主題 2：spring-boot-starter-validation

使用 JSR303 實現參數驗證 (Bean Validation)

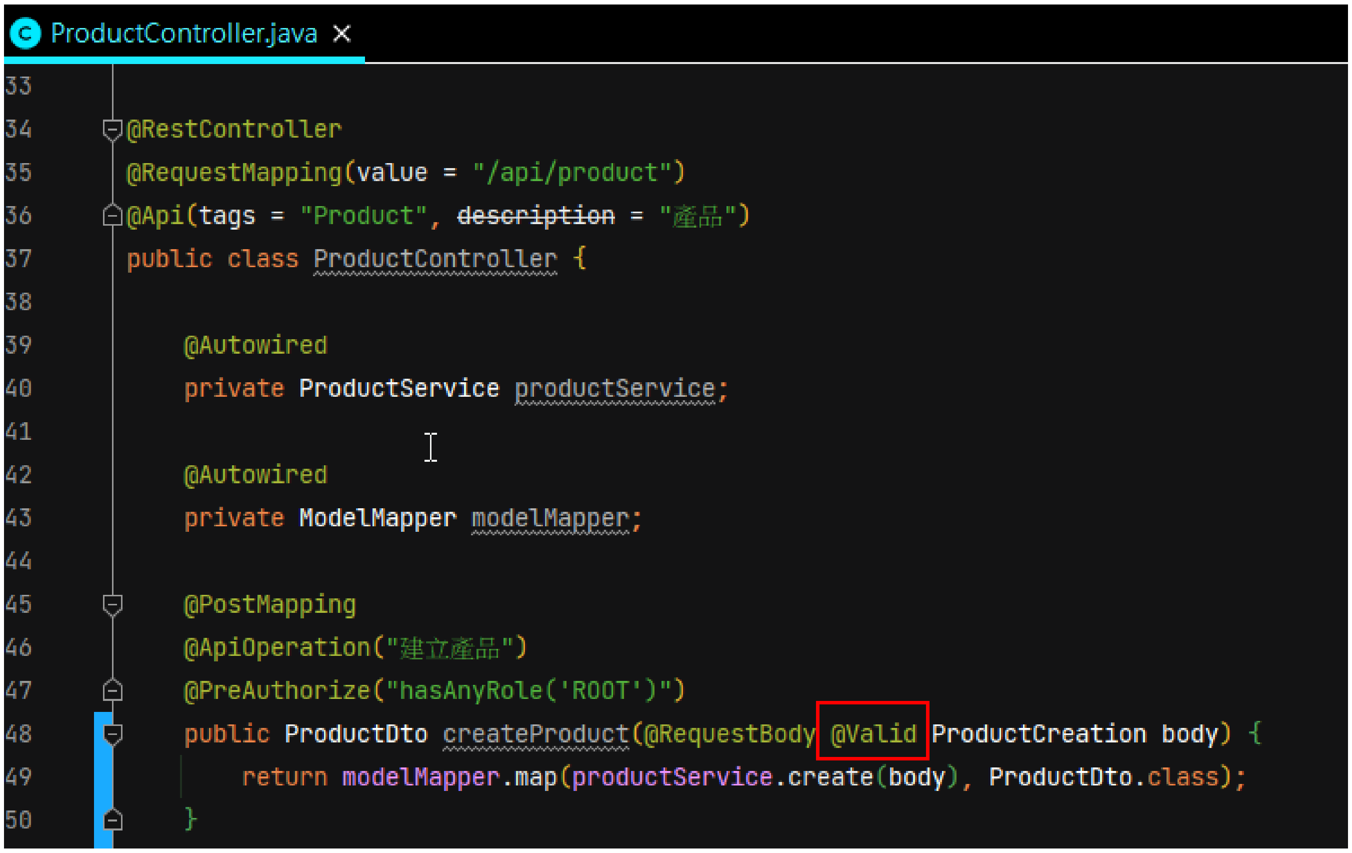
配置 maven 依賴

> *檔案位置：D:\workspace-IntelliJ\mytest-backend\mytest-api\pom.xml*
>
> *請留意....Bean Validation 若是 spring-boot-starter-web-2.3 以前的版本會直接內建*

pom.xml (mytest-backend)

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
    ....

