



Модуль 08 - Piscine Java

Весна

Резюме: Сегодня вы узнаете о разработке Java на уровне предприятия и основах фреймворка Spring.

Содержание

I	Предисловие	2
II	Инструкции	3
III	Упражнение Весенний контекст 00 :	5
IV	Упражнение JdbcTemplate 01 :	7
V	Упражнение AnnotationConfig 02 :	9

Глава I

Предисловие

Spring Framework является неотъемлемой частью большинства корпоративных систем на базе Java. Этот фреймворк значительно упрощает конфигурирование приложений и связь компонентов друг с другом. Благодаря этому разработчик может полностью сосредоточиться на реализации бизнес-логики.

Принцип работы Spring полностью основан на паттернах DI/IoC, о которых вам следует узнать, прежде чем использовать эту технологию.

Центральным понятием в Spring framework является бин (компонент), который представляет объект внутри контейнера ApplicationContext. Контейнер также создает связи между бинами.

Существует несколько способов конфигурирования бункеров:

1. Использование xml-файла.
2. Использование конфигурации Java (конфигурирование с помощью аннотаций).
3. Комбинированная конфигурация.

XML-конфигурация позволяет изменять поведение приложения без пересборки. В свою очередь, конфигурация Java делает код более удобным для разработчиков.



Глава II


Инструкции

- Используйте эту страницу как единственную ссылку. Не слушайте никаких слухов и домыслов о том, как приготовить раствор.
- Теперь для вас существует только одна версия Java - 1.8. Убедитесь, что компилятор и интерпретатор этой версии установлены на вашей машине.
- Вы можете использовать IDE для написания и отладки исходного кода.
- Код чаще читают, чем пишут. Внимательно прочитайте [документ](#), в котором приведены правила форматирования кода. При выполнении каждой задачи убедитесь, что вы следуете общепринятым [стандартам Oracle](#)
- Комментарии не допускаются в исходном коде вашего решения. Они затрудняют чтение кода.
- Обратите внимание на разрешения ваших файлов и каталогов.
- Для оценки ваше решение должно находиться в вашем GIT-репозитории.
- Ваши решения будут оценивать ваши товарищи по аквариуму.
- Вы не должны оставлять в своем каталоге никаких других файлов, кроме тех, которые явно указаны в инструкциях к упражнению. Рекомендуется изменить свой .gitignore во избежание несчастных случаев.
- Когда вам нужно получить точный вывод в ваших программах, запрещено выводить предварительно рассчитанный вывод вместо правильного выполнения упражнения.
- У вас есть вопрос? Спросите своего соседа справа. В противном случае попробуйте поговорить с соседом слева.
- Ваше справочное пособие: товарищи / Интернет / Google. И еще кое-что. На любой ваш вопрос есть ответ на Stackoverflow. Узнайте, как правильно задавать вопросы.
- Внимательно прочитайте примеры. В них могут потребоваться вещи, которые не указаны в предмете.
- Используйте "System.out" для вывода

- И да пребудет с вами Сила!
- Никогда не оставляйте на завтра то, что вы можете сделать сегодня ;)

Глава III

Упражнение 00: Весенний контекст

	Упражнение 00
	Весенний
Каталог для сдачи : ex00/	
Файлы для сдачи : Spring- folder Разрешенные функции : Все	контекст

Давайте реализуем слабосвязанную систему, состоящую из набора компонентов (бинов) и соответствующую принципам IoC/DI.

Предположим, что существует интерфейс `Printer`, предназначенный для отображения определенного сообщения.

У этого класса есть две реализации: `PrinterWithDateTimeImpl` и `PrinterWithPrefixImpl`. Первый класс выводит сообщения, указывая дату/время вывода с помощью `LocalDateTime`, а второй класс можно использовать для задания текстового префикса сообщения.

