

## MỤC TIÊU:

Kết thúc bài thực hành này bạn có khả năng

- ✓ Giải thích được Spring Bean
- ✓ Sử dụng được `@Autowired` và `@Qualifier` để tiêm các Spring Bean vào các thành phần trong ứng dụng
- ✓ Mô tả được cơ chế IoC
- ✓ Xây dựng được các lớp và nạp vào hệ thống thông qua cấu hình (`@Bean`, `@Primary`) và annotation (`@Component`, `@Service`, `@Repository`)
- ✓ Sử dụng được các annotation `@Scope`, `SessionScope`, `@RequestScope`, `@ApplicationScope` để quản lý vòng đời và chia sẻ Spring bean hiệu quả hơn

## PHẦN I

### Bài 1 (2 điểm)

Hãy cài đặt mã nguồn cho các phương thức làm việc với parameter được đặc tả trong lớp Spring Bean (ParamService) như sau:

```
@Service
public class ParamService {
    @Autowired
    HttpServletRequest request;

    /**
     * Đọc chuỗi giá trị của tham số
     * @param name tên tham số
     * @param defaultValue giá trị mặc định
     * @return giá trị tham số hoặc giá trị mặc định nếu không tồn tại
     */
    public String getString(String name, String defaultValue){...}
    /**
     * Đọc số nguyên giá trị của tham số
     * @param name tên tham số
     * @param defaultValue giá trị mặc định
     * @return giá trị tham số hoặc giá trị mặc định nếu không tồn tại
     */
}
```

```
*/
public int getInt(String name, int defaultValue){...}
/**
 * Đọc số thực giá trị của tham số
 * @param name tên tham số
 * @param defaultValue giá trị mặc định
 * @return giá trị tham số hoặc giá trị mặc định nếu không tồn tại
 */
public double getDouble(String name, double defaultValue){...}
/**
 * Đọc giá trị boolean của tham số
 * @param name tên tham số
 * @param defaultValue giá trị mặc định
 * @return giá trị tham số hoặc giá trị mặc định nếu không tồn tại
 */
public boolean getBoolean(String name, boolean defaultValue){...}
/**
 * Đọc giá trị thời gian của tham số
 * @param name tên tham số
 * @param pattern là định dạng thời gian
 * @return giá trị tham số hoặc null nếu không tồn tại
 * @throws RuntimeException lỗi sai định dạng
 */
public Date getDate(String name, String pattern){...}
/**
 * Lưu file upload vào thư mục
 * @param file chứa file upload từ client
 * @param path đường dẫn tính từ webroot
 * @return đối tượng chứa file đã lưu hoặc null nếu không có file upload
 * @throws RuntimeException lỗi lưu file
 */
public File save(MultipartFile file, String path) {...}
}
```

CookieService: Hãy cài đặt mã nguồn cho các phương thức làm việc với cookie được đặc tả trong lớp Spring Bean (CookieService) như sau:

```
@Service
public class CookieService {
    @Autowired
    HttpServletRequest request;
    @Autowired
    HttpServletResponse response;

    /**
     * Đọc cookie từ request
     * @param name tên cookie cần đọc
     * @return đối tượng cookie đọc được hoặc null nếu không tồn tại
     */
    public Cookie get(String name) {...}

    /**
     * Đọc giá trị của cookie từ request
     * @param name tên cookie cần đọc
     * @return chuỗi giá trị đọc được hoặc rỗng nếu không tồn tại
     */
    public String getValue(String name) {...}

    /**
     * Tạo và gửi cookie về client
     * @param name tên cookie
     * @param value giá trị cookie
     * @param hours thời hạn (giờ)
     * @return đối tượng cookie đã tạo
     */
    public Cookie add(String name, String value, int hours) {...}

    /**
     * Xóa cookie khỏi client
     * @param name tên cookie cần xóa
     */
    public void remove(String name) {...}
}
```

SessionService: Hãy cài đặt mã nguồn cho các phương thức làm việc với attribute trong session được đặc tả trong lớp Spring Bean (SessionService) như sau:

```
@Service
```

```
public class SessionService {
    @Autowired
    HttpSession session;
    /**
     * Đọc giá trị của attribute trong session
     * @param name tên attribute
     * @return giá trị đọc được hoặc null nếu không tồn tại
     */
    public <T> T get(String name) {...}
    /**
     * Thay đổi hoặc tạo mới attribute trong session
     * @param name tên attribute
     * @param value giá trị attribute
     */
    public void set(String name, Object value) {...}
    /**
     * Xóa attribute trong session
     * @param name tên attribute cần xóa
     */
    public void remove(String name) {...}
}
```