    <dependencies>
        ....
                <dependency>
                           <groupId>org.springframework.boot</groupId>
                           <artifactId>spring-boot-starter-validation</artifactId>
                </dependency>
    </dependencies>

    ....

</project>
```

基本應用


ProductController.java

```java

@RestController
@RequestMapping(value = "/api/product")
@Api(tags = "Product", ~~description~~ = "產品")
public class ProductController {

    @Autowired
    private ProductService productService;

    @Autowired
    private ModelMapper modelMapper;

    @PostMapping
    @ApiOperation("建立產品")
    @PreAuthorize("hasAnyRole('ROOT')")
    public ProductDto createProduct(@RequestBody @Valid ProductCreation body) {
        return modelMapper.map(productService.create(body), ProductDto.class);
    }
```

ProductCreation.java

```java
package com.fubonlife.mytest.common.model.product;

import com.fubonlife.mytest.common.beanvalidator.ConstraintValidatorGroupService;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;

import javax.validation.constraints.Min;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Null;
import java.math.BigDecimal;

@Data
public class ProductCreation {

    @NotBlank
    @ApiModelProperty("")
    private String name;

    @Min(0)
    @ApiModelProperty("")
    private BigDecimal listPrice;

    @Min(0)
    @ApiModelProperty("")
    private BigDecimal unitCost;

    @NotBlank
    @ApiModelProperty("")
    private String attribute;

    @NotNull
    @ApiModelProperty("")
    private ProductStatus status;

    @ApiModelProperty(" ID")
    private String ownerAccountId;
}
```

異常的統一處理

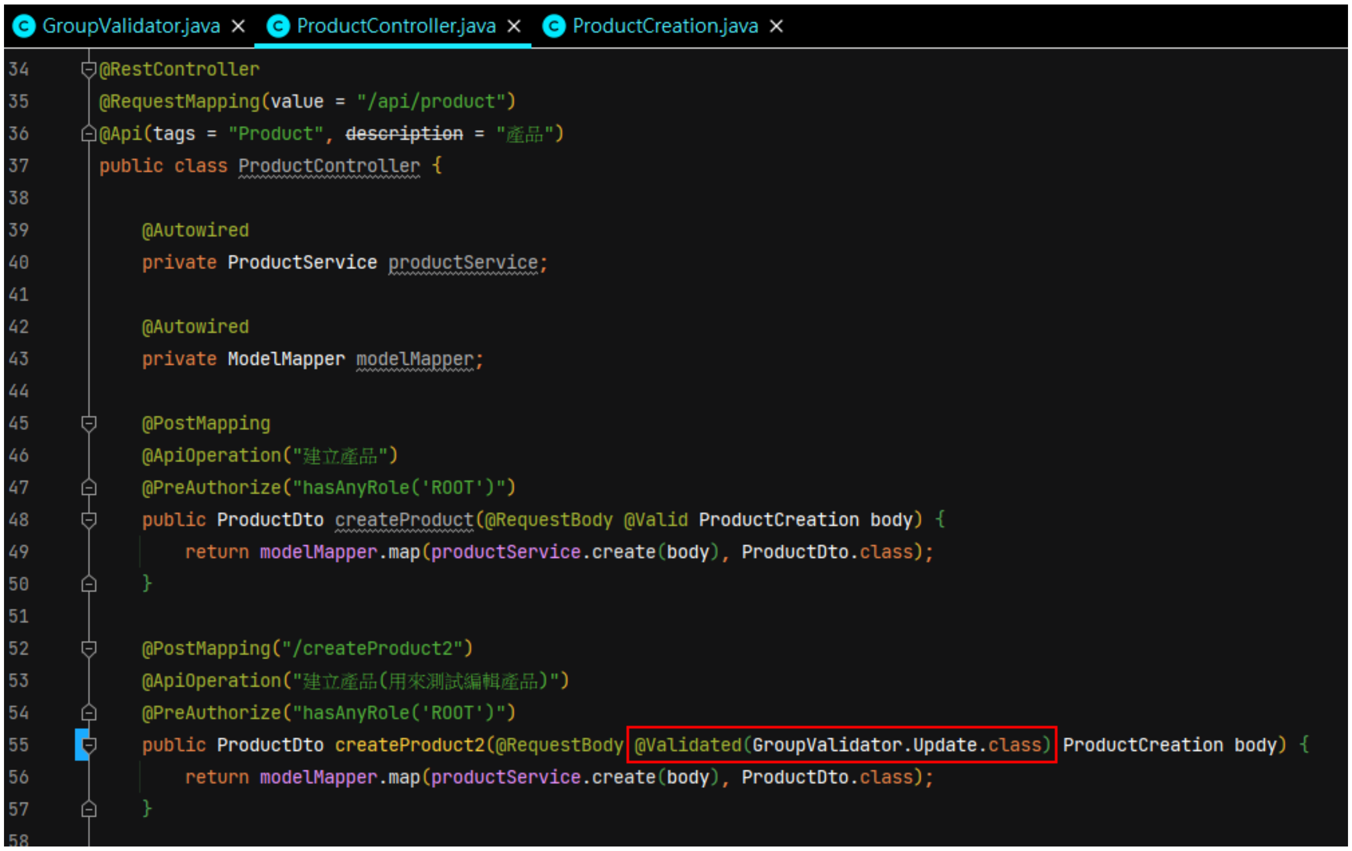
BeanValidationExceptionHandler.java

