



R a i s i n g t h e b a r

Spring AOP & Exception Handle

Mục tiêu

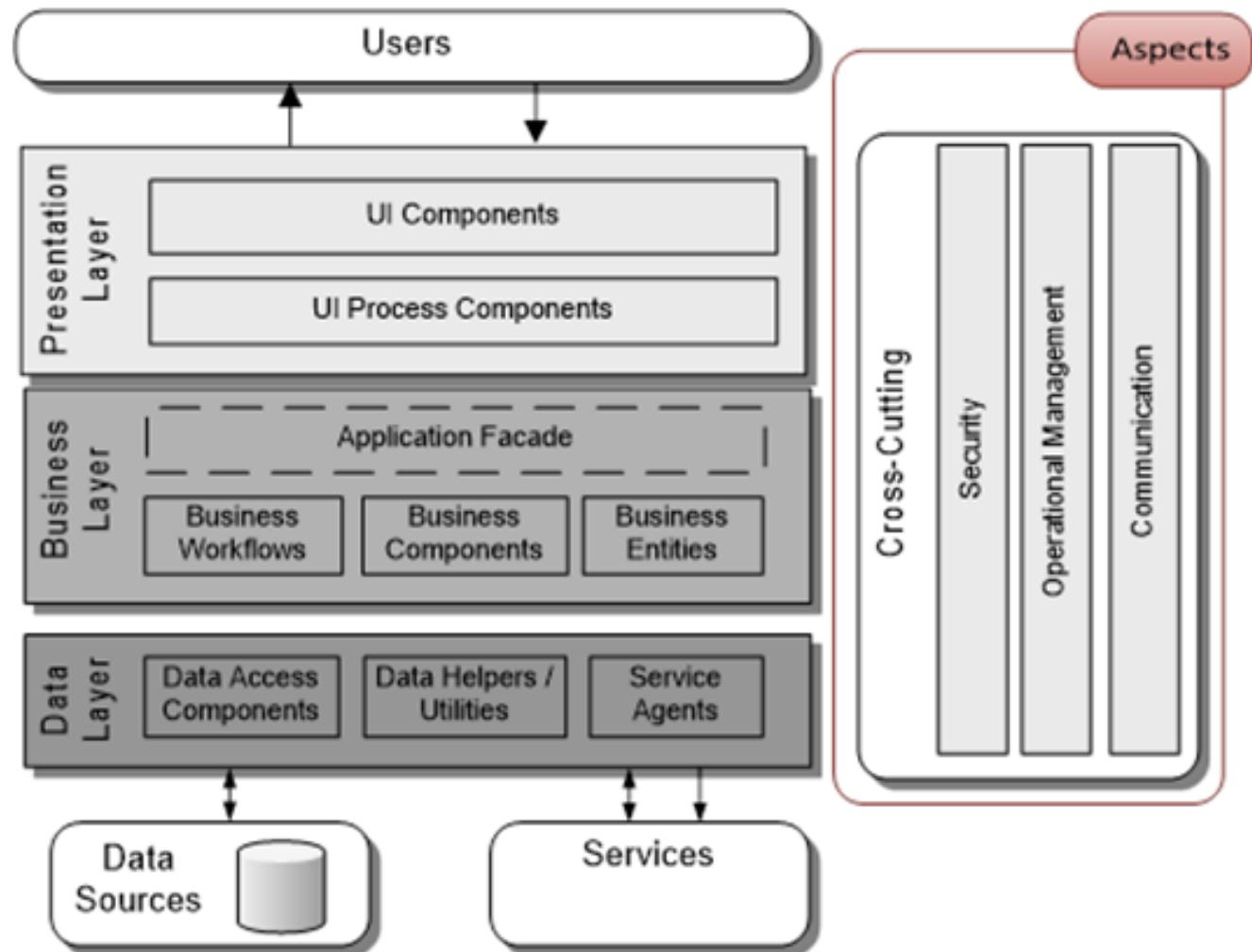
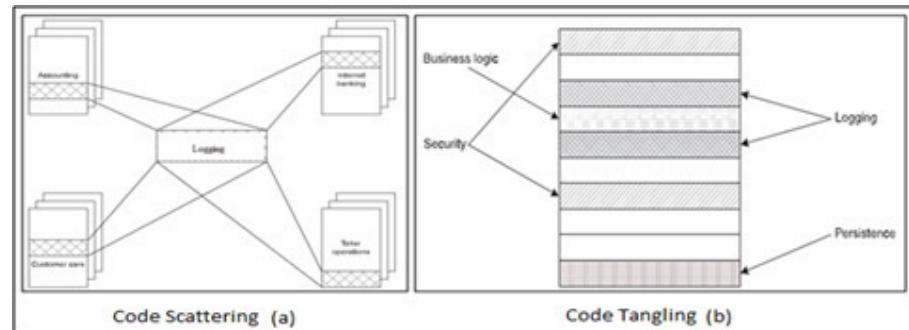
- Trình bày được về các Aspect
- Trình bày được kỹ thuật AOP
- Trình bày được về các thuật ngữ quan trọng trong AOP
- Triển khai được kỹ thuật AOP trong ứng dụng Spring
- Trình bày được tầm quan trọng của xử lý ngoại lệ trong ứng dụng Spring MVC
- Phân loại được các ngoại lệ trong ứng dụng Spring MVC
- Trình bày được các phương án xử lý ngoại lệ trong ứng dụng Spring MVC
- Sử dụng được try-catch để xử lý ngoại lệ cho ứng dụng Spring MVC
- Sử dụng được cơ chế ExceptionHandler để xử lý ngoại lệ cho ứng dụng Spring MVC

Spring AOP

Cross-cutting Concerns

Những mối quan tâm xuyên suốt, mà trong AOP được gọi bằng thuật ngữ “Aspect”.