## Bài 2 (2 điểm)

Tạo AccountController để xử lý đăng nhập có cấu trúc như sau

```
@Controller
public class AccountController {
    @GetMapping("/account/login")
    public String login1() {
        return "/account/login";
    }
    @PostMapping("/account/login")
    public String login2() {
        return "/account/login";
    }
}
```

Tạo view login.html có chứa form đăng nhập như sau

```
<form action="/account/login" method="post">
  <input name="username">
  <input name="password">
  <input type="checkbox" name="remember" value="true">
  <button>Login</button>
</form>
```

Tiêm ParamService, CookieService vào AccountController để làm việc với tham số và cookie

```
@Autowired
CookieService cookieService;
@Autowired
ParamService paramService;
@Autowired
SessionService sessionService;
```

Viết mã để đọc các tham số (username, password, remember) và ghi nhớ tài khoản đăng nhập được cho login2()

```
String un = paramService.getString("username", "");
String pw = paramService.getString("password", "");
boolean rm = paramService.getBoolean("remember", false);
```

Nếu đăng nhập thành công (un="poly", pw="123") thì

- Lưu username vào session  
`sessionService.set("username", username);`
- Xử lý ghi nhớ tài khoản
  - Nếu remember là true thì ghi nhớ tài khoản 10 ngày  
`cookieService.add("user", username, 10);`
  - Ngược lại thì xóa cookie tài khoản đã ghi nhớ trước đó  
`cookieService.remove("user");`

### Bài 3 (1 điểm)

Giảng viên cho thêm đăng ký có upload hình và sử dụng ParamService.save() để lưu hình.

## PHẦN II

### Bài 4 (2 điểm)

Cho interface ShoppingCartService mô tả các hoạt động của giỏ hàng như sau:

```
public interface ShoppingCartService {  
    /**  
     * Thêm mặt hàng vào giỏ hoặc tăng số lượng lên 1 nếu đã tồn tại  
     * @param id là mã mặt hàng cần thêm  
     * @return mặt hàng đã được thêm vào hoặc cập nhật số lượng  
     */  
    Item add(Integer id);  
    /**  
     * Xóa mặt hàng khỏi giỏ  
     * @param id mã mặt hàng cần xóa  
     */  
    void remove(Integer id);  
    /**  
     * Thay đổi số lượng lên của mặt hàng trong giỏ  
     * @param id mã mặt hàng  
     * @param qty số lượng mới  
     * @return mặt hàng đã được thay đổi số lượng  
     */  
    Item update(Integer id, int qty);  
    /**  
     * Xóa sạch các mặt hàng trong giỏ  
     */  
    void clear();  
    /**  
     * Lấy tất cả các mặt hàng trong giỏ  
     */  
    Collection<Item> getItems();  
    /**  
     * Lấy tổng số lượng các mặt hàng trong giỏ  
     */  
    int getCount();  
    /**  
     * Lấy tổng số tiền tất cả các mặt hàng trong giỏ  
     */  
}
```

```
*/  
double getAmount();  
}
```

Trong đó lớp Item mô tả thông tin của mặt hàng như sau

```
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
public class Item {  
    Integer id;  
    String name;  
    double price;  
    int qty = 1;  
}
```

Hãy tạo lớp Spring Bean ShoppingCartServiceImpl và cài đặt mã nguồn cho các phương thức thực hiện như đã đặc tả trong interface ShoppingCartService.

Hướng dẫn:

Sử dụng @Service và @SessionScope để khai báo cho ShoppingCartServiceImpl để Spring Bean được tạo theo từng phiên làm việc.

Có thể sử dụng List<Item> hoặc Map<Integer, Item> để chứa các mặt hàng đã chọn. Nếu dùng map thì việc truy xuất và kiểm tra sự tồn tại của một mặt hàng trong giỏ sẽ thuận tiện hơn.

```
@SessionScope  
@Service  
public class ShoppingCartServiceImpl implements ShoppingCartService{  
    Map<Integer, Item> map = new HashMap<>();  
    @Override  
    public Item add(Integer id) {  
        // TODO Auto-generated method stub  
        return null;  
    }  
    @Override  
    public void remove(Integer id) {  
        // TODO Auto-generated method stub  
    }  
}
```

