FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

SCHEDULE TASK & INTERCEPTOR

GIẢNG VIÊN:

www.poly.edu.vn

AGENDA

- SCHEDULE TASKS
  - TÌM HIỂU SCHEDULE TASKS
  - TẠO SCHEDULE TASKS
  - ÁP DỤNG SCHEDULE TASKS ĐỂ GIẢI QUYẾT
    - XỬ LÝ RÁC TRONG CSDL
    - XẾP MAIL VÀO HÀNG ĐỢI
    - BACKUP DATABASE…
- INTERCEPTOR
  - TÌM HIỂU INTERCEPTOR
  - TẠO INTERCEPTOR
  - ÁP DỤNG INTERCEPTOR ĐỂ GIẢI QUYẾT
    - GHI NHẬN THÔNG TIN CÁC REQUEST
    - BẢO MẬT HỆ THỐNG

Schedule Task

What is Schedule Task ?

- Trong Spring Boot, Schedule Task là cơ chế cho phép thực hiện các công việc được xếp lịch sẵn.
- Schedule Task thường đảm nhận những công việc phía hậu trường
  - Gửi email theo định kỳ, chiến dịch
  - Xóa dữ liệu
  - Vô hiệu hóa dữ liệu hết hiệu lực
  - Sao lưu dữ liệu…
- Với bản chất đó, Schedule Task tham gia vào việc chống ùn tắc các nhiệm vụ tốn nhiều thời gian rất hiệu quả, như gửi email.
  - Nhiệm vụ gửi email tốn nhiều thời gian, nếu nhiều user cùng thực hiện chức năng này cùng một thời điểm thì hệ thống rất dễ dẫn đến quá tải. Vì vậy cần xếp công việc gửi email vào hàng đợi, Schedule Task sẽ thực hiện từ từ ở phía hậu trường.

@EnableScheduling

- Bước 1: Kích hoạt chế độ Schedule Task bằng @EnableScheduling trên file cấu hình chính của ứng dụng

```
@SpringBootApplication
@EnableScheduling
public class HelloApplication {...}
```

- Bước 2: Đặt lịch thực hiện cho một phương thức của một Spring Bean nào đó với @Scheduled

```
@Scheduled(fixedRate = 1000)
public void run() {...}
```