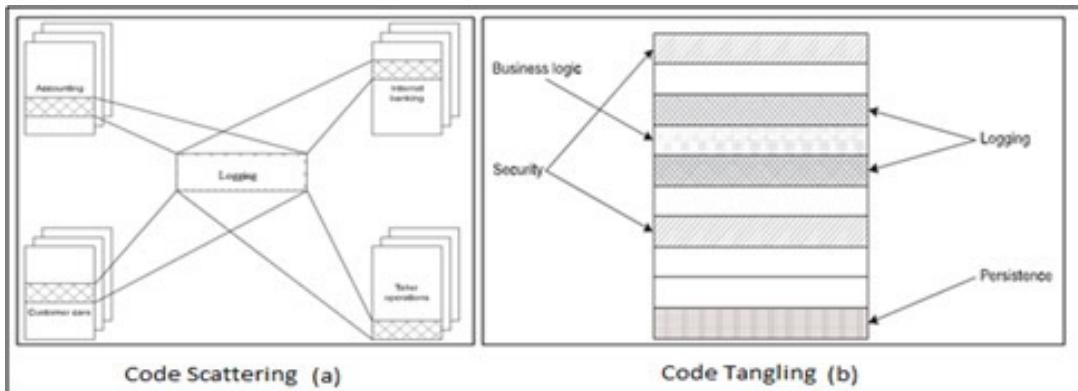
Có thể gây ra trạng thái “dính (a)” và “rối (a)” cho mã. Cản trở việc phát triển sản phẩm.

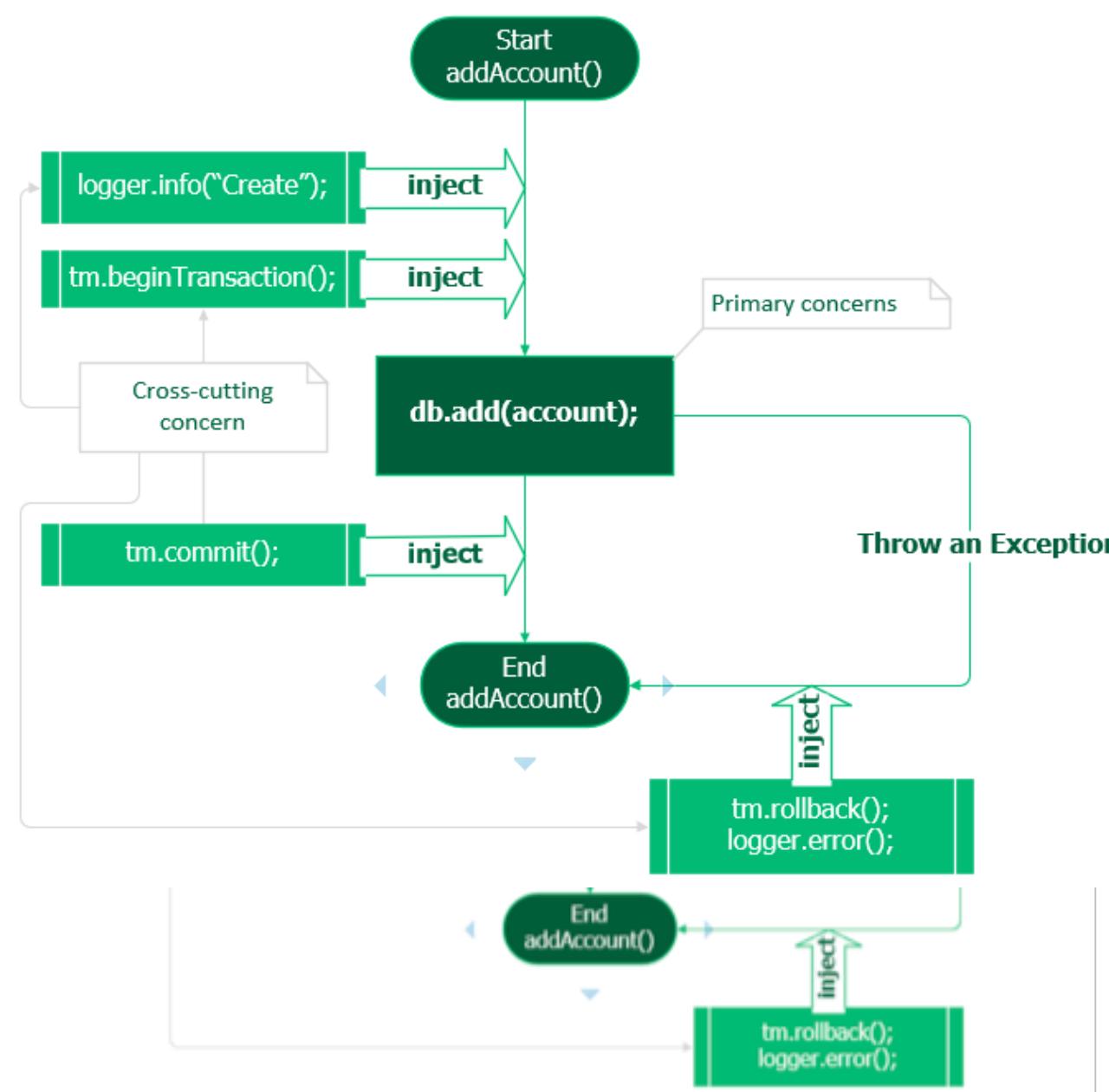


Cross-cutting Concerns (2)

Có thể gây ra trạng thái xấu cho mã:

- **Scattering** (phân tán): tạo ra mã lặp
- **Tangling** (rối): khiến cho các khía cạnh khác nhau bị cuốn vào một nơi





Imming

- Một hệ tư tưởng lập trình mà trong đó các mối quan tâm xuyên suốt được cô lập khỏi nghiệp vụ chính của chương trình.
- Các mối quan tâm xuyên suốt tham gia vào nghiệp vụ chính (bằng thao tác **weave** - đan) thông qua các **advice** (lời khuyên) mà không làm thay đổi mã đó.

Aspect (khía cạnh)

- Một thuật ngữ trong AOP, chỉ “mối quan tâm xuyên suốt”.
- Là một khối mã có lập có khả năng tái sử dụng.
- Do được cô lập khỏi nghiệp vụ chính, aspect giúp mã nghiệp vụ chính trở nên dễ hiểu hơn.

