Test task – описание реализации

Описание сопровождается ссылками на GitHub

Оглавление

[Описание базы данных 2](#_Toc384319273)

[Web framework 3](#_Toc384319274)

[Краткое описание классов 3](#_Toc384319275)

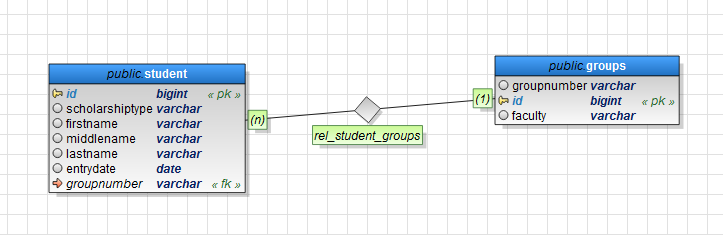
[Список использованных источников 4](#_Toc384319276)

[Приложение A – диаграмма классов 5](#_Toc384319277)

# Описание базы данных

Для выполнения задания взята простая модель БД – вариант 1.

1. Институт (группы): Студент (ФИО\*, группа\*, тип стипендии, дата зачисления), Группа (Номер\*, Факультет).



Так как по условию задания таблица 2 не редактируемая, то связь между entity бинами Student и Groups реализована как однонаправленная [ManyToOne](https://github.com/vladiksun/TestTask/blob/master/src/main/java/com/testTask/entity/Student.java), что позволяет JPA провайдеру генерировать следующий SQL запрос

CREATE TABLE student

(

id bigint NOT NULL,

entrydate date,

firstname character varying(255),

lastname character varying(255),

middlename character varying(255),

scholarshiptype character varying(255),

id\_group bigint,

CONSTRAINT student\_pkey PRIMARY KEY (id),

CONSTRAINT fk\_25ymf4j36i62y3wlmupcg3fom FOREIGN KEY (id\_group)

REFERENCES groups (id) MATCH SIMPLE

ON UPDATE NO ACTION ON DELETE NO ACTION

Для взаимодействия с БД используется JPA провайдер – Hibernate. Настройка с помощью [persistence.xml](https://github.com/vladiksun/TestTask/blob/master/src/main/resources/META-INF/persistence.xml)

При запуске приложения провайдер подключается к заранее созданной БД в [PostgreSQL – tesktask](https://github.com/vladiksun/TestTask/blob/master/src/main/resources/META-INF/persistence.xml).

Для целей разработки hibernate настроен таким образом, что при развертывании приложения на сервере (используется WildFly 8 Final c предварительно настроенным JNDI) таблицы создаются, а при удалении приложения удаляются из БД.

Генерация id настроена как @GeneratedValue(strategy=GenerationType.AUTO). Данная стратегия генерации id используется только для разработки. В рабочем режиме рекомендуется использовать другие стратегии генерации id –

* Id Generation Using a Table
* Id Generation Using a Database Sequence
* Id Generation Using Database Identity

# Web framework

Для выполнения задания используется веб фреймворк JSF последней версии. Работа данного фреймворка основана не на управлении запросами, а на использовании компонентов.

JSF является стандартом для приложений Java EE и активно развивается Oracle и IBM. Для него на данный момент создано множество библиотек, позволяющих использовать нестандартные UI компоненты, основанные на jQuery. Одна из таких библиотек используется для тестового задания – [PrimeFaces.](http://primefaces.org/)

Выбор данного фреймворка также обусловлен возможностью использования встроенного механизма валидации введенных пользователем данных и поддержкой ajax, что позволяет обращаться к серверу на лету без перезагрузки всей страницы.

Навигация между страницами осуществляется также механизмами JSF – с помощью заранее прописанных правил навигации в [faces-config.xml](https://github.com/vladiksun/TestTask/blob/master/src/main/webapp/WEB-INF/faces-config.xml)

Также в этом файле прописаны настройки ресурс бандла, который содержит заготовки текстовых сообщений для вставки в методы отправки сообщений.

# Краткое описание классов

Диаграмма классов в приложении A

Описание классов присутствует в JavaDoc на английском языке.