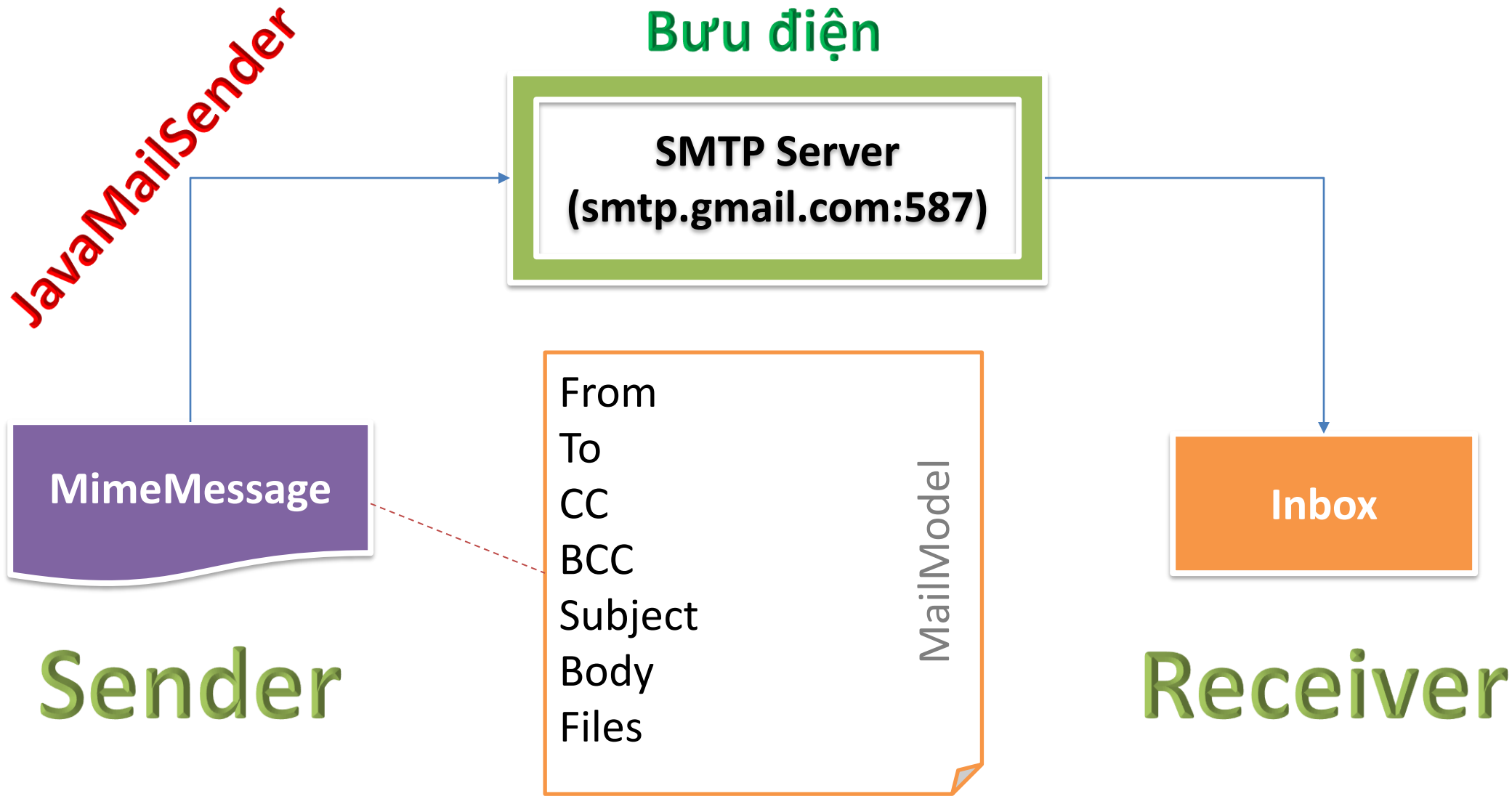
- @Scheduled() được sử dụng để lên lịch trình cho một phương thức Spring Bean thực hiện
- @Scheduled() có các đối số chia thành 3 nhóm
  - Delay: Khoảng thời gian (mili giây) giữa kết thúc hoạt động trước và bắt đầu hoạt động sau
    - fixedDelay: long
    - fixedDelayString: String
  - Rate: Khoảng thời gian (mili giây) cố định giữa các lần thực hiện
    - fixedRate: long
    - fixedRateString: String
  - CRON: Biểu thức lịch trình thực hiện tùy biến
    - Cú pháp: "* * * ? * * *" = [giây] [phút] [giờ] ? [ngày] [tháng] [năm]
    - CRON có tính mềm dẻo nhưng thực sự phức tạp. Để xây dựng biểu thức này hãy sử dụng https://www.freeformatter.com/cron-expression-generator-quartz.html

@Scheduled Examples

- @Scheduled(fixedDelay = 3000, initialDelay = 5000)
  - Thời gian chờ hoạt động tiếp sau là 3 giây
  - Sau khi khởi động 5 giây thì lịch trình bắt đầu
- @Scheduled(fixedRate = 3000, initialDelay = 5000)
  - Thời gian giữa các hoạt động là 3 giây
  - Sau khi khởi động 5 giây thì lịch trình bắt đầu
- @Scheduled(cron = "0 * * * * MON-FRI“, initialDelay = 5000)
  - Giây đầu tiên của mỗi phút các ngày trong tuần (thứ 2 đến thứ 6)
  - Sau khi khởi động 5 giây thì lịch trình bắt đầu