Advice

Một hành động cụ thể của aspect.

Join Point (điểm gia nhập)

- Một vị trí nào đó trong luồng thực thi chương trình mà **advice** sẽ được **weave** (đan vào).
- Các join point phổ biến bao gồm: vị trí đọc/ghi giá trị của field, vị trí gọi/thực thi một phương thức/constructor, hay vị trí mà một ngoại lệ được tung ra.
- Join point cần được thiết kế kỹ.

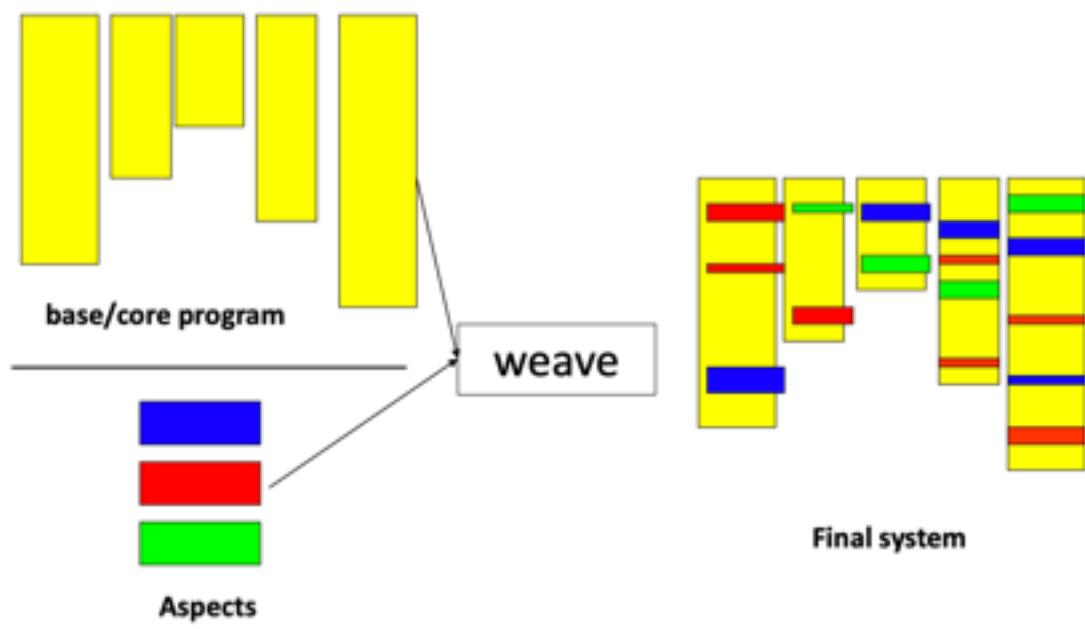
Point Cut

Chỉ ra một tập hợp các **join point** để công cụ AOP đan các advice vào nghiệp vụ chính.

Các loại advice

- **Before Advice:** thực thi trước join point.
- **After Returning Advice:** thực thi sau khi joint point hoàn thành bình thường
- **After Throwing Advice:** thực thi nếu joint point kết thúc do có ngoại lệ được tung
- **After (finally) Advice:** thực thi sau khi joint point hoàn thành bất kể dưới điều kiện nào
- **Around Advice:** thực thi cả trước và sau joint point

Weave (đan)



Tiến trình cài đặt các advice vào mã nghiệp vụ chính. Được thực hiện vào **thời điểm compile hoặc trễ hơn**, chẳng hạn khi nạp mã đã compile vào môi trường thực thi, hay ngay trong thời điểm thực thi.

AspectJ

- Một phần mở rộng cho Java, trao cho lập trình viên Java khả năng lập trình hướng khía cạnh.
- Được tích hợp chặt chẽ vào hầu hết các IDE java hiện đại
- Tương thích với tất cả các phiên bản Java 1.x

Spring AOP

- Cung cấp khả năng AOP, nhưng tương thích tốt hơn với Spring IoC Container so với các phương án triển khai AOP khác.
- Tiến trình weave được thực hiện khi IoC Container xây dựng các bean.
- Không triển khai trọn vẹn AOP, chẳng hạn chỉ hỗ trợ join point tại lời gọi method
- Hỗ trợ sử dụng các annotation của Aspect để định nghĩa các chủ thể trong AOP (nhưng tiến trình weave vẫn được thực hiện bởi Spring AOP)

Định nghĩa một Aspect

```
package org.xyz;  
import org.aspectj.lang.annotation.Aspect;
```

```
@Aspect  
public class NotVeryUsefulAspect {  
}
```

Định nghĩa một Point Cut

```
// the pointcut expression  
@Pointcut("execution(* transfer(..))")  
// the pointcut signature  
private void anyOldTransfer() {}
```

Ngôn ngữ định nghĩa Point Cut

- *execution* - khớp những join point là những điểm thực thi phương thức
- *within* - khớp join point trong một phạm vi cụ thể, như một package hay class
- *this* - khớp join point là lời gọi phương thức mà lời gọi đó nằm trong một bean có quan hệ *is-A* với một kiểu chỉ được đưa ra
- *target* - khớp joint point là lời gọi phương thức mà phương thức đó là của một đối tượng có quan hệ *is-A* với kiểu được đưa ra
- *args* - khớp những join point có bộ tham số có quan hệ *is-A* với kiểu được đưa ra

510J Mix

Ví dụ về point cut

- `execution(public * *(..))` // lời gọi bất cứ method public nào
- `execution(* set*(..))` // lời gọi bất cứ method nào có tên bắt đầu bằng set
- `execution(* com.xyz.service.AccountService.*(..))` // lời gọi bất cứ method nào được định nghĩa bởi interface AccountService
- `execution(* com.xyz.service.*.*(..))` // lời gọi bất cứ method nào trong service package
- `execution(* com.xyz.service..*.*(..))` // lời gọi bất cứ method nào trong service package hoặc package con của nó
- `within(com.xyz.service.*)` // bất cứ join point nào trong package service
- `within(com.xyz.service..*)` // bất cứ join point nào trong package service hoặc package con

Khai báo advice

- Advice được khai báo như một phương thức trong class mô tả aspect.
- Khai báo advice đi kèm với biểu thức khai báo point cut, và kèm với kiểu của advice (thứ mô tả thời điểm trigger advice).