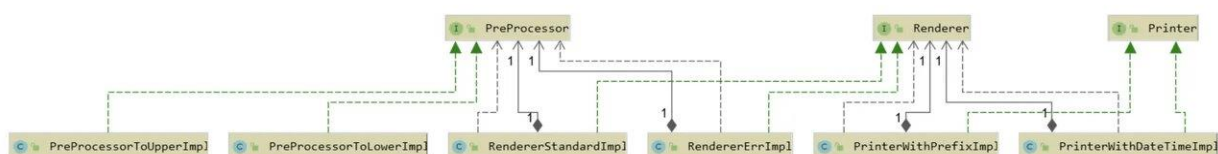
В свою очередь, обе реализации `Printer` имеют зависимость от интерфейса `Renderer`, который

отправляет сообщения на консоль. `Renderer` также имеет две реализации:

`RendererStandardImpl` (выводит сообщение через стандартный `System.out`) и `RendererErrImpl` (выводит сообщения через `System.err`).

`Renderer` также имеет зависимость от интерфейса `PreProcessor`, который осуществляет предварительную обработку сообщений. Реализация `PreProcessorToUpperImpl` переводит все буквы в верхний регистр, а реализация `PreProcessorToLower` переводит все буквы в нижний регистр.

UML-диаграмма классов показана ниже:



Пример кода, использующего эти классы стандартным образом:


```
public class Main {  
    public static void main(String[] args) {  
        PreProcessor preProcessor = new PreProcessorToUpperImpl();  
        Renderer renderer = new RendererErrImpl(preProcessor);  
    }  
}
```

```
PrinterWithPrefixImpl printer = new PrinterWithPrefixImpl(renderer);
принтер . setPrefix ("Префикс");
принтер . print("Hello!");
    }
}
```

Выполнение этого кода приведет к следующему

результату: **PREFIX HELLO**


- Вам необходимо описать файл context.xml для Spring, в котором будут указаны все настройки для каждого компо-
и связи между ними будут указаны.

Использование этих компонентов с Spring

```
public class Main {
    public static void main(String[] args) {
        ApplicationContext context = новый ClassPathXmlApplicationContext("context.xml");
        Printer printer = context.getBean("printerWithPrefix", Printer.class);
        принтер . print("Hello!");
    }
}
```


Глава IV

Упражнение 01 : Шаблон JdbcTemplate

	Упражнение 01
Шаблон JdbcTemplate	
Входящий каталог : <i>ex01/</i>	
Файлы для сдачи : Сервисная	
папка Разрешенные функции	
: Все	

JdbcTemplate и его расширение NamedParameterJdbcTemplate являются удобными механизмами для работы с базами данных. Эти классы позволяют отказаться от написания шаблонного кода для выполнения и обработки запросов, а также от необходимости перехвата исключений при проверке.

Кроме того, они предоставляют удобную концепцию RowMapper для обработки ResultSet и преобразования результирующих таблиц в объекты.

Теперь необходимо реализовать модель User со следующими полями:

- Идентификатор
- Электронная почта

Вам также необходимо реализовать интерфейс CrudRepository<T> со следующими методами:

- T findById(Long id)
- List<T> findAll()
- void save(T entity)
- void update(T entity)
- void delete(Long id)

Интерфейс UsersRepository, объявленный как UsersRepository extends CrudRepository<User>, должен содержать следующий метод:

- Optional<T> findByEmail(String email)

Кроме того, необходимы две реализации `UsersRepository`: `UsersRepositoryJdbcImpl` (использует стандартные механизмы `Statements`) и `UsersRepositoryJdbcTemplateImpl` (основан на `JdbcTemplate/NamedParameterJdbcTemplate`). Оба класса принимают объект `DataSource` в качестве аргумента конструктора.

В файле `context.xml` должны быть объявлены бины для обоих типов хранилищ с различными идентификаторами, а также два бина типа `DataSource`: `DriverManagerDataSource` и `HikariDataSource`.

Кроме того, данные для подключения к БД должны быть указаны в файле `db.properties` и включены в `context.xml` с помощью заполнителей `${db.url}`.