Java Mail Sender

Email Sending Model



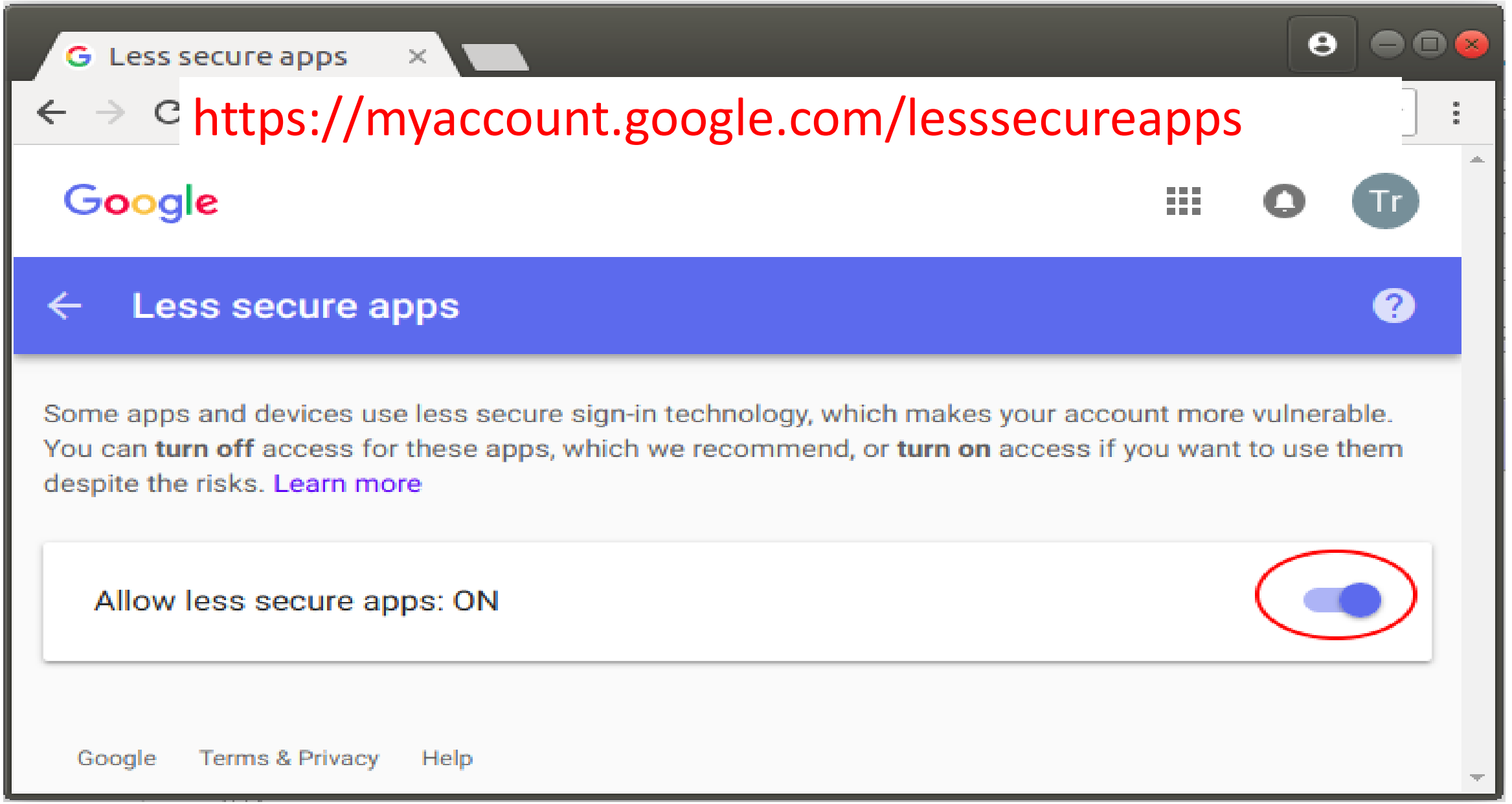
JavaMailSender Configuration

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
</dependency>
```

```
#GMail Configuration for JavaMailSender
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=<user@gmail.com>
spring.mail.password=<******>
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
```

Tài khoản gmail cần phải được kích hoạt mới được phép sử dụng gửi email qua ứng dụng (slide sau)

ACTIVATING GMAIL ACCOUNT



SENDING EMAIL

```
@Autowired
JavaMailSender sender;

@RequestMapping("/mail/send")
public void send(){
        MimeMessage message = sender.createMimeMessage();
        // ...viết thư  (slide sau)
        sender.send(message); // bỏ vào bưu điện
}
```

*Viết thư là công việc cung cấp vào **message** những thông tin như: **from, to, cc, bcc, subject, body, attachment files***

MIMEMESSAGEHELPER