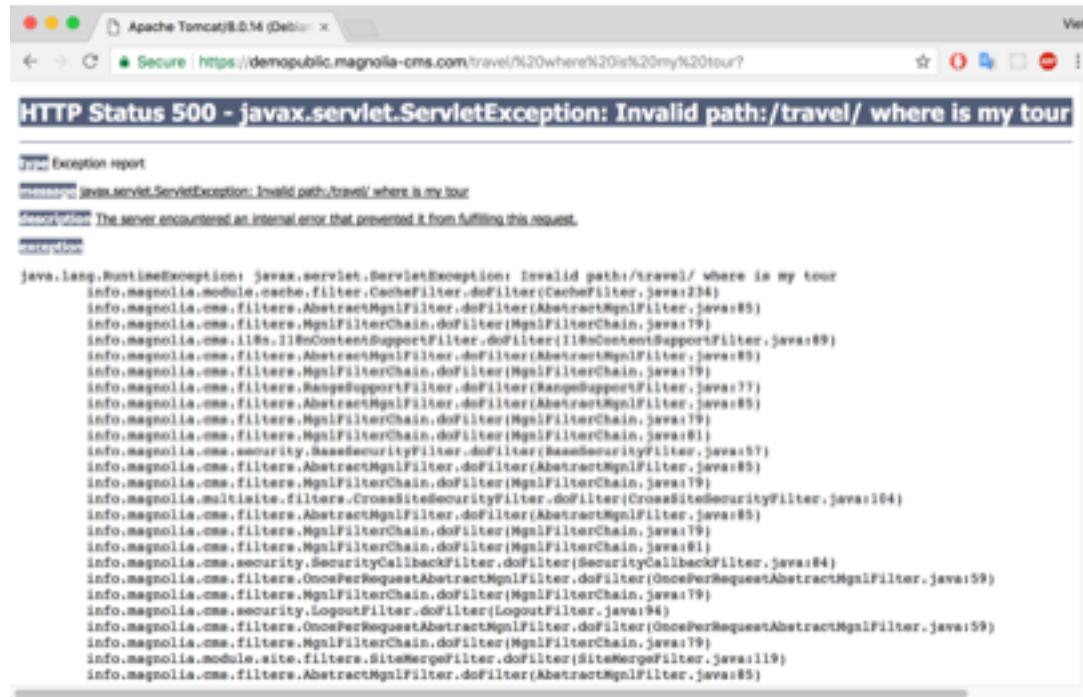
Ví dụ về khai báo advice

```
@Before("com.xyz.myapp.SystemArchitecture.dataAccessOperation()")  
public void doAccessCheck() {  
    // ...  
}  
  
@AfterThrowing(pointcut="com.xyz.myapp.SystemArchitecture.dataAccessOperation()", throwing="ex")  
public void doRecoveryActions(DataAccessException ex) {  
    // ...  
}
```

Expectation Handling

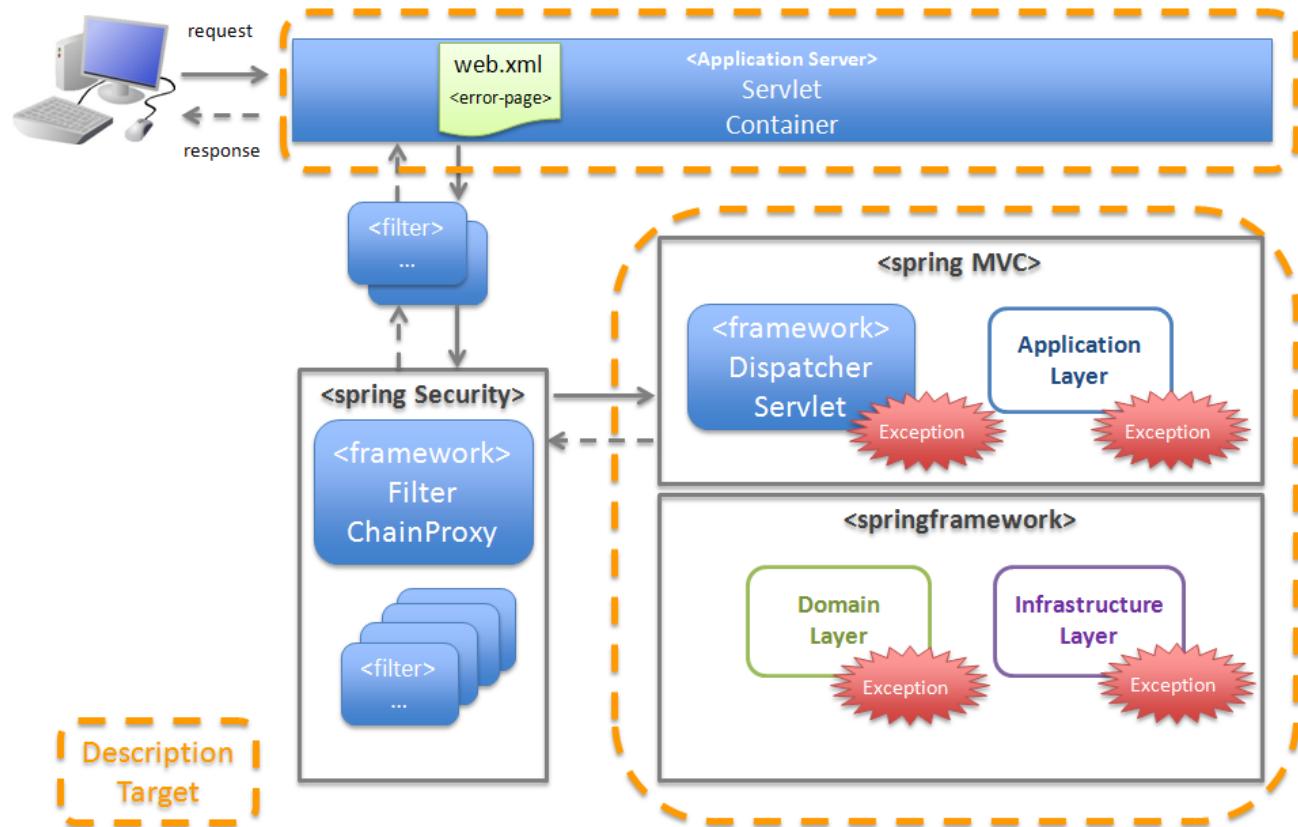
Xử lý ngoại lệ cho webapp

- Mục tiêu là không gửi thông tin ngoại lệ tới client
- Mấu chốt là bắt ngoại lệ đúng lúc và có hướng xử lý (handle) phù hợp