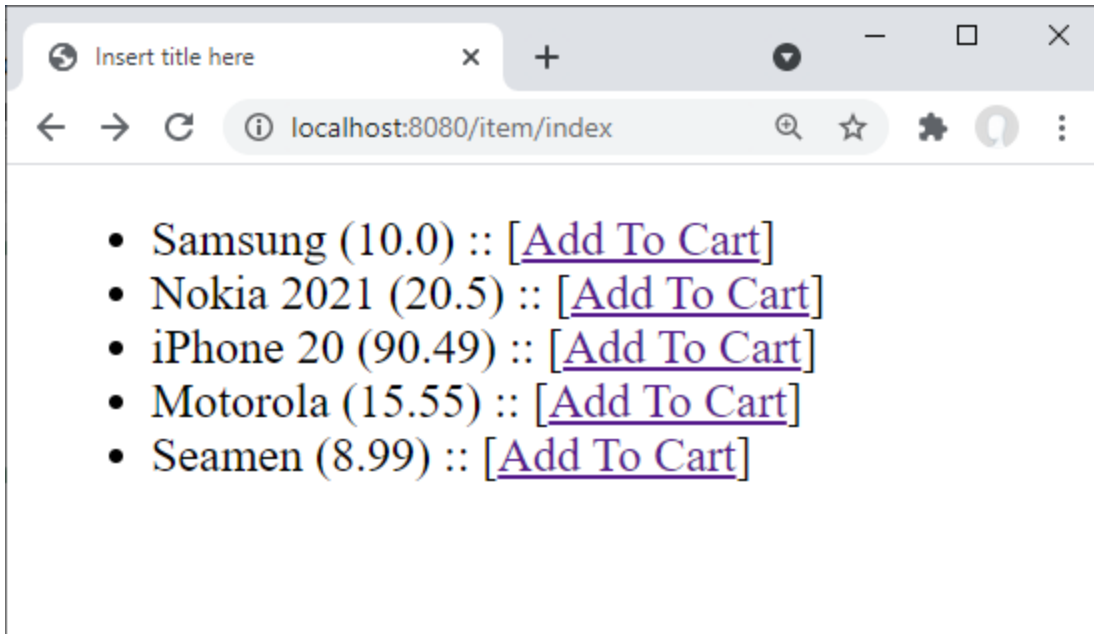
```
}  
@Override  
public Item update(Integer id, int qty) {  
    // TODO Auto-generated method stub  
    return null;  
}  
@Override  
public void clear() {  
    // TODO Auto-generated method stub  
  
}  
@Override  
public Collection<Item> getItems() {  
    // TODO Auto-generated method stub  
    return null;  
}  
@Override  
public int getCount() {  
    // TODO Auto-generated method stub  
    return 0;  
}  
@Override  
public double getAmount() {  
    // TODO Auto-generated method stub  
    return 0;  
}  
}
```

### Bài 5 (2 điểm)

Xây các trang web quản lý giỏ hàng theo hướng dẫn sau

1. Hiển thị các mặt hàng lên trang web





Để tạo trang trên cần các thành phần sau:

- Nguồn dữ liệu hàng hóa

```
public class DB {
    public static Map<Integer, Item> items = new HashMap<>();
    static {
        items.put(1, new Item(1, "Samsung", 10.0, 0));
        items.put(2, new Item(2, "Nokia 2021", 20.50, 0));
        items.put(3, new Item(3, "iPhone 20", 90.49, 0));
        items.put(4, new Item(4, "Motorola", 15.55, 0));
        items.put(5, new Item(5, "Seamen", 8.99, 0));
    }
}
```

- ItemController chuyển dữ liệu sang giao diện để hiển thị

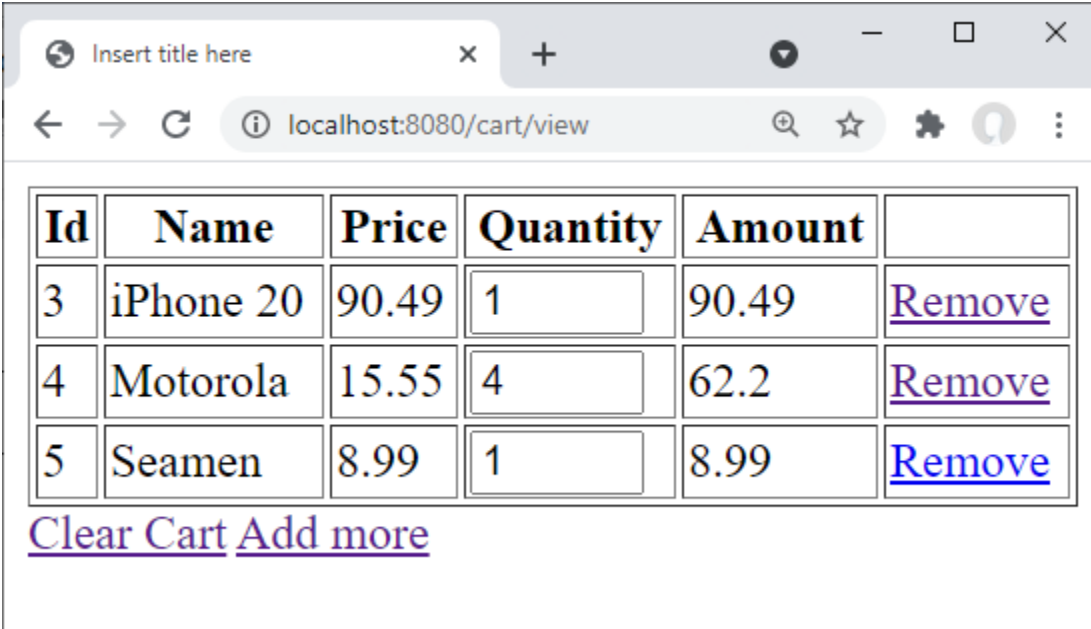
```
@Controller
public class ItemController {
    @RequestMapping("/item/index")
    public String list(Model model) {
        model.addAttribute("items", DB.items.values());
        return "item/index";
    }
}
```

```
}  
}
```