```
MimeMessageHelper helper = new MimeMessageHelper(message, true, "utf-8");
helper.setFrom(mail.getFrom());
helper.setTo(mail.getTo());
helper.setSubject(mail.getSubject());
helper.setText(mail.getBody(), true);
helper.setReplyTo(mail.getFrom());
for(String email: mail.getCc()) {
    helper.addCc(email);
}
for(String email: mail.getBcc()) {
    helper.addBcc(email);
}
for(File file: mail.getFiles()) {
    helper.addAttachment(file.getName(), file);
}
```

Lớp MimeMessageHelper giúp cung cấp những thông tin cần thiết vào message một cách nhanh chóng và đơn giản.

*Biến **mail** trong minh họa này là đối tượng của MailModel ở slide sau*

MAILMODEL

```
@Data @NoArgsConstructor @AllArgsConstructor
public class MailModel {
    String from = "FPT Polytechnic <poly@gmail.com>";
    String to;
    String subject;
    String body;
    List<String> cc = new ArrayList<>();
    List<String> bcc = new ArrayList<>();
    List<File> files = new ArrayList<>();
    public MailModel(String to, String subject, String body) {
        super();
        this.to = to;
        this.subject = subject;
        this.body = body;
    }
}
```

Lớp này nhằm mô tả thông tin đầy đủ nhất của một email.

Trong thực tế, đôi khi chỉ cần cung cấp người nhận, tiêu đề và nội dung.

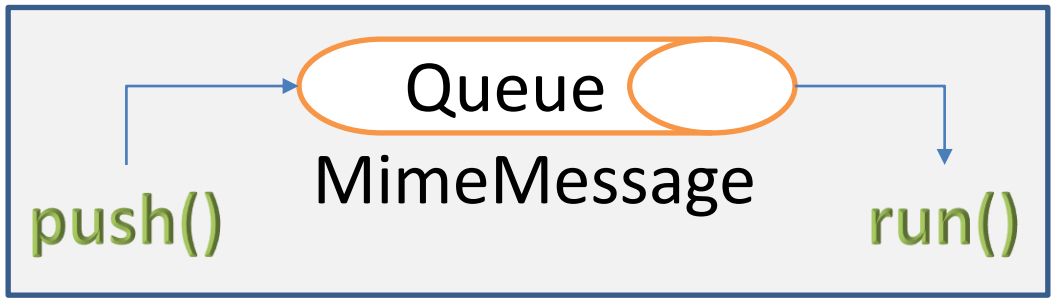
XẾP HÀNG ĐỢI EMAIL

MailerService

```
@Service
public class MailerService {
    @Autowired
    JavaMailSender sender;
    List<MimeMessage> queue = new ArrayList<>();

    public void push(String to, String subject, String body) {
        MailModel mail = new MailModel(to, subject, body);
        this.push(mail);
    }
    public void push(MailModel mail) {...}

    @Scheduled(fixedDelay = 1000)
    public void run() {...}
}
```



*Tạo MimeMessage và bỏ vào **queue***

*Thực hiện lấy và gửi tất cả email trong **queue** định kỳ sau mỗi giây.*

PUSH() IMPLEMENTATION

```
MimeMessage message = sender.createMimeMessage();
MimeMessageHelper helper = new MimeMessageHelper(message, true, "utf-8");
helper.setFrom(mail.getFrom());
helper.setTo(mail.getTo());
helper.setSubject(mail.getSubject());
helper.setText(mail.getBody(), true);
helper.setReplyTo(mail.getFrom());
for(String email: mail.getCc()) {
        helper.addCc(email);
}
for(String email: mail.getBcc()) {
        helper.addBcc(email);
}
for(File file: mail.getFiles()) {
        helper.addAttachment(file.getName(), file);
}
queue.add(message);
```

RUN() IMPLEMENTATION

```
int success = 0, error = 0;
while(!queue.isEmpty()) {
        MimeMessage message = queue.remove(0);
        try {
                sender.send(message);
                success++;
        } catch (Exception e) {
                error++;
        }
}
System.out.printf(">> Sent: %d, Error: %d\r\n", success, error);
```

Lấy MimeMessage từ queue

Gửi MimeMessage đến SMTP Server

SỬ DỤNG MAILERSERVICE