Biên giới hạn xử lý ngoại lệ

Ngoại lệ xảy ra trong từng phạm vi phải được xử lý trước khi luồng thực thi đi ra khỏi phạm vi đó



Phân loại ngoại lệ

Ngoại lệ mà sẽ hết gấp nếu người dùng thao tác lại	<ul style="list-style-type: none">Ngoại lệ nghiệp vụNgoại lệ từ thư viện	Được handle trong mã của webapp, bởi nhà phát triển
Ngoại lệ mà sẽ vẫn gặp lại nếu người dùng thao tác lại	<ul style="list-style-type: none">Ngoại lệ từ hệ thốngNgoại lệ không mong muốn từ hệ thốngLỗi fatal	Được handle bởi framework, bởi kiến trúc sư hệ thống
Ngoại lệ từ chối request	<ul style="list-style-type: none">Ngoại lệ từ framework	

Ngoại lệ nghiệp vụ

- Mô tả một vi phạm về luật nghiệp vụ.
- Ví dụ:
 - Chuyển một số tiền vượt quá hạn mức
 - Đặt phòng vào một ngày ở quá khứ
 - `vn.codegym.andy.common.exception.RegisterException`

Ngoại lệ từ thư viện

- Chương trình (webapp) hoạt động bình thường, nhưng thư viện tung ra exception.
- Ví dụ:
 - Thời gian chờ quá lâu
 - Chỉnh sửa dữ liệu đang bị khóa bởi một transaction khác
 - org.springframework.dao.OptimisticLockingFailureException
 - org.springframework.dao.PessimisticLockingFailureException
 - org.springframework.dao.DuplicateKeyException

Ngoại lệ từ hệ thống

- Hệ thống tung ngoại lệ thông báo một trạng thái bất thường
- Ví dụ:
 - Thư mục làm việc không tồn tại
 - Lỗi khi đọc ghi file

Ngoại lệ không mong muốn từ hệ thống

- Những ngoại lệ unchecked mà sẽ không xảy ra nếu hệ thống hoạt động bình thường
- Không được phép để webapp handle những ngoại lệ này
- Ví dụ:
 - Lỗi không được tung ra mà bị webapp, thư viện, framework dấu đi
 - org.springframework.dao.DataAccessResourceFailureException

