

Wytwarzanie aplikacji w Javie przy pomocy Spring Framework - zagadnienia podstawowe

Ćwiczenia

Przygotowanie środowiska:

Kroki:

1. Zainstaluj JDK w wersji 6 używając standardowych opcji instalacyjnych
2. Ustaw zmienną środowiskową użytkownika JAVA_HOME tak, by wskazywała katalog z JDK
3. Zainstaluj Mavena rozpakowując jego archiwum do wybranego katalogu np. C:\maven
4. Do zmiennej systemowej PATH dodaj wskazanie na katalog <katalog_mavena>/bin
5. Sprawdź działanie mavena wpisując w konsoli polecenie mvn
6. Aby przyspieszyć proces budowania rozpakuj archiwum repository.zip do katalogu z danymi mavena <katalog_domowy>/m2
7. Zainstaluj Netbeans w wersji Java
8. Uruchom środowisko Netbeans

Lab 1_00 – Konfiguracja aplikacji Spring

Cel ćwiczenia:

Budowa aplikacji w oparciu o Springa.

Tworzenie deskryptorów i dostęp do beanów

Kroki:

1. Pobierz moduł `lab00`.
2. W aplikacji są zdefiniowane dwie klasy: `Person` i `Address`. Zapewnij, że klasy te będą dostępne, jako beany w kontenerze. Dla klasy `Person` skorzystaj z deklaracji w deskrytorze xml. Dla klasy `Address` skorzystaj z adnotacji.
3. W klasie `Lab0`
 1. Skonfiguruj prosty kontener.
 2. Pobierz referencję do beanów i wyświetl ich właściwości.

Lab 1_01 – Ustawianie właściwości

W poprzednim ćwiczeniu właściwości obiektów `Person` i `Address` były zaszyte w aplikacji.

Cel ćwiczenia:

Konfiguracja właściwości beanów w deskrytorze

Wstrzykiwanie zależności przez settery i konstruktor

Kroki:

1. Pobierz moduł `lab01`.
2. Zapoznaj się z konfiguracją maven
3. Uzupełnij deskryptor `context.xml` o wpisy dotyczące beanów `Person` i `Address`. Wpisz własne dane, jako własności.
 1. Dla `Address` własności ustaw przez settery.
 2. Dla `Person` własności ustaw przez konstruktor.

Lab 1_02 – Powiązania między beanami

Cel ćwiczenia:

Tworzenie powiązań między beanami w deskrypcji

Kroki:

1. Pobierz moduł `lab02`.
2. Do klasy `Entry` wstrzyknij zależne obiekty klas `Person`, `Address` i `Phone`.
 1. Obiekt klasy `Person` wstrzyknij przez seter, jako beana zewnętrznego
 2. Obiekt klasy `Address` wstrzyknij przez seter, jako beana wewnętrznego.
 3. Wstrzyknięcie obiektu klasy `Phone` zapewnij przez zastosowanie odpowiednio adnotacji `@Autowired`

Lab 1_03 – Wstrzykiwanie kolekcji

Cel ćwiczenia:

Wstrzykiwanie do beana zależności, które są kolekcjami

Kroki:

1. Pobierz moduł `lab03`.
2. W deskrypcji przypisz do repozytorium pierwszego wpisy `entry1`, `entry2`, a do repozytorium drugiego `entry3`, `entry4`.
3. Wpisy w repozytorium pierwszym wstrzyknij jako listę, natomiast wpisy w repozytorium drugim jako mapę.

Lab 1_04 – Postprocessing

Cel ćwiczenia:

Modyfikacja właściwości beanów poprzez wykorzystanie mechanizmu postprocessing

Kroki:

1. Pobierz moduł `lab04`.
2. Uzupełnij implementację klasy `PhonePostProcessor` tak, by działała jako `PostProcessor`. Dla wszystkich beanów posiadających właściwości o nazwie `phone` i typie `Phone` wyświetl na konsoli telefon w postaci `+<kod kraju> (<kierunkowy>) xxx-xx-xx`.
3. Zapewnij wykorzystanie `PostProcessor`a przez kontener.