```
@Autowired
MailerService mailer;

@ResponseBody
@RequestMapping("/mailer/send")
public void send(){
        mailer.push("user1@gmail.com", "Subject 1", "Content 1");
        mailer.push("user2@gmail.com", "Subject 2", "Content 2");
}
```

Tiêm MailerService => gọi push() để đưa vào queue.

FPT Education

FPT POLYTECHNIC

cdio™

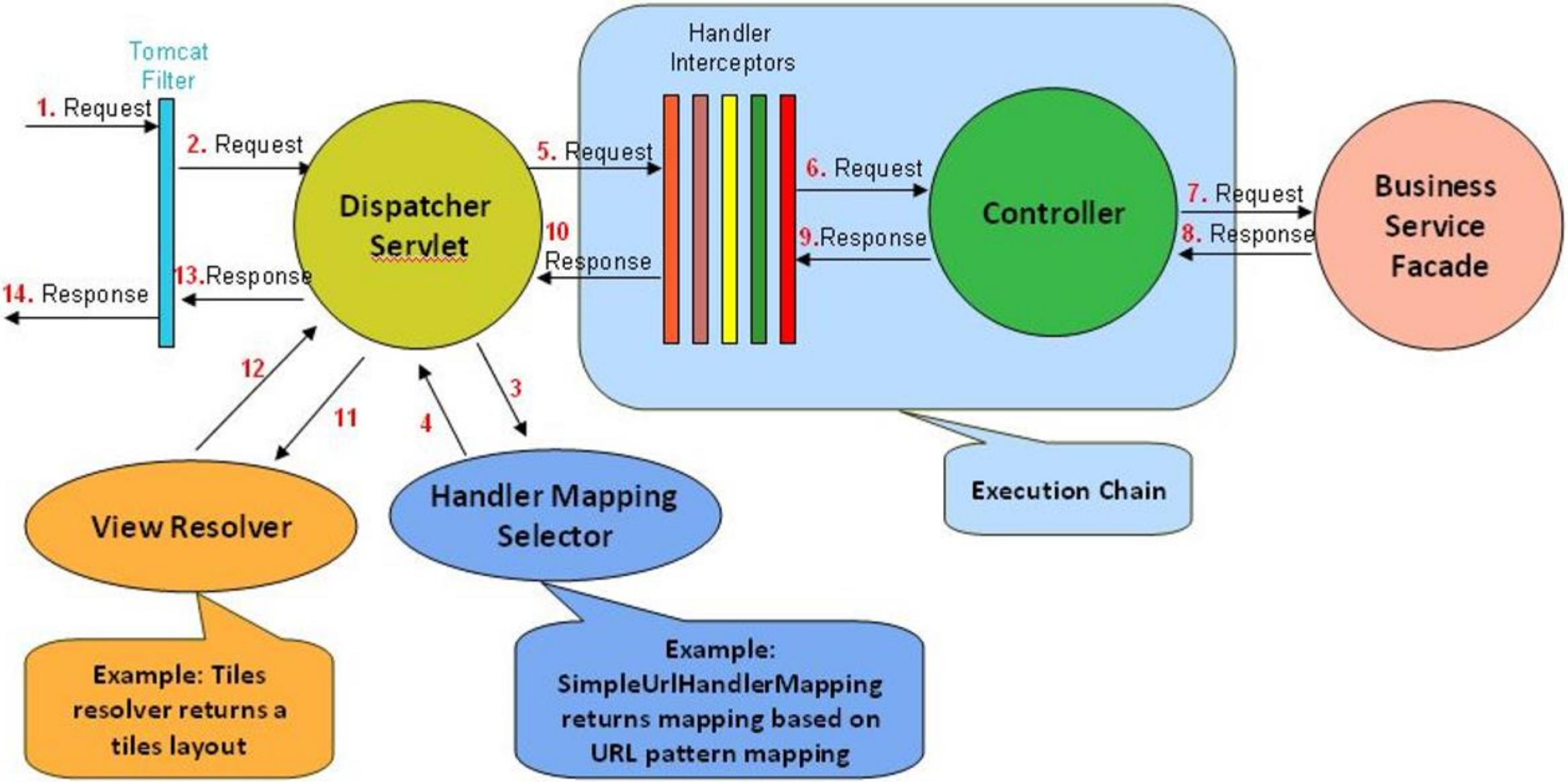
Conceive Design Implement Operate

INTERCEPTOR

GIẢNG VIÊN:

www.poly.edu.vn

INTERCEPTOR

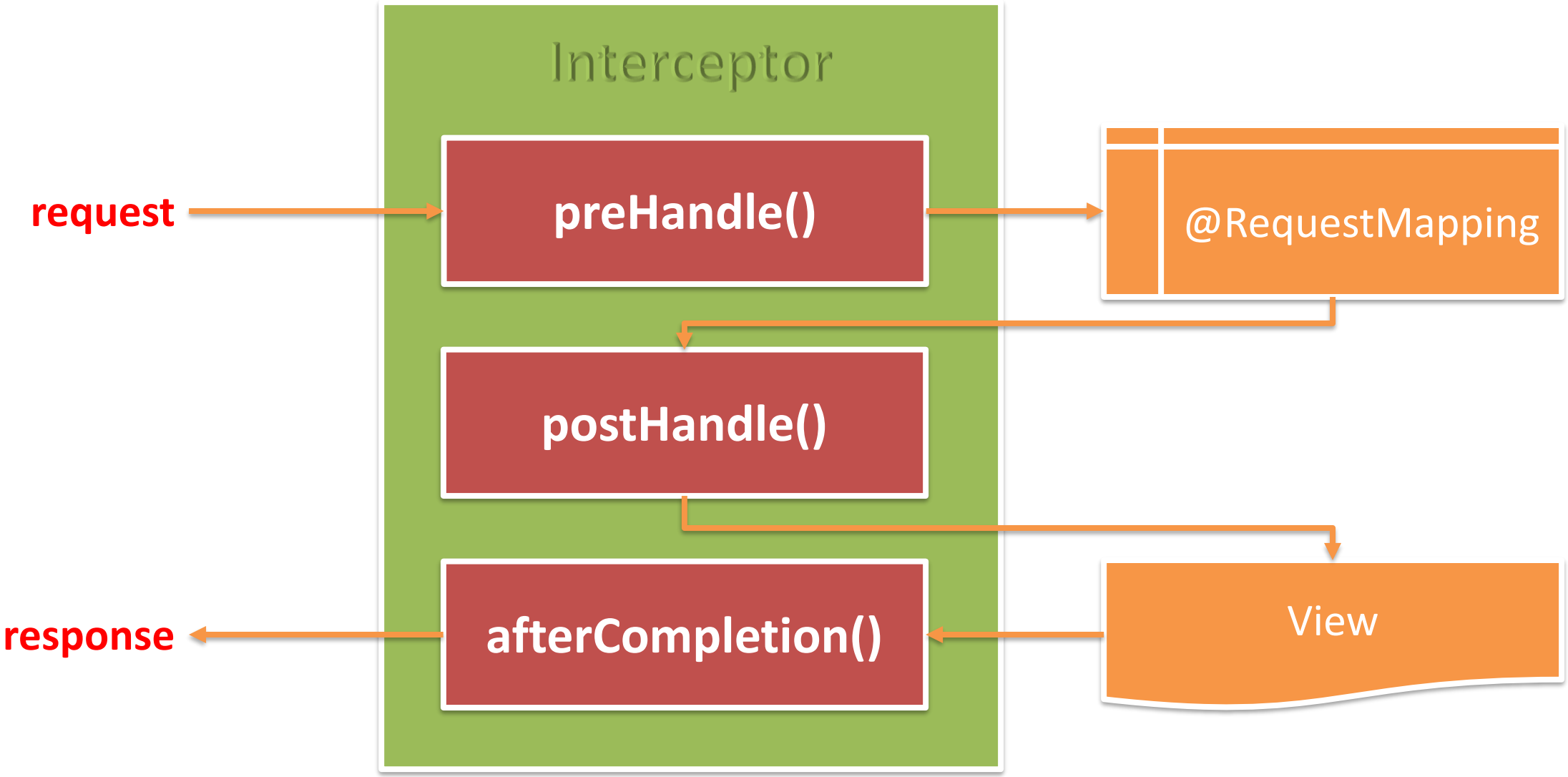


INTERCEPTOR CONSTRUCTURE