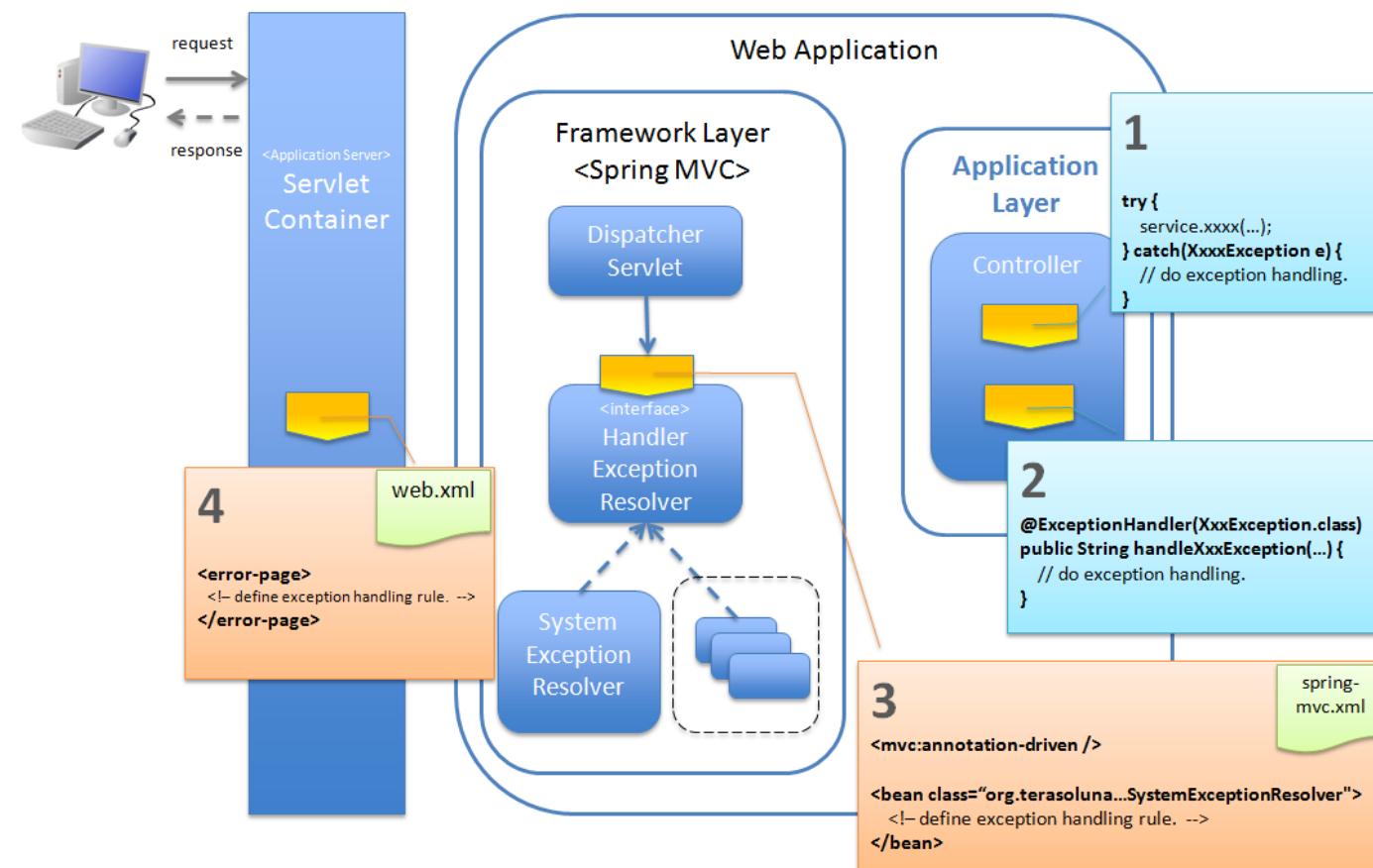
Lỗi fatal

- Những lỗi ảnh hưởng tới toàn bộ hệ thống
- Không được phép handle bởi webapp
- Ví dụ:
 - `java.lang.OutOfMemoryError`

Ngoại lệ từ chối request

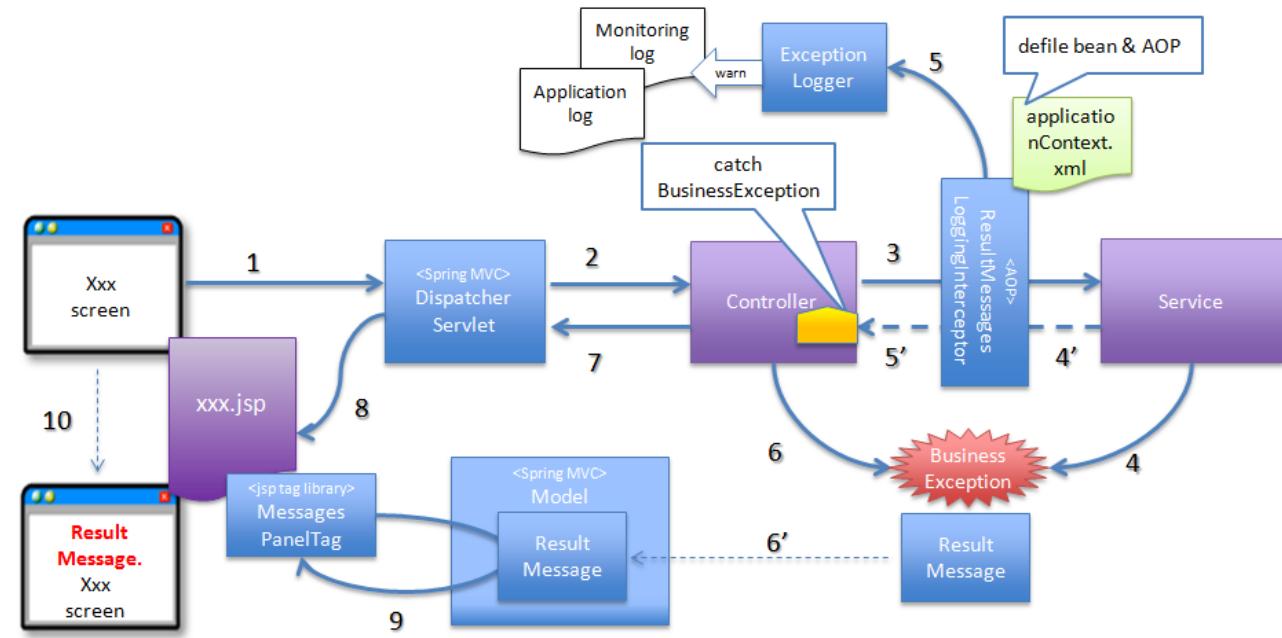
- Framework từ chối xử lý request
- Ví dụ:
 - Đường dẫn không tồn tại
 - Method không được chấp nhận
 - Không thể chuyển đổi param sang định dạng yêu cầu
 - org.springframework.web.HttpRequestMethodNotSupportedException
 - org.springframework.beans.TypeMismatchException