Lab 1_05 – Własny edytor właściwości

W poprzednim ćwiczeniu beany adresu i osoby były deklarowane jako beany wewnętrzne, w tym ćwiczeniu wykorzystamy możliwość konwersji `String`a na docelowe typy danych `Address` i `Person`

Cel ćwiczenia:

Wstrzykiwanie do beana zależności, które powinny automatycznie skonwertowane do obiektu

Kroki:

1. Pobierz moduł `lab05`.
2. Zaimplementuj metody `setAsText` klas `AddressEditor` i `PersonEditor`. W implementacji zwróć uwagę na wpisy deskryptorze.
3. Zdefiniuj i zarejestruj edytory właściwości w kontenerze

Lab 1_06 – Internacjonalizacja i wiązanie beanów z kontenerem

Cel ćwiczenia:

- Uniezależnienie właściwości beanów od nazw narodowych.
- Zmiana wyświetlanych tekstów w zależności od kraju i języka.
- Wykorzystanie wiązania beanów z kontenerem

Kroki:

1. Pobierz moduł `lab06`.
2. W deskrytorze dodaj beana `messageSource`. Odpowiednie pliki w katalogu `locale` pozwolą odgadnąć wartości parametrów.
3. Przetestuj działanie aplikacji
4. Dodaj nowy język do aplikacji: zrefaktoruj odpowiednio działanie metody `ComponentMediator.doLocalization()`
5. Przenieś obiekt `ComponentMediator` do kontenera. `AddressBookWindowBuilder` powinien pobierać ten obiekt z kontenera.

Lab 1_07 – Metody inicjalizujące

Cel ćwiczenia:

- Zastosowanie metod inicjalizujących i finalizujących.

Kroki:

1. Pobierz moduł `lab07`.
2. Do klasy `InMemoryRepository` dodaj metodę inicjalizującą i wykonaj w niej sprawdzenie czy pole `entries` zostało zainicjowane. Jeśli nie, metoda powinna rzucać wyjątek `IllegalStateException`.
3. Dodaj również metodę finalizującą i wypisz w niej komunikat o niszczeniu klasy.
4. Zadanie wykonaj używając atrybutu `init-method` oraz adnotacji `@PreDestroy`.

Lab 1_08 – Konfiguracja aplikacji webowej

Cel ćwiczenia:

Konfiguracji deskryptorów Spring tak, aby działały w kontekście aplikacji webowej

Kroki:

1. Pobierz moduł lab08
2. Zapoznaj się ze zmianą konfiguracji maven
3. Utwórz deskryptor Springa w katalogu `src/main/webapp/WEB-INF`
4. Skonfiguruj parametr kontekstu w pliku `web.xml` tak, aby ładowane były deskryptory Spring'a.
5. Skonfiguruj `ServletDispatcher`'a tak, aby przechwytywał wszystkie URLe zakończone na `*.do`
6. W deskryptorze Springa:
 1. Skonfiguruj prostego `viewResolvera`, aby obsługiwał JSTL.
 2. Skonfiguruj kontroler na link `/home.do` dla klasy `DefaultController`. Ustaw właściwości kontrolera tak, aby przekierowywał na stronę `home.jsp`
7. Wykonaj deployment aplikacji webowej.
8. Do przeglądarki wpisz adres:
http://localhost:8080/<nazwa_projektu>/home.do

Lab 1_09 – Pobieranie danych z formularza

Cel ćwiczenia:

Uczestnicy szkolenia będą rozwijali aplikację Książka Telefoniczna
Pobranie danych od użytkownika i zapisanie ich w książce telefonicznej

Kroki:

1. Pobierz moduł lab09
2. Zapoznaj się ze strukturą aplikacji i elementami już zaimplementowanymi.
3. Stwórz kontroler o nazwie `AddEntryController`.

Celem działania kontrolera ma być: przyjęcie requestu `/addEntry.do`, obsługa zapisu danych wprowadzonych przez użytkownika i przekierowanie do strony

startowej.

W implementacji posłuż się adnotacjami.

4. Rozbuduj stronę `addEntryForm.jsp` tak, aby pobrać wszystkie dane od użytkownika i zapisać je do obiektu `Entry`

Lab 1_10 – Internacjonalizacja aplikacji

Cel ćwiczenia:

Uniezależnienie aplikacji od używanego języka

Kroki:

1. Pobierz moduł `lab10`
1. Dodaj plik `properties` i umieść w nim wszystkie teksty używane w formularzu
2. Zdefiniuj obiekt `messageSource`
3. Na stronie `addEntryForm.jsp` odwołaj się do tekstów wpisanych w pliku `properites`

Lab 1_11 – Walidacja formularzy

Cel ćwiczenia:

Zabezpieczenie aplikacji przed podaniem nieprawidłowych danych przez użytkownika

Kroki:

