FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

SCHEDULE TASK & INTERCEPTOR

SLIDE 8.1

www.poly.edu.vn

AGENDA

- PHẦN I: SCHEDULE TASKS
  - TÌM HIỂU SCHEDULE TASKS
  - TẠO SCHEDULE TASKS
  - JAVAMAILSENDER
  - GỬI EMAIL Ở CHẾ ĐỘ NỀN
- PHẦN II: INTERCEPTOR
  - GIỚI THIỆU INTERCEPTOR
  - TẠO INTERCEPTOR
  - BẢO MẬT HỆ THỐNG VỚI INTERCEPTOR

Scheduled Task

SCHEDULED TASK INTRODUCTION

- Scheduled Task là cơ chế cho phép thực hiện các công việc được xếp lịch sẵn.
- Scheduled Task thường đảm nhận những công việc phía hậu trường (chạy nền trên thread độc lập)
- Scheduled Task giúp chống ùn tắc các nhiệm vụ tốn nhiều thời gian để nâng cao hiệu suất ứng dụng.
- Các công việc thường có nhu cầu chạy theo cơ chế này:
  - Gửi email theo định kỳ, chiến dịch
  - Xóa hoặc vô hiệu hóa dữ liệu
  - Sao lưu cơ sở dữ liệu…

@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 phương thức của Spring Bean nào đó với @Scheduled

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

@SCHEDULED

- @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

JAVA MAIL SENDER

- JavaMailSender là API cung cấp các hoạt động gửi email được hỗ trợ bởi Spring Boot.
- Dependency

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

- Bạn cũng cần khai báo các thông số SMTP Server để tạo Spring Bean JavaMailSender trước khi sử dụng (slide sau)

JavaMailSender Configuration

```
#JavaMailSender - https://myaccount.google.com/apppasswords
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=user@gmail.com
spring.mail.password=apppassword
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.ssl.protocols=TLSv1.2
```

- Khai báo các thông tin vào file application.properties để cấu hình tạo Spring Bean JavaMailSender
- Vào liên kết [https://myaccount.google.com/apppasswords] tạo và chép apppassword

GỬI MAIL

```
@Autowired
JavaMailSender mailSender;

public void send() {
        try {
                // 1. Tạo Mail
                MimeMessage mimeMessage = mailSender.createMimeMessage();

                // 2. Hỗ trợ tạo nội dung Mail
                MimeMessageHelper helper = new MimeMessageHelper(mimeMessage, true, "utf-8");
                // 2.1. Ghi thông tin người gửi
                // 2.2. Ghi thông tin người nhận
                // 2.3. Ghi tiêu đề và nội dung Mail
                // 2.4. Đính kèm file
                // 3. Gửi Mail
                mailSender.send(mimeMessage);
        } catch (Exception e) {
                throw new RuntimeException(e);
        }
}
```

Căn cứ vào thông tin cấu hình của file application.properties, hệ thống sẽ tạo và nạp bean JavaMailSender vào hệ thống, bạn chỉ việc tiêm vào và sử dụng khi cần

TẠO NỘI DUNG MAIL

```
String from = "FPT Polytechnic <poly@fpt.edu.vn>";
helper.setFrom(from);
helper.setReplyTo(from);
```

2.1 Ghi thông tin người gửi

```
helper.setTo("instructor@gmail.com");
helper.setCc("student1@gmail.com, student2@gmail.com");
helper.setBcc("manager1@gmail.com, manager2@gmail.com");
```

2.2 Ghi thông tin người nhận

```
helper.setSubject("Welcome to FPT Polytechnic");
helper.setText("I love you so much", true);
```

2.3 Ghi tiêu đề và nội dung

```
String[] filenames = "C:/files/1.doc, C:/files/f2.png".split("[,;]+");
for(String filename: filenames) {
        File file = new File(filename.trim());
        helper.addAttachment(file.getName(), file);
}
```

2.4 Đính kèm file

THIẾT KẾ MAILSERVICE

```
public interface MailService{
    @Data
    @Builder
    public static class Mail{
        @Default
        private String from = "WebShop <web-shop@gmail.com>";
        private String to, cc, bcc, subject, body, filenames;
    }

    void send(Mail mail);

    default void send(String to, String subject, String body) {
        Mail mail = Mail.builder().to(to).subject(subject).body(body).build();
        this.send(mail);
    }
}
```