Các phương thức xử lý ngoại lệ



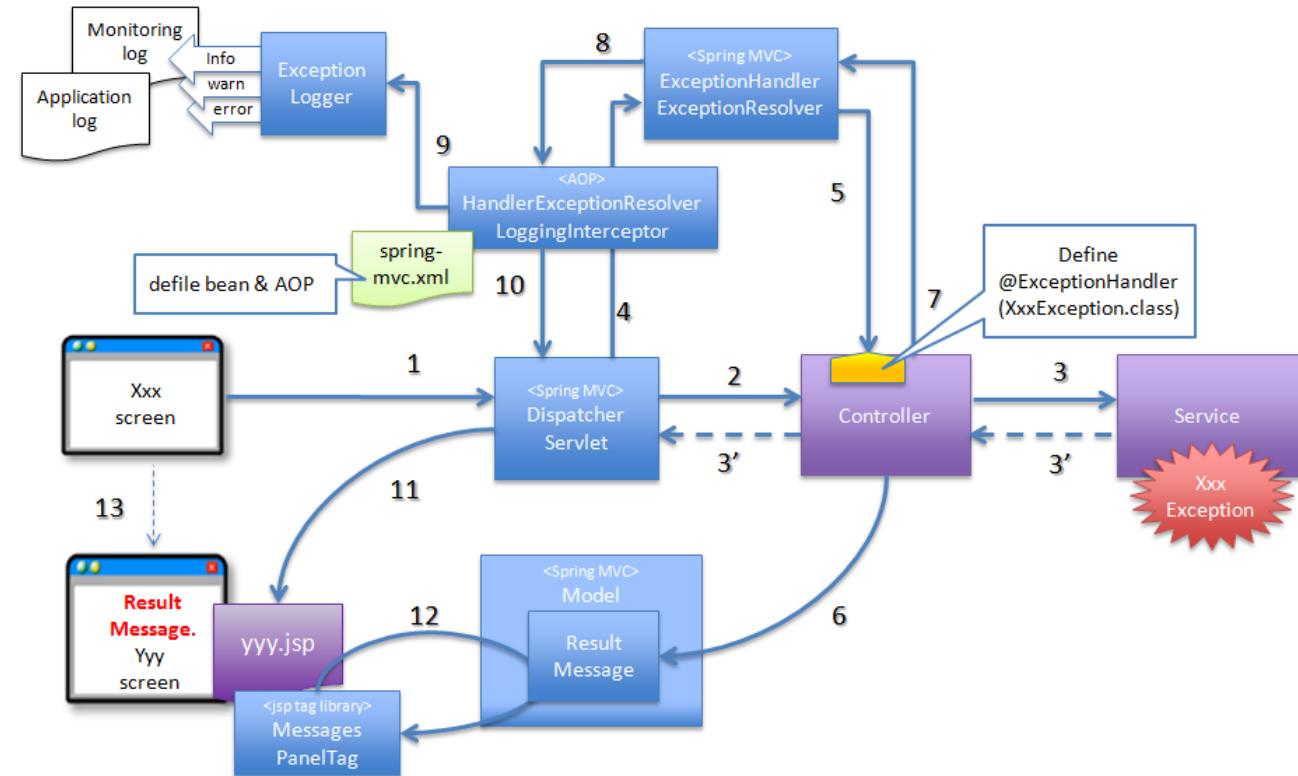
Sử dụng try-catch

- Xử lý tại controller
- Dùng để xử ngoại lệ nghiệp vụ
- Được viết bởi nhà phát triển



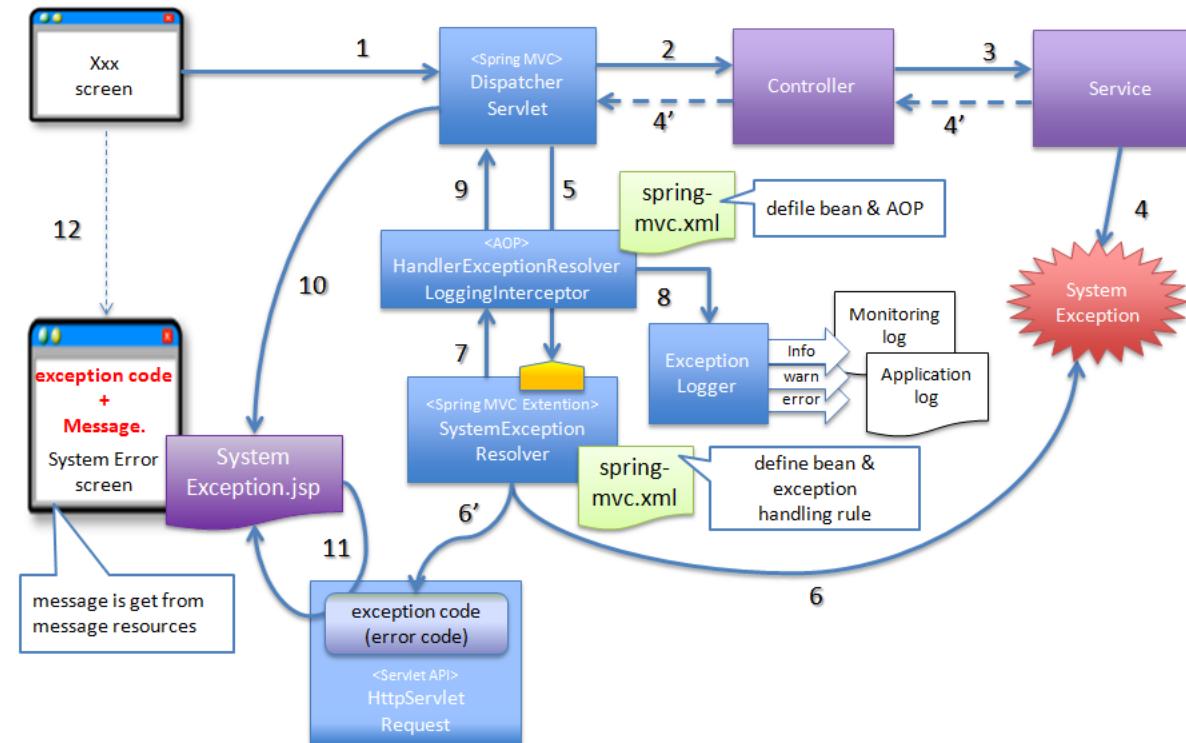
Sử dụng @ExceptionHandler

- Xử lý tại controller
- Dùng để bắt ngoại lệ nghiệp vụ
- Được viết bởi nhà phát triển



