

# 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 dodawnia produktów do repo
- 4. Wyświetl komunikat po uruchomienia aplikacji na serwerze

# Lab 1\_01 – Pierwszy kontroler

## Cel ćwiczenia:

Stworzenie kontrolera HelloWold

- 1. W pakiecie pl.altkom.shop.controller dodaj kontroler AltkomKontroler
- 2. Zmapuj kontroler na /altkom
- 3. Zadeklaru 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
- Dodaj mapowanie które odczyta wartość dla mopowania /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

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

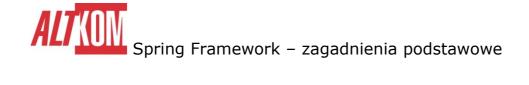
# Lab 1\_04 – Kontrolery reakcja na akcje użytkownika

# Cel ćwiczenia:

• Zareagowanie na interakcje użytkownika z aplikacją

- 1. Dodać kontroler który będzie usuwał produkt po identyfikatorze /product/12/delete sprawdzić działąnie w przegląradce 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	<b>⊗</b>
3				⊗



# Lab 1\_05 - Kontrolery obsługa dodawania

# Cel ćwiczenia:

• Stworzenie formularza dodającego produkt

- 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

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

# Lab 1\_07- Lokalizacja

# Cel ćwiczenia:

• Wielojęzyczność w aplikacji

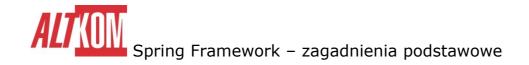
- 1. Skonfigurować ResourceBundleMessageSource
- 2. Założyć plik messages.properties
- 3. W aplikacji zamienić wszystki stringi tak aby korzystały z <spring:message code="name.label"/>
- 4. Dodać plik messages\_en.properties i sprawidzić działanie w sytuacji kiedy zminimy locale w przeglądarce

# Lab 1\_08 – Walidacja formularzy

# Cel ćwiczenia:

Walidacja formularzy

- 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

- 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 błedy na formularzu

# Lab 1\_10 - Widok PDF

# Cel ćwiczenia:

• Praca z innymi typami widoków

- 1. Skonfigurować BeanNameViewResolver
- 2. Dodać bean ProductListPDFView który dziedziczy po AbstractPdfView i korzystająć 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

- 1. Dodaj kontroler RestPhoneController z mapowaniem /rest/product
- 2. Korzystająć 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. <a href="http://www.baeldung.com/swagger-2-documentation-for-spring-rest-api">http://www.baeldung.com/swagger-2-documentation-for-spring-rest-api</a>
- 2. Zależności
- 3. Config
- 4. Mapowanie resourceów

# Lab 1\_12.5 - RestClient

# Cel ćwiczenia:

 Wykorzystać stworzenę webservice restowe za pomocą RestTemplate

- 1. <a href="http://www.baeldung.com/swagger-2-documentation-for-spring-rest-api">http://www.baeldung.com/swagger-2-documentation-for-spring-rest-api</a>
- 2. Zależności
- 3. Config
- 4. Mapowanie resourceów

# Lab 1\_13 - Hibernate - podstawy

# Cel ćwiczenia:

• Stworzenie HibernateProductRepository który będzie zapisaywał produkty w bazie danych

- 1. Przeanalizować zawartość pakietu pl.altkom.shop.model
- 2. Przeanalizować konfigurację pl.altkom.shop.DBConfig
- 3. Zaimplementować operację z ProductRepository za pomocą Hibernate'a
- 4. Sprawdzić działanie za pomocą aplikacji

# Lab 1\_13 - Hibernate - zapis

## Cel ćwiczenia:

 Stworzenie serwisu SaleDocumentService który umożliwi wystawienie faktury

- 1. Dodać serwis który powinien otrzymać informację ile sztuk jakiego produktu chcemy kupić
- 2. Wytorzyć brakujące obiekty Repository dla SaleDocumet oraz SaleDocumentItem
- 3. Korzystając z encji Product, Sale Documet, Sale Document Item oraz ich repozytoriów wytworzyć odpowiedni zapis w bazie danych
- 4. Wystawienie faktury powinien skutkować zmniejszeniem stanów (quantity) na produktach
- 5. Czy umiesz wydzielić do klasy bazowej BaseRepo metody powtarzające się w repozytoriach

# Lab 1\_14 - Hibernate - zapytania

## Cel ćwiczenia:

Wytworzenie w SaleDocumentRepo metody umożliwiającyh zaawansowane wyszukiwanie dokumentów