CÀI ĐẶT MÃ NGUỒN CHO MAILSERVICE

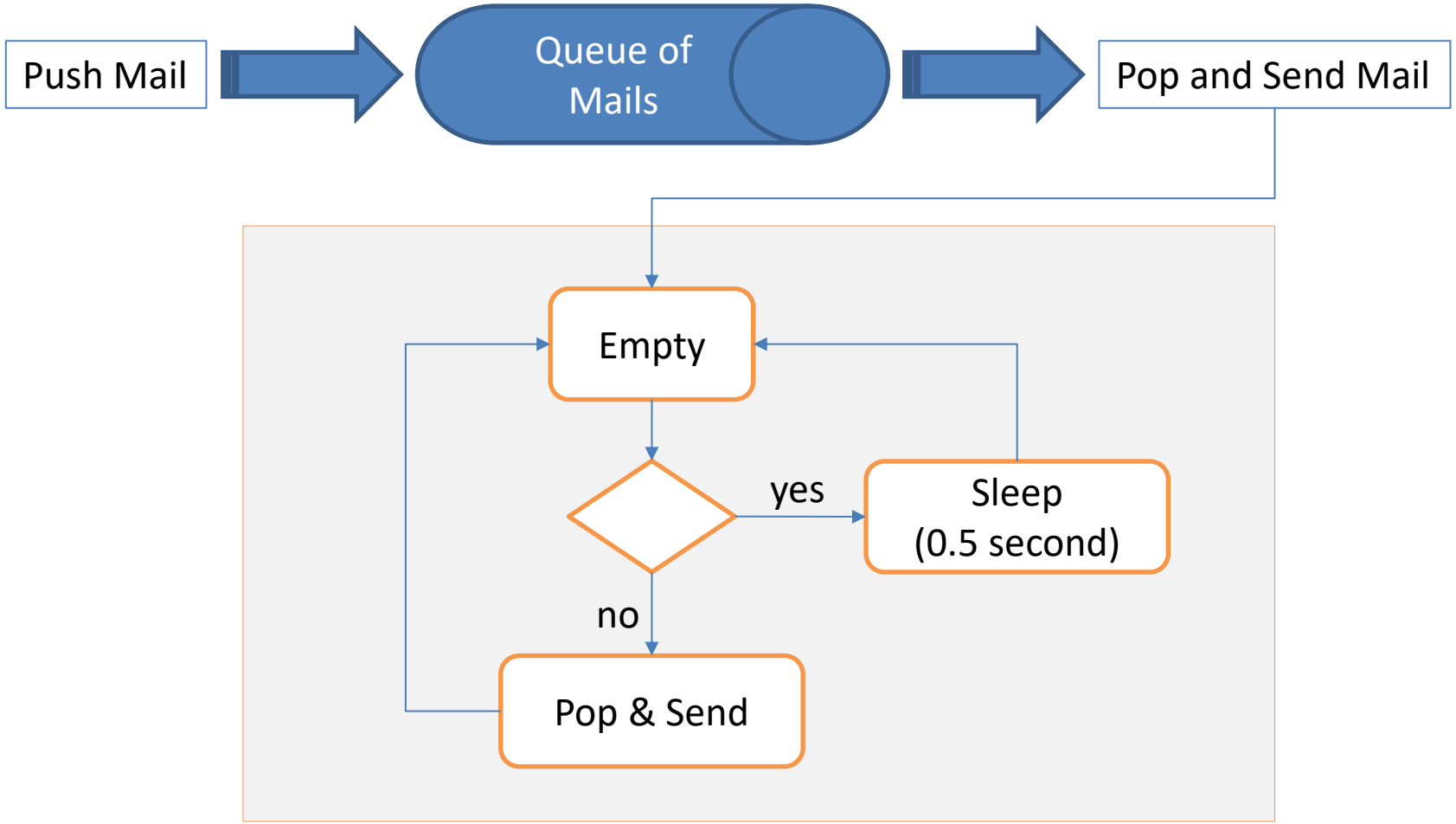
```
@Service("mailService")
public class MailServiceImpl implements MailService{
	@Autowired
	JavaMailSender mailSender;

	public void send(Mail mail) {
		try {
			MimeMessage message = mailSender.createMimeMessage();

			MimeMessageHelper helper = new MimeMessageHelper(message, true, "utf-8");
			/*--Dựa vào dữ liệu của Mail để hoàn thành mã tạo nội dung mail ở đây--*/
			mailSender.send(message);
		} catch (Exception e) {
			throw new RuntimeException(e);
		}
	}
}
```

