

# Ćwiczenia

## Lab 1\_00 – Konfiguracja aplikacji Spring MVC

Cel ćwiczenia:

- Budowa aplikacji web w oparciu o Springa 4.
- Uruchomienie projektu na Tomcacie

Kroki:

1. W projekcie znajduje się klasa WebBootstrap przeanalizuj jej zawartość
2. Dodaj projekt do Tomcata 8
3. Wyświetl komunikat podczas dodawania produktów do repo
4. Wyświetl komunikat po uruchomieniu aplikacji na serwerze

## Lab 1\_01 – Pierwszy kontroler

Cel ćwiczenia:

- Stworzenie kontrolera HelloWorld

Kroki:

1. W pakiecie pl.altkom.shop.controller dodaj kontroler AltkomKontroler
2. Zmapuj kontroler na /altkom
3. Zadeklaruj HttpServletResponse response i skorzystaj z niego aby wyświetlić dane dla użytkownika np.  
`res.getWriter().append("Witaj na szkoleniu Altkom");`
4. Uruchom kontroler w przeglądarce /shop/altkom

## **Lab 1\_02 – Kontrolery pobieranie wartości GET**

Cel ćwiczenia:

- Odebranie parametrów w kontrolerze

Kroki:

1. Dodaj kontroler ProductController który odczyta wartości parametrów dla /shop/product/list?page=1&size=20&orderBy=name
2. Dodaj mapowanie które odczyta wartość dla mapowania /show/product/12
3. Otrzymane parametry zaprezentuj za pomocą HttpServletResponse response

## **Lab 1\_03 – Kontrolery przekazywanie danych do widoków**

Cel ćwiczenia:

- Prezentacja danych na stronie JSP

Kroki:

1. Informacje z poprzedniego ćwiczenia (/shop/product/list?page=1&size=20&orderBy=name) zaprezentuj na stronie JSP korzystając z wyrażień \${nazwaParametru}. Dodaj odpiedni plik JSP /WEB-INF/pages/product/product-list.jsp(skopiuj index.jsp) oraz zwróć z kontrolera odpowiedni identyfikator widoku który pokaże product/product-list.jsp
2. Pobierz listę wszystkich produktów i wyświetl je w formie tabelki




## Lab 1\_04 – Kontrolery reakcja na akcje użytkownika

Cel ćwiczenia:

- Zareagowanie na interakcje użytkownika z aplikacją

Kroki:

1. Dodać kontroler który będzie usuwał produkt po identyfikatorze /product/12/delete – sprawdzić działanie w przeglądarce wpisując adres bezpośrednio
2. Stworzoną funkcjonalność dodać w aplikacji (link)
3. Po usunięciu produktu pokazać odświeżoną listę

Number	Name	Quantity	Price	Actions
1	Rower	12	10	
2	Sanki	123	12.45	
3				

## **Lab 1\_05 – Kontrolery obsługa dodawania**

Cel ćwiczenia:

- Stworzenie formularza dodającego produkt

Kroki:

1. Do product-list.jsp dodaj akcję/linka która pokaże formularz product/new z product-form.jsp
2. Dodaj kontroler obsługujący formularz POST product/new
3. Po zapisie produktu pokaż odświeżoną listę produktów

## Lab 1\_06 – Kontrolery edycja

Cel ćwiczenia:

- Stworzenie formularza edytującego produkt

Kroki:

1. Analogicznie jak podczas usuwania produktów dodać akcję edytującą dany produkt
2. W kontrolerze pobrać produkt o id i wyświetlić dane na formularzu (product-form)
3. Podczas zapisu zaaktualizować wpis o produkcie i przejść do listy

## Lab 1\_07– Lokalizacja

Cel ćwiczenia:

- Wielojęzyczność w aplikacji

Kroki:

1. Skonfigurować ResourceBundleMessageSource
2. Założyć plik messages.properties
3. W aplikacji zamienić wszystkie stringi tak aby korzystały z `<spring:message code="name.label"/>`
4. Dodać plik messages\_en.properties i sprawdzić działanie w sytuacji kiedy zmienimy locale w przeglądarce

## **Lab 1\_08 – Walidacja formularzy**

Cel ćwiczenia:

- Walidacja formularzy

Kroki:

1. W kontrolerze wstrzyknąć jako parametr metody obiekt BindingResult
2. Dodać błąd dla pola reject
3. Wyświetlić błąd na formularzu



## **Lab 1\_09 – Walidacja formularzy deklaratywnie**

Cel ćwiczenia:

- Walidacja formularzy JSR 303

Kroki:

1. Dodać zależność maven hibernate-validator
2. Dodać adnotacje NotNull/NotEmpty na odpowiednie pola
3. Dodać adnotację @Valid do parametru wejściowego  
@ModelAttribute @Valid Product product
4. Wyświetlić błędy na formularzu

## Lab 1\_10 – Widok PDF

Cel ćwiczenia:

- Praca z innymi typami widoków

Kroki:

1. Skonfigurować BeanNameViewResolver
2. Dodać bean ProductListPDFView który dziedziczy po AbstractPdfView i korzystając z api Table stworzyć tabelkę

```
Table table = new Table( 1 );  
table.addCell(„Name”)  
table.addCell(„Wartość”)  
document.add(table)
```

## Lab 1\_11 – Kontrolery REST

Cel ćwiczenia:

- Tworzenie kontrolerów Rest'owych
- Testowanie za pomocą Soap-UI

Kroki:

1. Dodaj kontroler RestPhoneController z mapowaniem /rest/product
2. Korzystając z adnotacji @RestController oraz @RequestBody zaimplementuj wszystkie funkcjonalności kontrolera PhoneController
3. Zaobserwuj jak działa walidacja
4. Dodaj funkcjonalność wyszukiwania produktów po nazwie

## Lab 1\_12 – Swagger

Cel ćwiczenia:

- Dodać bibliotekę Swagger

Kroki:

1. <http://www.baeldung.com/swagger-2-documentation-for-spring-rest-api>
2. Zależności
3. Config
4. Mapowanie resource'ów

## Lab 1\_13 – JDBC Template

Cel ćwiczenia:

- Stworzenie JDBCProductRepository który będzie zapisywał produkty w bazie danych

Kroki:

1. Dodać zależność spring-jdbc
2. Założyć bazę danych products za pomocą PHPMyAdmin
3. Założyć tabelę PRODUCT z kolumnami takimi jak w klasie Product
4. Skonfigurować DataSource dla MySQL (patrz propsy poniżej)
5. Stworzyć JDBCTemplate przekazując mu DataSource
6. Zaimplementować CRUD

```
jdbc.driverClass=com.mysql.jdbc.Driver
```

```
jdbc.url=jdbc:mysql://localhost:3306/products
```

```
jdbc.userName=root
```

```
jdbc.password=
```