
- 使用组合（Composite），对象职责的增删改更灵活

应用

- Spring Security
- 微服务网关过滤器



Flow of a request through Spring Security's core filters



课后练习

- 优化案例代码，抽取公共功能到抽象超类

参考

- Understanding and Implementing Chain of Responsibility Pattern in C#
  - https://www.codeproject.com/Articles/494241/Understanding-and-Implementing-Chain-of-Responsibi
- Chain of Responsibility Design Pattern in Java
  - https://www.journaldev.com/1617/chain-of-responsibility-design-pattern-in-java

代码

- https://github.com/spring2go/core-spring-patterns

波波微课

spring2go.com