SENDING MAIL IN BACKGROUND

- Gửi email mất nhiều thời gian làm giảm hiệu năng phục vụ các yêu cầu vì vậy cần chuyển công việc gửi email chạy nền



SỬ DỤNG SCHEDULED TASK

```
void push(Mail mail);
default void push(String to, String subject, String body){
        this.push(Mail.builder().to(to).subject(subject).body(body).build());
}
```

Bổ sung vào interface **MailService**

Bổ sung vào class **MailServiceImpl**

```
List<Mail> queue = new ArrayList<>();

public void push(Mail mail){
        queue.add(mail);
}

@Scheduled(fixedDelay = 500)
public void run() {
        while(!queue.isEmpty()) {
                this.send(queue.remove(0));
        }
}
```

FPT Education

FPT POLYTECHNIC

cdio™

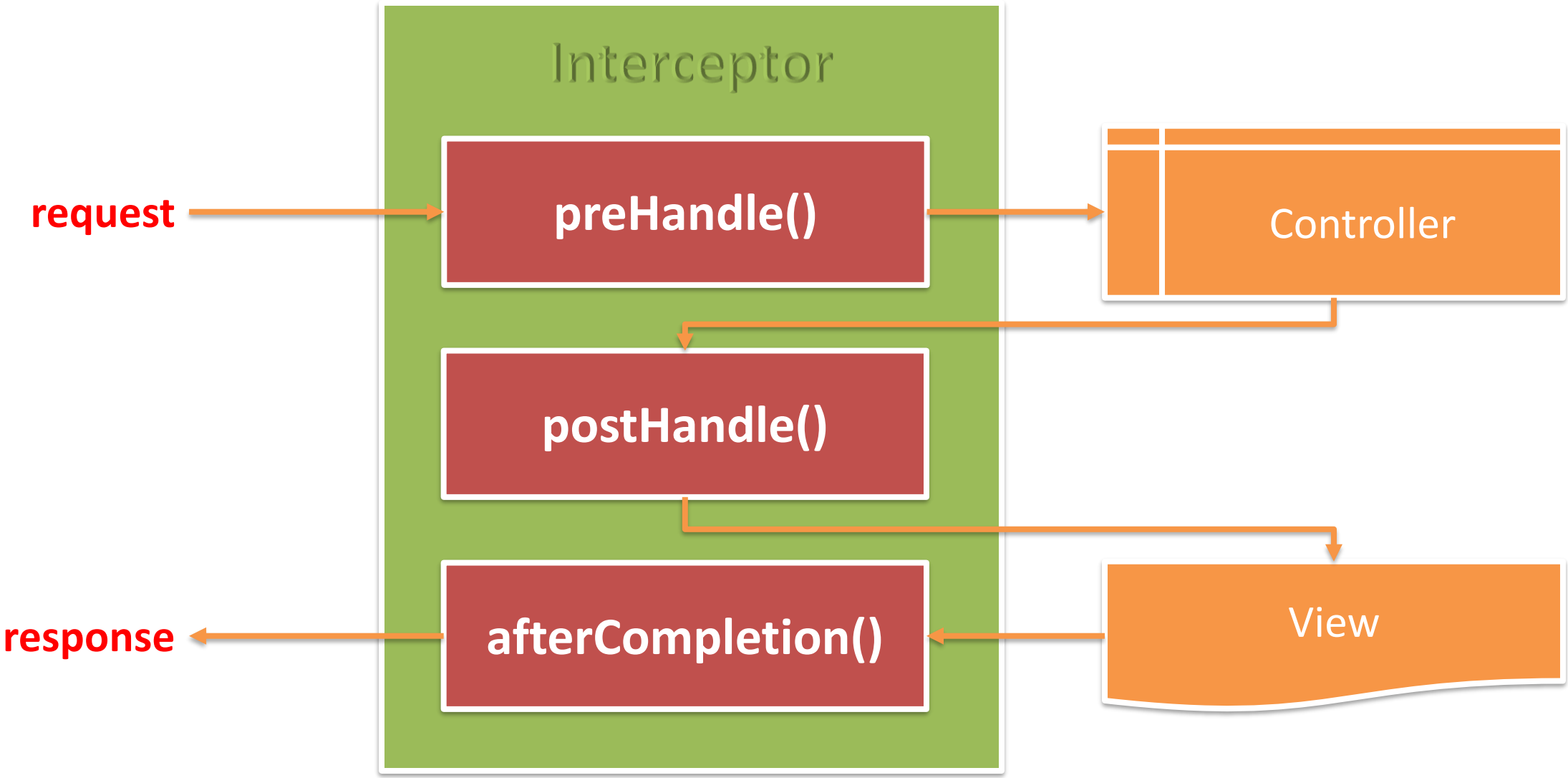
Conceive Design Implement Operate

INTERCEPTOR

SLIDE 8.2

www.poly.edu.vn

Interceptor Handling Flow



INTERCEPTOR STRUCTURE

```
@Component
public class MyInterceptor implements HandlerInterceptor{
        @Override // chạy trước khi đến controller method
        public boolean preHandle(HttpServletRequest request,  HttpServletResponse response,
                        Object handler) throws Exception {
                return true; // chuyển tiếp đến controller method, false sẽ dừng lại
        }
        @Override // chạy sau khi controller method return nhưng trước khi đến view
        public void postHandle(HttpServletRequest request, HttpServletResponse response,
                        Object handler, ModelAndView modelAndView) throws Exception {
        }
        @Override // chạy sau khi view đã được render
        public void afterCompletion(HttpServletRequest request, HttpServletResponse response,
                        Object handler, Exception ex) throws Exception {
        }
}
```

INTERCEPTOR CONFIGURATION

- Cấu hình Interceptor là để kích hoạt nó khi người dung truy xuất đến các URI cụ thể nào đó.
- Với cấu hình sau thì myInterceptor được kích hoạt khi người dung truy xuất đến tất cả các URI của website ngoại trừ /home/index.

```
@Configuration
public class InterceptorConfig implements WebMvcConfigurer {
    @Autowired
    MyInterceptor myInterceptor;