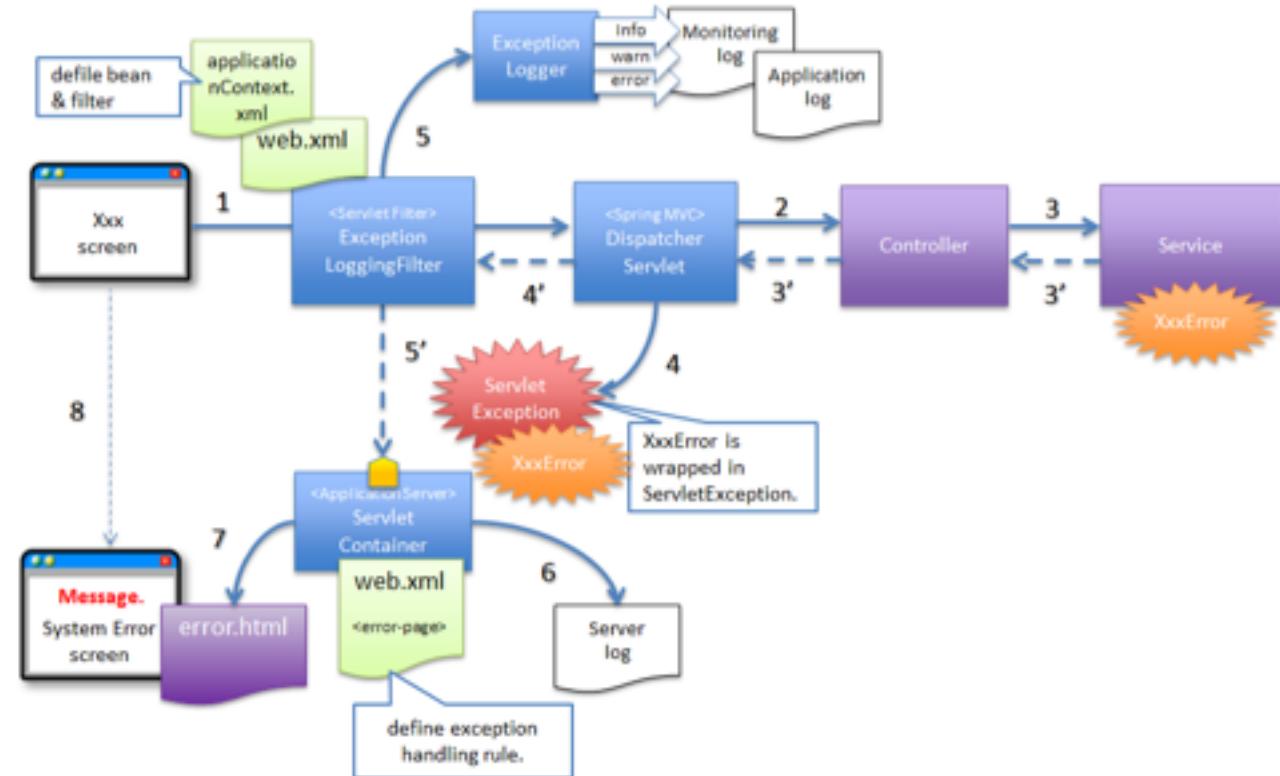
Sử dụng HandlerExceptionResolver

- Ngoại lệ được xử lý tại servlet
- Dùng để xử lý ngoại lệ hệ thống và từ chối request
- Được cấu hình bởi kiến trúc sư hệ thống



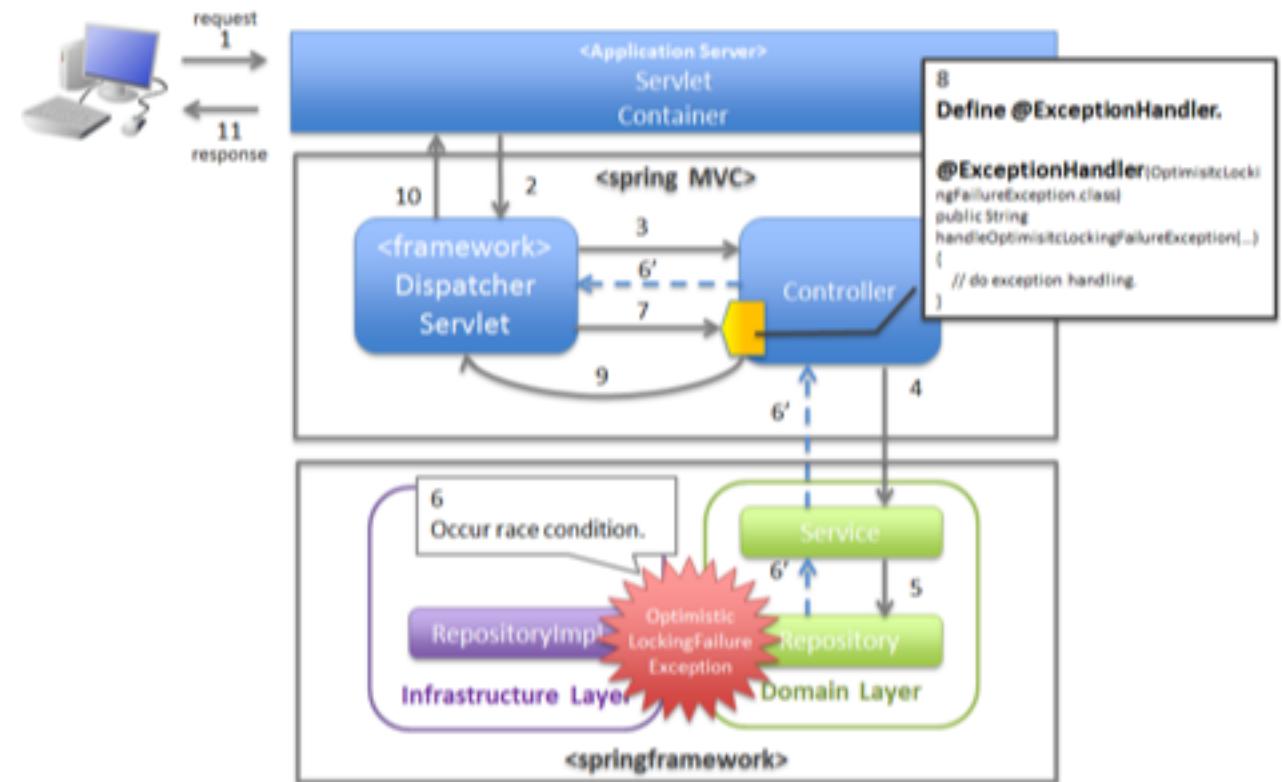
Sử dụng error-page

- Được xử lý bởi application server
- Dùng để xử lý lỗi fatal và các lỗi vượt quá khả năng điều khiển của framework
- Cấu hình bởi kiến trúc sư hệ thống



Spring @ExceptionHandler

- Là một annotation cho method, được sử dụng để xử lý exception tại controller hoặc method của nó
- Giúp mã dễ đọc và tăng khả năng tái sử dụng mã



Spring @ExceptionHandler (2)

```
@Controller
public class MyController {
    @RequestMapping(value = "/getData")
    public ModelAndView getData() {
        // ...
    }

    @ExceptionHandler(MyDataException.class)
    public String handleError(MyDataException e) {
        return "redirect:/showError.html";
    }
}
```

CODEGYM

CODEGYM

R a i s i n g t h e b a r