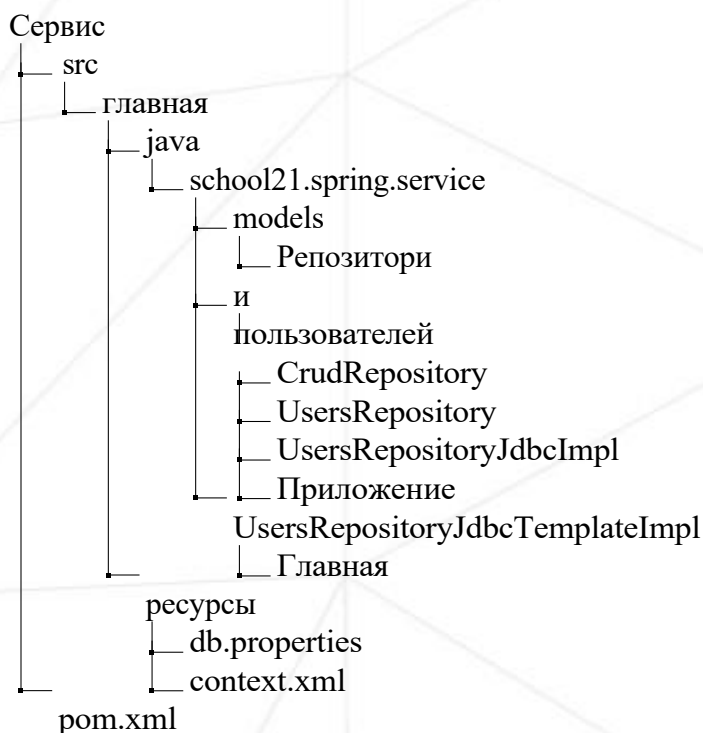
Пример `db.properties`:

```
db.url=jdbc:postgresql://localhost:5432/database
db.user=postgres
db.password=qwerty007
db.driver.name=org.postgresql.Driver
```

В классе `Main` работа метода `findAll` должна быть продемонстрирована с использованием обоих хранилищ:


```
ApplicationContext context = new ClassPathXmlApplicationContext("context.xml");
UsersRepository usersRepository = context.getBean("usersRepositoryJdbc", UsersRepository.class);
System.out.println(usersRepository.findAll());
usersRepository = context.getBean("usersRepositoryJdbcTemplate", UsersRepository.class);
System.out.println(usersRepository.findAll());
```

Структура проекта:



Глава V

Упражнение 02 : AnnotationConfig

	Упражнение 02
	AnnotationConfig
Входящий каталог : <i>ex02/</i>	
Файлы для сдачи : Сервисная	
папка Разрешенные функции	
: Все	

Теперь необходимо настроить механизмы конфигурации Spring-приложения с помощью аннотаций. Для этого используйте класс конфигурации, помеченный как `@Configuration`. Внутри этого класса необходимо описать бины для подключения к `DataSource` DB с помощью аннотации `@Bean`. Как и в предыдущем задании, данные подключения должны быть расположены в файле `db.properties`. Также необходимо убедиться, что файл `context.xml` отсутствует.

Также реализуйте пару интерфейс/класс `UserService/ServiceImpl` с зависимостью от объявленного в ней `UsersRepository`. Вставка правильного бина репозитория должна быть реализована с помощью аннотации `@Autowired` (аналогично необходимо привязать `DataSource` внутри репозитория). Коллизии в автоматическом связывании должны быть разрешены с помощью аннотации `@Qualifier`.

Бины для `UserService` и `UsersRepository` должны быть определены с помощью аннотации `@Component`.

В `ServiceImpl` реализуйте метод `String signUp(String email)` который регистрирует нового пользователя и сохраняет его данные в БД. Этот метод возвращает временный пароль, назначенный пользователю системой (эта информация также должна быть сохранена в базе данных).

Чтобы проверить, правильно ли работает ваш сервис, реализуйте интеграционный тест для `ServiceImpl` с использованием базы данных in-memory (H2 или HSQLDB). Конфигурация контекста для тестовой среды (`DataSource` для базы данных in-memory) должна быть описана в отдельном классе `TestApplicationConfig`. Этот тест должен проверить, был ли возвращен временный пароль в методе `signUp`.

Структура проекта:

Сервис
└─ src