    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(myInterceptor)
            .addPathPatterns("/**")
            .excludePathPatterns("/home/index");
    }
}
```

INTERCEPTOR EXAMPLE 1

```
@Component
public class LogInterceptor implements HandlerInterceptor{
        @Override
        public boolean preHandle(HttpServletRequest request,
                        HttpServletResponse response, Object handler) throws Exception {
                System.out.println(new Date() + ": " +request.getRequestURI());
                return true;
        }
}
```

```
@Configuration
public class InterceptorConfig implements WebMvcConfigurer {
        @Autowired
        LogInterceptor logInterceptor;
        @Override
        public void addInterceptors(InterceptorRegistry registry) {
                registry.addInterceptor(logInterceptor)
                        .addPathPatterns("/admin/**").excludePathPatterns("/admin/home/index");
        }
}
```

INTERCEPTOR EXAMPLE 2

```
@Service
public class AppInterceptor implements HandlerInterceptor{
        @Autowired
        CategoryService categoryService;
        @Override
        public void postHandle(HttpServletRequest request, HttpServletResponse response,
                        Object handler, ModelAndView modelAndView) throws Exception {
                if(modelAndView != null) {
                        modelAndView.addObject("categories", categoryService.findAll());
                }
        }
}
```

- *Chú ý: Hành động cung cấp data cho view dung chung bởi nhiều controller chúng ta cần phải viết mã trong postHandle() để đảm bảo dữ liệu luôn luôn đúng cho dù trong controller có làm thay đổi.*

Authentication Interceptor

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;
    }
}
```

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");
    }
}
```

SUMMARY

- ☑ SCHEDULE TASKS
  - ☑ TÌM HIỂU SCHEDULE TASKS
  - ☑ TẠO SCHEDULE TASKS
  - ☑ JAVAMAILSENDER
  - ☑ GỬI EMAIL Ở CHẾ ĐỘ NỀN
- ☑ INTERCEPTOR
  - ☑ GIỚI THIỆU INTERCEPTOR
  - ☑ TẠO INTERCEPTOR
  - ☑ BẢO MẬT HỆ THỐNG VỚI INTERCEPTOR


FPT Education
FPT POLYTECHNIC
Thank you