```
@Component
public class MyInterceptor implements HandlerInterceptor {
    @Override
    public boolean preHandle(HttpServletRequest req,
            HttpServletResponse resp, Object handler) throws Exception {
        return true;
    }
    @Override
    public void postHandle(HttpServletRequest req, HttpServletResponse resp,
            Object handler, ModelAndView modelAndView) throws Exception {
    }
    @Override
    public void afterCompletion(HttpServletRequest req,
            HttpServletResponse resp, Object handler, Exception ex) throws Exception {
    }
}
```

true: cho đi tiếp đến controller

Interceptor Handling Flow



LoggerInterceptor

```
@Component
public class LoggerInterceptor implements HandlerInterceptor {
    @Override
    public boolean preHandle(HttpServletRequest req,
            HttpServletResponse resp, Object handler) throws Exception {
        System.out.println("preHandle()->" + req.getRequestURI());
        return true;
    }
    @Override
    public void postHandle(HttpServletRequest req, HttpServletResponse resp,
            Object handler, ModelAndView modelAndView) throws Exception {
        System.out.println("postHandle()->" + req.getRequestURI());
    }
    @Override
    public void afterCompletion(HttpServletRequest req,
            HttpServletResponse resp, Object handler, Exception ex) throws Exception {
        System.out.println("afterCompletion()->" + req.getRequestURI());
    }
}
```

INTERCEPTOR CONFIGURATION

- Cấu hình interceptor là để chỉ ra các địa chỉ URL nào bị lọc bởi interceptor.

```
@Configuration
public class InterceptorConfig implements WebMvcConfigurer{
    @Autowired
    LoggerInterceptor interceptor;
    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(interceptor)
            .addPathPatterns("/**")
            .excludePathPatterns("/assets/**");
    }
}
```

Lọc tất cả các URI ngoại trừ các URI bắt đầu bởi /assets/

Authentication Interceptor

AUTHENTICATION CONFIGURATION

```
@Configuration
public class InterceptorConfig implements WebMvcConfigurer{
    @Autowired
    AuthInterceptor interceptor;
    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(interceptor)
            .addPathPatterns("/order/**", "/account/change", "/account/edit/**")
            .addPathPatterns("/admin/**")
            .excludePathPatterns("/assets/**", "/admin/home/index");
    }
}
```

AUTHENTICATION WITH INTERCEPTOR

```
@Component
public class AuthInterceptor implements HandlerInterceptor {
    @Override
    public boolean preHandle(HttpServletRequest req,
            HttpServletResponse resp, Object handler) throws Exception {
        if(req.getSession().getAttribute("user") == null) {
            req.getSession().setAttribute("secureUri", req.getRequestURI());
            resp.sendRedirect("/account/login");
            return false;
        }
        return true;
    }
}
```

SUMMARY

- ☑ SCHEDULE TASKS
  - ☑ TÌM HIỂU SCHEDULE TASKS
  - ☑ TẠO SCHEDULE TASKS
  - ☑ ÁP DỤNG SCHEDULE TASKS ĐỂ GIẢI QUYẾT
    - ☑ XỬ LÝ RÁC TRONG CSDL
    - ☑ XẾP MAIL VÀO HÀNG ĐỢI
    - ☑ BACKUP DATABASE…
- ☑ INTERCEPTOR
  - ☑ TÌM HIỂU INTERCEPTOR
  - ☑ TẠO INTERCEPTOR
  - ☑ ÁP DỤNG INTERCEPTOR ĐỂ GIẢI QUYẾT
    - ☑ GHI NHẬN THÔNG TIN CÁC REQUEST
    - ☑ BẢO MẬT HỆ THỐNG


FPT Education
FPT POLYTECHNIC
Thank you
TIN HỌC VĂN PHÒNG QUỐC TẾ THEO CHUẨN ICDL
Syllabus 5 - Office 2007
LUẬT KINH TẾ
KỸ THUẬT SOẠN THẢO VÀ TRÌNH BÀY VĂN BẢN