- View item/index.html hiển thị các mặt hàng lên trang web

```
<ul>  
<th:block th:each="item: ${items}">  
    <li>  
        ${item.name} (${item.price}) ::  
        [<a th:href="@{/cart/add/${item.id}/*}">Add To Cart</a>]  
    </li>  
</th:block>  
</ul>
```

## 2. Hiển thị giỏ hàng



Id	Name	Price	Quantity	Amount	
3	iPhone 20	90.49	<input type="text" value="1"/>	90.49	<a href="#">Remove</a>
4	Motorola	15.55	<input type="text" value="4"/>	62.2	<a href="#">Remove</a>
5	Seamen	8.99	<input type="text" value="1"/>	8.99	<a href="#">Remove</a>

[Clear Cart](#) [Add more](#)

Để hiển thị được giỏ hàng chúng ta cần các thành phần sau

- ShoppingCartController chuyển dữ liệu giỏ hàng sang giao diện để hiển thị

```
@Controller  
public class ShoppingCartController {  
    @Autowired  
    ShoppingCartService cart; // thêm Spring Bean đã viết ở bài trước
```

```
@RequestMapping("/cart/view")
public String view(Model model) {
    model.addAttribute("cart", cart);
    return "cart/index";
}
}
```

- View cart/index.html hiển thị các mặt hàng trong giỏ

```
<table border="1" style="width:100%">
<tr>
    <th>Id</th>
    <th>Name</th>
    <th>Price</th>
    <th>Quantity</th>
    <th>Amount</th>
    <th></th>
</tr>
<tr th:each="item: ${cart.items}">
    <form th:action="@{||/cart/update/${item.id}||}" method="post">
        <input type="hidden" name="id" th:value="${item.id}">
        <tr>
            <td th:text="${item.id}"></td>
            <td th:text="${item.name}"></td>
            <td th:text="${item.price}"></td>
            <td><input name="qty" th:value="${item.qty}"
                onblur="this.form.submit()"></td>
            <td th:text="${item.price * item.qty}"></td>
            <td>
                <a
th:href="@{||/cart/remove/${item.id}||}">Remove</a>
            </td>
        </tr>
    </form>
</tr>
</table>
<a href="/cart/clear">Clear Cart</a>
<a href="/item/index">Add more</a>
```

3. Các tương tác liên quan đến giỏ hàng thể hiện qua các trang html gồm

```
<a th:href="@{/cart/add/${item.id}]">Add To Cart</a>
<form th:action="@{/cart/update/${item.id}]" method="post">
<a th:href="@{/cart/remove/${item.id}]">Remove</a>
<a href="/cart/clear">Clear Cart</a>
```

4. Hoàn thiện ShoppingCartController bằng cách bổ sung các phương thức và ánh xạ với các địa chỉ url trên giao diện html để điều khiển các request tương ứng

```
@RequestMapping("/cart/add/{id}")
public String add(@PathVariable("id") Integer id) {
    cart.add(id);
    return "redirect:/cart/view"; // hiển thị giỏ hàng
}
@RequestMapping("/cart/remove/{id}")
public String remove(@PathVariable("id") Integer id) {
    cart.remove(id);
    return "redirect:/cart/view";
}
@RequestMapping("/cart/update/{id}")
public String update(@PathVariable("id") Integer id,
                    @RequestParam("qty") Integer qty) {
    cart.update(id, qty);
    return "redirect:/cart/view";
}
@RequestMapping("/cart/clear")
public String clear() {
    cart.clear();
    return "redirect:/cart/view";
}
```

## Bài 6 (1 điểm)

Giảng viên cho thêm