1. Pobierz moduł `lab11`
2. Napisz walidator dla formularza, który wymusi uzupełnienie wszystkich pól
3. Na stronie `addEntryForm.jsp` dodaj znacznik pozwalający wypisać błędy

Lab 1_12 – Konfiguracja widoku

Cel ćwiczenia:

Wygenerowanie raportu z książki telefonicznej w formacie PDF

Kroki:

1. Pobierz moduł `lab12`
2. Zdefiniuj w deskrytorze resolver odpowiedzialny za przekierowywanie obsługi żądań do beanów
3. Klasa `EmployeesPdfReportView` służy do generowania raportu w formacie PDF, umieść ją w kontenerze
4. Napisz kontroler, który:
 1. Przechwyci żądanie `<server>/phonebook/generateReport.do`
 2. Pobierze zawartość książki telefonicznej z repozytorium
 3. Umieści pobrane wpisy w modelu pod nazwą odpowiadającą nazwie widoku generującego raport PDF
 4. Przetestuj działanie aplikacji
 5. Czy potrafisz dodać proste menu, które pojawi się w momencie uruchomienia aplikacji? Menu powinno mieć możliwość wyboru dodawania wpisów albo wygenerowania raportu.

Lab 1_13 – Zapisywanie danych w bazie

Cel ćwiczenia:

Zapewnienie trwałego przechowywania danych w bazie danych przy pomocy JDBC

Napisanie implementacji interfejsu `PhoneBookRepository`, który będzie zachowywał w bazie danych obiektu `Entry`

Kroki:

1. Pobierz moduł `lab13`
2. Zapoznaj się ze zmianą konfiguracji maven
3. Konfiguracja bazy danych:
 1. Przy pomocy narzędzia NETBEANS-DERBY utwórz bazę danych `phonebook`

2. W bazie danych `phonebook` w schemacie `APP`, utwórz tabelę o nazwie `pb_persons`
3. W tabeli utwórz pola:
 - `name` – typu `varchar`
 - `surname` – typu `varchar`
4. W deskrytorze `phonebook-repository.xml` zdefiniuj źródło danych o parametrach. Parametry pobierz z pliku `jdbc.properties`
5. Zaimplementuj metody `save` oraz `findAllEntries` klasy `JdbcRepository`. Metody powinny zapisywać obiekt `Entry.person` do tabeli `pb_persons`.
6. Uruchom aplikację i przetestuj jej działanie stan bazy danych możesz podejrzeć przy pomocy narzędzia `NETBEANS`.

Lab 1_14 – Integracja aplikacji webowej z Hibernate

Cel ćwiczenia:

Zapewnienie trwałego przechowywania danych w bazie danych, przy pomocy biblioteki `Hibernate`

Kroki:

1. Pobierz moduł `lab14`
2. W deskrytorze `phonebook-repository.xml` skonfiguruj obiekt klasy `HibernateTemplate`
3. Uzupełnij obiekt `Entry` o zapytania nazwane:
 - pobierające wszystkie obiekty `Entry` z bazy
4. Napisz implementację interfejsu `PhoneBookRepository` wykorzystującą obiekt `HibernateTemplate` do zapewnienia trwałego przechowywania danych
5. Wstrzyknij napisaną implementację do odpowiedniego kontrolera
6. Uruchom i przetestuj działanie aplikacji
7. Czy potrafisz napisać wyszukiwanie wpisów w książce telefonicznej ze względu na nazwisko?

Lab 1_15 – Logowanie

Cel ćwiczenia:

Zaimplementowanie funkcji logowania za pomocą Spring AOP

Kroki:

1. Pobierz moduł `lab15`
2. Zapoznaj się ze zmianą konfiguracji maven
3. W deskrytorze `phonebook-service.xml` włącz wsparcie dla adnotacji AspectJ
4. Zaimplementuj aspekt `altkom.aspect.LoggerAspect`
 1. aspekt powinien logować wywołania metod odwołujących się do bazy danych
 2. logowane powinny być: nazwa klasy, nazwa metody, aktualne parametry metody oraz rezultat działania metody

Lab 1_16 – Transakcje

Cel ćwiczenia:

Zapewnienie transakcyjności w aplikacji

Kroki:

1. Pobierz moduł `lab16`
2. Dodaj transakcyjność do metod pracujących bezpośrednio na źródle danych
 1. Skonfiguruj menadżera transakcji w deskrytorze
 2. Dodaj odpowiednie adnotacje do odpowiednich metod