* **package com.testTask.controller;**;
  + **public abstract class AbstractController implements Serializable** – базовый класс для объектов – контроллеров. Включает в себя 2 метода для генерации сообщений для пользователя. Один метод напрямую принимает строку сообщения как аргумент. Второй метод принимает ссылку на строку в ресурс бандле где содержаться заранее подготовленный сообщения.
  + **public class TableController extends AbstractController** – cdi бин-контроллер для главной страницы приложения. Расширяет класс **AbstractController.** Включает в себя все необходимые поля и методы для редактирования записи в БД. Подробное описание см в java doc.
  + **public class CreateController extends AbstractController** – cdi бин-контроллер для страницы создания записи в БД. Подробное описание методов см в java doc.
* **package com.testTask.entity;**
  + **public class Groups implements Serializable**  - entity класс для мапинга таблицы Groups.
  + **public class Student implements Serializable** - entity класс для мапинга таблицы Student.
* **package com.testTask.service;**
  + **public interface TableService**  - основной и единственный интерфейс приложения. Описывает совокупность методов для доступа к базе данных с целью создания, удаления и редактирования записей в БД.
* **package com.testTask.service.impl;** 
  + **public abstract class AbstractService**  - базовый абстрактный класс, который расширяют все сервис классы (в данным приложении только один сервис класс). Инкапсулирует Enitity manager для общения с БД.
  + **public class TableServiceImpl extends AbstractService implements TableService** – сервисный EJB stateless бин, который расширяет **AbstractService** и имплементирует методы описанные в интерфейсе **TableService** методы. Подробное описание методов см в java doc.
* **package com.testTask.utils;**
  + **public class ConstantValuesHolder**  - CDI бин со скоупом @ApplicationScoped. Создается контейнером при первом использовании и существует пока приложение развернуто на сервере. Служит для одной цели – хранить величины для выпадающих списков. Также в этот бин включены массивы объектов – компонентов типа SelectItem[] для дальнейшего использования механизмом фильтрации от PrimeFaces (<http://www.primefaces.org/showcase/ui/datatableFiltering.jsf>).
  + **public class Credentials** – CDI бин со скоупом @RequestScoped. Используется для удобства сборки информации о студенте при создании и редактировании записей в БД.
  + **public class TestDataGenerator** – EJB singleton бин созданный для цели разработки. Служит для заполнения БД одной и той же первоначальной информацией при старте приложения. В дальнейшем планируется использовать для изучения написания тестов. Бин содержит интерцептор @PostConstruct на методе insert() что позволяет вызвать метод при старте приложения. Бин стартует при старте приложения.

# Список использованных источников

Использовались книги в электронном виде

[Introduction to Java Programming, Comprehensive Version by Y. Daniel Liang](http://www.amazon.com/Introduction-Programming-Comprehensive-Version-Edition/dp/0132936526/ref=sr_1_1?ie=UTF8&qid=1396540904&sr=8-1&keywords=Introduction+to+Java+Programminghttp://www.cs.armstrong.edu/liang/index.html)

[Arun Gupta - Java EE 7 Essentials](http://www.amazon.com/Java-EE-Essentials-Arun-Gupta/dp/1449370179/ref=sr_1_1?ie=UTF8&qid=1396541062&sr=8-1&keywords=Arun+Gupta+-+Java+EE+7+Essentials)

[Goncalves A. - Beginning Java EE 7 (The Expert's Voice in Java) - 2013](http://www.amazon.com/Beginning-Java-EE-Antonio-Goncalves/dp/143024626X/ref=sr_1_fkmr0_1?ie=UTF8&qid=1396541120&sr=8-1-fkmr0&keywords=Goncalves+A.+-+Beginning+Java+EE+7+%28The+Expert%27s+Voice+in+Java%29+-+2013)

[Juneau J. - Java EE 7 Recipes (The Expert's Voice in Java) - 2013](http://www.amazon.com/Java-EE-Recipes-Problem-Solution-Approach-ebook/dp/B00ACC69NS/ref=sr_1_fkmr2_1?ie=UTF8&qid=1396541174&sr=8-1-fkmr2&keywords=Juneau+J.+-+Java+EE+7+Recipes)

[Pro JPA 2](http://www.amazon.com/Pro-JPA-2-Mike-Keith/dp/1430249269/ref=sr_1_1?ie=UTF8&qid=1396541221&sr=8-1&keywords=Pro+JPA+2) by Mike Keith and Merrick Schincariol

[Java ee7 tutorial](http://docs.oracle.com/javaee/7/tutorial/doc/home.htm)

[Primefaces users guide](http://primefaces.org/documentation)

[Various Useful WildFly BOMs](https://github.com/wildfly/boms)

<https://github.com/wildfly/quickstart>

# Приложение A – диаграмма классов