```
package com.fubonlife.mytest.api.security;

import com.fubonlife.boot.error.ApiError;
import lombok.extern.slf4j.Slf4j;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.BindException;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;

import java.util.ArrayList;
import java.util.List;

@ControllerAdvice
@Slf4j
public class BeanValidationExceptionHandler {

    @ExceptionHandler(value = {MethodArgumentNotValidException.class, BindException.class})
    public ResponseEntity<ApiError> handleBindException(Exception e) {

        List<String> errors = new ArrayList<>();
        BindingResult bindingResult;

        if (e instanceof MethodArgumentNotValidException) {
            log.warn("MethodArgumentNotValidException: {}", e.getMessage());
            bindingResult = ((MethodArgumentNotValidException) e).getBindingResult();
        } else {
            log.warn("BindException: {}", e.getMessage());
            bindingResult = ((BindException) e).getBindingResult();
        }

        bindingResult.getFieldErrors().forEach((fieldError) -> {
            errors.add(String.format("%s: %s", fieldError.getField(), fieldError.getDefaultMessage()));
        });

        ApiError apiError = new ApiError(HttpStatus.BAD_REQUEST, "", errors, "ARGUMENT_ERROR");
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(apiError);
    }
}
```

讓新增和修改的校驗規則可同時存在

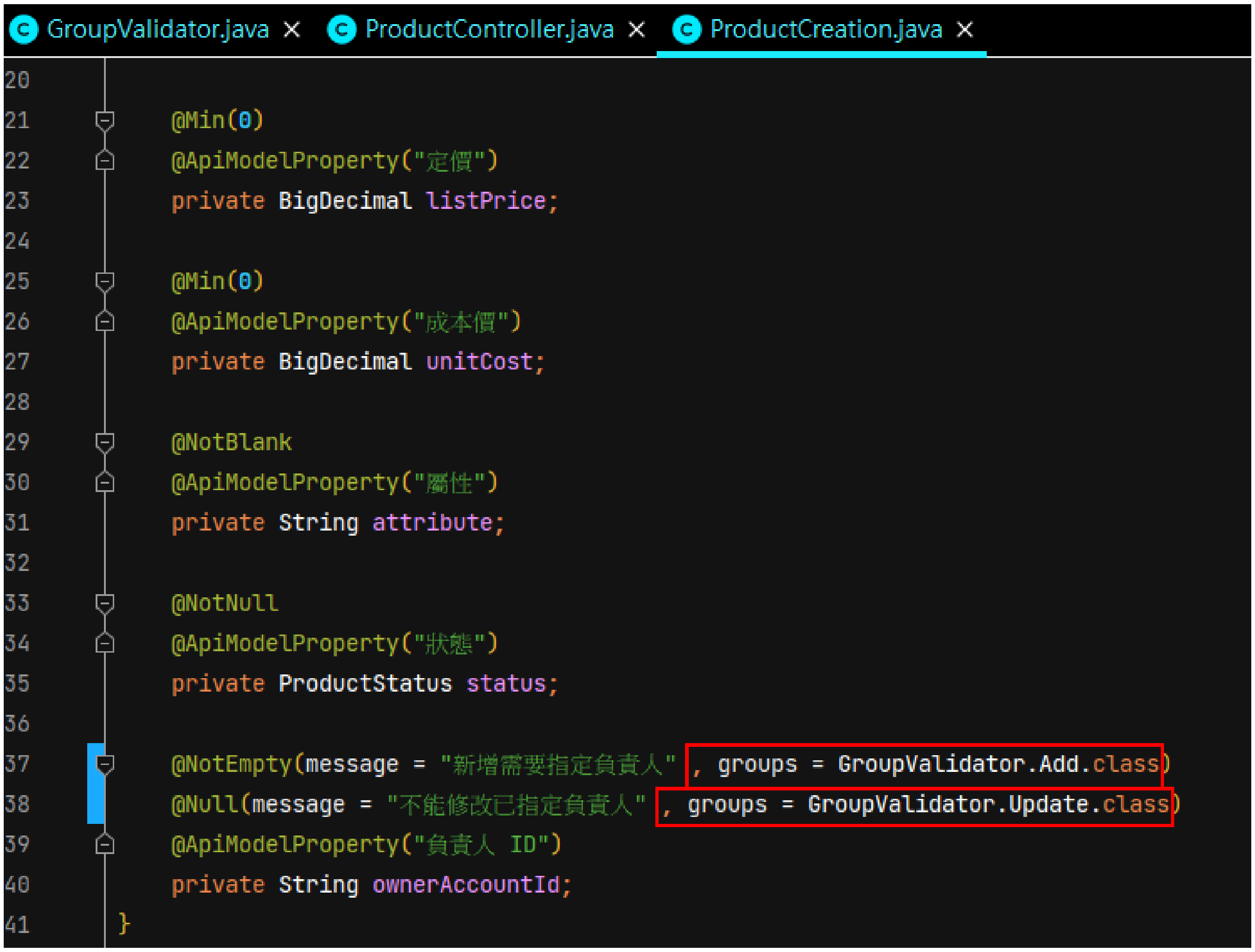
GroupValidator.java

```
package com.fubonlife.mytest.common.model.beanvalidation;

public class GroupValidator {

