Practice 6: MVC, Services, Controllers and Views, part II

1. Open the project lab6 in IntelliJ IDE: File – New Project from Existing Sources. Add H2 and MySql run configurations **-Dspring.profiles.active=** MySql. Remember if H2 profile is running, http://localhost:8080/h2-console is available.

2. Add classes com.awbd.lab6.services.CategoryServiceImpl and com.awbd.lab6.services.CategoryService.

   Add in ProductsController a field of type com.awbd.lab6.services.CategoryService. Annotate categoryService @Autowired.

```
public interface CategoryService {
    List<Category> findAll();
}
```

```
@Service
public class CategoryServiceImpl implements CategoryService{

    CatagoryRepository categoryRepository;

    @Autowired
    public CategoryServiceImpl(CatagoryRepository categoryRepository) {
        this.categoryRepository = categoryRepository;
    }

    @Override
    public List<Category> findAll() {
        List<Category> categories = new LinkedList<>();
     categoryRepository.findAll().iterator().forEachRemaining(categories
::add);
        return categories ;
    }
}
```

3. Add in productform.html:

```
<label th:for="categories">Categories</label>
<ul id ="categories"  style="list-style: none;">
    <li th:each="category: ${categoriesAll}" >
        <input type="checkbox"
               th:field="*{categories}"
               th:text="${category.name}"
               th:value="${category.id}" />
        <label
                th:for="${#ids.prev('categories')}"
                th:text="${category.name}">
        </label>
    </li>
</ul>
```

4. Add in info.html:

```
<ul>
    <li
th:each="category:${product.categories}"th:text="${category.getName()}">
        category
    </li>
</ul>
```

5. When "/product/new" request is processed add in model an object containing the list of categories:

```
@RequestMapping("/product/new")
public String newFilm(Model model) {
    model.addAttribute("product", new Product());
    model.addAttribute("categories", categoryService.findAll());
    return "productform";
}
```

6. Add Boolean attribute *restored* in class Products and filed restored. In tymeleaf templates info.html and new.html add:

```
<div class="col-md-3">
    <label th:for="restored">Restored</label>
    <input type="checkbox" th:field="${product.restored}" />
</div>
```

```
<div class="col-md-3">
    <label th:for="restored">Restored</label>
    <input type="checkbox" th:field="*{restored}" />
</div>
```

```
<label th:for="reservePrice" >price:</label>
<input type="number" class="form-control" th:field="*{reservePrice}"/>
```

```
<div th:text="${product.reservePrice != null ? product.reservePrice:'
'}">price</div>
```

7. Add in info.html tymeleaf template:

```
<div th:if="${product.currency ==
T(com.awbd.lab6.domain.Currency).EUR}">
    &euro;
</div>

<div th:if="${product.currency ==
T(com.awbd.lab6.domain.Currency).USD}">
    &dollar;
</div>

<div th:if="${product.currency ==
T(com.awbd.lab6.domain.Currency.GBP}">
    &pound;
</div>
```

8. Modify Currency Enumeration. Add an attribute description, a constructor, and a getter method:

```
public enum Currency {
    USD("USD $"), EUR("EUR"), GBP("GBP");

    private String description;

    public String getDescription() {
        return description;
    }

    Currency(String description) {
        this.description = description;
    }
}
```

9. Add in productform.html template:

```
<label for="currency">Currency: </label>
<select id="currency" name="currency" th:field="*{currency}">
    <option

th:each="currencyOpt:${T(com.awbd.lab6.domain.Currency).values()}"
            th:value="${currencyOpt}"
th:text="${currencyOpt.getDescription()}">

    </option>
</select>
```

Info **Enumerations** [1]

T from Spring Expression Language specifies an instance of a class or static methods.
Enums are special types of classes extending *java.lang.Enum*. We can define custom methods and constructors. We may use Enums in if or switch statements.

Info **@Transactional** [2] annotation defines the scope of a single database transaction. It can be used at method or class level. TransactionManager is responsible for starting new transactions.

HttpServletResponse [3] can be used as argument to endpoints (request methods) to add headers to the Http response or to specify the content type of the response.

10. Add classes classes com.awbd.lab6.services.ImageServiceImpl and com.awbd.lab7.services.ImageService.

```
public interface ImageService {
    void saveImageFile(Long productId, MultipartFile file);
}
```

```
@Service
public class ImageServiceImpl implements ImageService{
    ProductRepository productRepository;

    @Autowired
    public ImageServiceImpl(ProductRepository productRepository) {
        this.productRepository = productRepository;
    }

    @Override
    @Transactional
    public void saveImageFile(Long productId, MultipartFile file) {
        try {
            Product product =
productRepository.findById(productId).get();

            Byte[] byteObjects = new Byte[file.getBytes().length];
            int i = 0; for (byte b : file.getBytes()){
                byteObjects[i++] = b; }

            Info info = product.getInfo();
            if (info == null) {
                info = new Info();
            }

            info.setImage(byteObjects);
            product.setInfo(info);
            info.setProduct(product);
            productRepository.save(product); }
        catch (IOException e) {
        }
    }
}
```

11. In productform.html template add a file for product image:

```
<label class="control-label">Select Image File</label>
<input id="imagefile" name="imagefile" type="file" class="file">
```

12. Modify form in productform.html to accept multipart/form-data encoding:

```
<form th:object="${product}" th:action="@{/product/}" method="post"
enctype="multipart/form-data">
```

13. Add field ImageService imageService in ProductController class:

```
@Autowired
ImageService imageService;
```

14. Modify ProductController, call image.Service in saveOrUpdate method:

```
@PostMapping("/product")
public String saveOrUpdate(@ModelAttribute Product product,
                           @RequestParam("imagefile") MultipartFile
file){
    Product savedProduct = productService.save(product);
    imageService.saveImageFile(Long.valueOf(savedProduct.getId()),
file);
    return "redirect:/product/list" ;
}
```

15. Add class com.awbd.lab6.controllers.ImageController

```
@Controller
public class ImageController {

    private final ProductService productService;


    public ImageController(@Autowired ImageService imageService,
@Autowired ProductService productService) {
        this.productService = productService;
    }

    @GetMapping("product/getimage/{id}")
    public void downloadImage(@PathVariable String id,
HttpServletResponse response) throws IOException {
        Product product = productService.findById(Long.valueOf(id));
        if (product.getInfo() != null) {
            Info info = product.getInfo();

            if (product.getInfo().getImage() != null) {
                byte[] byteArray = new byte[info.getImage().length];
                int i = 0;
                for (Byte wrappedByte : info.getImage()) {
                    byteArray[i++] = wrappedByte;
                }
                response.setContentType("image/jpeg");
                InputStream is = new ByteArrayInputStream(byteArray);
                try {
                    IOUtils.copy(is, response.getOutputStream());
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

16. Add in info.html template:

```
<div class = "row">
    <div class="col-md-6"> <img th:src="@{'/product/getimage/' +
${product.id}}" width="200" height="200"> </div>
</div>
```

B

[1] https://www.baeldung.com/thymeleaf-enums

[2] https://www.baeldung.com/transaction-configuration-with-jpa-and-spring

[3] https://www.baeldung.com/spring-response-header

[4] https://www.baeldung.com/spring-boot-bean-validation

[5] http://hibernate.org/validator/