# Metody:

- 1. Wyszukiwanie dokumentu po numerze
- 2. Wyszukiwanie dokumentów przekraczających kwotę z parametru
- 3. Zwrócenie dokumentów posotrowanych po kwocie malejąco
- 4. Zwrócenie dokumentów posiadających co najmniej 3 pozycje
- 5. Zwrócenie 10 pierwszych dokumentów
- 6. Zwrócenie x pierwszych dokumentów pomijając y wierszy gdzie x,y parametry
- 7. Pobranie dokumentu od razu z wszystkimi pozycjami

# Lab 1\_15 - AOP

# Cel ćwiczenia:

- Stworzenie prostego aspektu do monitorowania aplikacji Metody:
  - 1. Dodaj adnotację Monitroing
  - 2. Dodaj Aspect który będzie tropił wszystkie wowłania metod oznaczonych adnotacją Monitoring i wypisywał je na konsole za pomocą log4j

```
@Around("@annotation(pl.altkom.shop.aop.Monitoring)")
public Object monitpr(ProceedingJoinPoint pjp) throws Throwable {
    Object obj = pjp.proceed();
    return obj;
}
```

# Lab 1\_16 - AOP - rozbudowa

# Cel ćwiczenia:

 Dodanie parametru do aspektu Monitoring mówiącego o maksymalnym czasie wykonania metody. W razie przegroczenia logować na konsole na poziomie ERROR

# Metody:

1. Dodaj parametr do adnotacji Monitroing

# Lab 1\_17 - AOP - cache

# Cel ćwiczenia:

 Dodać adnotację ResponseCache która będzie cache'ować wyniki wywołania metod oznaczonych adnotacją

# Metody:

- 1. Dodaj adnotację ResponseCache
- 2. Dodać aspekt cache'ujący

# Lab 1\_18 - Transakcje

# Cel ćwiczenia:

- Skorzystać z adnotacji @Transactional w opowiednich miejscach Metody:
  - 1. Na poziomie serwisów dodać adnotację @Transactionl
  - 2. W repo skorzystać z

@PersistenceContext

EntityManager em;

- 3. Sprawdzić działanie w SaleDocumentService czy wyjątek wycofuje wszystkie zmiany
- 4. Jak obsługiwać wyjątki w transakcji/ korzystać z try/catch

# Lab 1\_19 - Transakcje readOnly

# Cel ćwiczenia:

 Skorzystać z adnotacji @Transactional(readOnly) dla metod odczytujacych

- Na poziomie metod odczytujących dodać adnotację @Transactional(readOnly=true)
- 2. Co się stanie jeżeli bęziemy chcieli zapisać coś w takiej metodzie
- 3. Co się stanie jeżęli uruchomimy metodę modyfikującą z serwisu i w trakcie zawołamy metodą tylk odczytującą

# Lab 1\_20 - Bezpieczoństwo wywołań metod

# Cel ćwiczenia:

- Zapoznać się z podstawową konfiguracją Spring-security
- Umożliwośc wywołanie funkcji dodaj produkt tylko użytkowniką z rolą ADMIN

- 1. Dodać do metody insert serwisu od produktów odpowiednią adnotację
- 2. Sprawdzić działąnia aplikacji podczas logowania się na użytkoniwka z rolą USER oraz ADMIN

# Lab 1\_21 – Bezpieczoństwo WWW – własna strona logowania

# Cel ćwiczenia:

 Dodać możliwość logowania za pomocą specjalnie przygotowanej strony login.jsp

- 1. W sekcji <a href="http">ttp</a> dodać <form-login login-page="/login" />
- 2. Sprawdzić działanie nowej strony logowania

# Lab 1\_21 - Bezpieczoństwo WWW - webservice

# Cel ćwiczenia:

 Dodać zabezpieczenie dla webserviceów aby logowały się do systemu za pomocą BasicAuthentication

## Kroki:

1. Dodać nowy wpis w pliku security.xml który sla wzroca /rest/\*\* użyje logowania za pomocą <br/> <br/>basic-auth/>

# Lab 1\_22 – Bezpieczoństwo WWW – ukrywanie elementów

# Cel ćwiczenia:

 Pokazywać akcję "Dodaj nowy produkt" tylko dla użytkowników z rolą ADMIN

## Kroki:

1. Korzystająć z <authz:authorize ifAllGranted="ROLE\_ADMIN"> ukryć akcję