    public interface Add {
    }

    public interface Update {
    }


        //...
}
```



GroupValidator.java ProductController.java ProductCreation.java

```
@RestController
@RequestMapping(value = "/api/product")
@Api(tags = "Product", ~~description~~ = "產品")
public class ProductController {

    @Autowired
    private ProductService productService;

    @Autowired
    private ModelMapper modelMapper;

    @PostMapping
    @ApiOperation("建立產品")
    @PreAuthorize("hasAnyRole('ROOT')")
    public ProductDto createProduct(@RequestBody @Valid ProductCreation body) {
        return modelMapper.map(productService.create(body), ProductDto.class);
    }

    @PostMapping("/createProduct2")
    @ApiOperation("建立產品(用來測試編輯產品)")
    @PreAuthorize("hasAnyRole('ROOT')")
    public ProductDto createProduct2(@RequestBody @Validated(GroupValidator.Update.class) ProductCreation body) {
        return modelMapper.map(productService.create(body), ProductDto.class);
    }

```



```java

    @Min(0)
    @ApiModelProperty("定價")
    private BigDecimal listPrice;

    @Min(0)
    @ApiModelProperty("成本價")
    private BigDecimal unitCost;

    @NotBlank
    @ApiModelProperty("屬性")
    private String attribute;

    @NotNull
    @ApiModelProperty("狀態")
    private ProductStatus status;

    @NotEmpty(message = "新增需要指定負責人" , groups = GroupValidator.Add.class)
    @Null(message = "不能修改已指定負責人" , groups = GroupValidator.Update.class)
    @ApiModelProperty("負責人 ID")
    private String ownerAccountId;
}
```