# ВВЕДЕНИЕ

Проблема образования в нашем мире является одной из самой обсуждаемых. Многие родители хотят, чтобы их дети получили релевантное образование, но не знают, каким образом можно повлиять на ребенка и стимулировать его к обучению.

Последнее и было главной задумкой при создании дипломного проекта. Стимулирование через интересные задания, красивое оформление и оплату мобильного телефона должны привлечь детей к обучению, а автоматизация проверки и ведение статистики позволят родителям наблюдать за успехами детей не тратя время на проверку и составление тестов.

Конечно, для удобства задания распределены по темам, а темы по предметам. В свою очередь у каждого предмета имеется атрибут «Класс» («schoolclass»). Этот атрибут был введен для регулирования сложности заданий и тем относительно друг друга. На данный момент программа содержит предметы для 11 различных классов, что соответствует российской учебной программе вплоть до полного среднего образования. Несмотря на всё это диапазон классов не является статичным и может быть расширен в случае, например, введения программы в пользование в стране с иной системой образования

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## Общая информация о проекте

### Описание проекта

Проект представляет из себя распределенное приложение, написанное на языке Java в соответствии со спецификацией Java Platform Enterprise Edition (Java EE) и запускаемое на сервере приложений Jboss (Wildfly), или любом другом, поддерживающем спецификацию Java EE в качестве серверной части (backend).

В качестве клиентской части (frontend) может выступать любое клиентское приложение, которое поддерживает обмен данными по протоколу HTTP и способно взаимодействовать в соответствии с архитектурным стилем REST.

### Описание основных компонентов и библиотек

Spring Framework – универсальный фреймворк для Java-платформы, интегрированный для упрощения реалиации инверсии управления (IoC), упрощения реализации паттерна программирования MVC (Spring MVC), упрощения доступа к данным (Spring Data), упрощения организации информационной безопасности (Spring Security).

Apache Maven – фреймворк для автоматизации сборки проекта на основе описания его структуры на языке POM (Project Object Model) в файле pom.xml

Log4j – библиотека журналирования (logging) Java-программ.

Java Platform Enterprise Edition (Java EE) API – набор различных программных интерфейсов, соответствующих спецификации Java EE, таких как JPA, Validation, JavaMail и других

Hibernate – реализация Java Persistence API. Является технологией ORM.

FasterXML/Jackson – библиотека для представления объектов в качестве JSON/XML строки или файла.

Apache Velocity – библиотека для создания шаблонов на языке Velocity Template Language.

OkHttp – библиотека для создания HTTP-клиентов и выполнения HTTP-запросов.

### Инверсия контроля и внедрение зависимостей

Внедрение зависимостей (Dependency Injection, DI) – одна из реализаций инверсии контроля (Inversion of Control, IoC) – процесс предоставления внешней зависимости программному компоненту.

Проще говоря, построение объектов производится внешним механизмом, который исходя из настроек построения внедряет те, или иные зависимости в соответствующие компоненты.

По такому принципу, например, работает Spring Framework. При развертывании приложения существует механизм, который исходя из XML-контекста, или аннотаций в Java-коде создает компоненты, содержащие ссылки на другие компоненты (внедрение зависимостей) и использующие их для реализации собственного функционала. См. Рис 1 и 2.

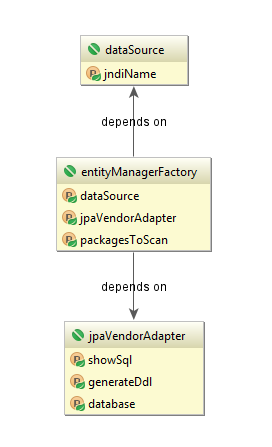
Рис. 1. Простейший пример внедрения зависимостей

Рис. 2. То же самое в схематичном виде

## Организация ввода-вывода и программного интерфейса

### Входные и выходные данные

Входными данными приложения являются HTTP-запросы, а выходные, соответственно, HTTP-ответы в соответствии со спецификацией RFC 2616 HTTP/1.1.

Внимание! Для использования кросс-доменных запросов убедитесь в том, что приложение настроено для их принятия и дает ответ с соответствующими заголовками ответа (response headers) на предварительный запрос методом OPTIONS.

Приложение может использовать в качестве данных для обработки такие части запроса, как переменные путей, параметры запроса (path variables и request parameters, представляющие из себя query string) и тело запроса (request body).

Сервер приложений принимает строку, определяет контекст к которому сделан запрос, соотносит это с приложением, которое находится в этом контексте и направляет его соответствующему сервлету (сервлеты объявлены в web.xml). Данное приложение построено на основе Spring MVC Framework, так что в данном случае управление по выбору обработчика обычно предоставляют сервлету dispathcerServlet (класс org.springframework.web.servlet.DispatcherServlet) см. Рис 3.

DispatcherServlet

Request

Response

Handler Mapping

RestController

Рис 3. Схема обработки запроса внутри приложения Spring MVC

Как можно увидеть из вышеприведенной схемы, DispatcherServlet является посредником между приложением основанном на технологии Spring MVC и сервером и занимается решением задач связанных с обработкой запроса. Так, например, в случае нашего приложения, где каждая функция обработки помечена аннотацией RestController, автоматически результат функции будет помечен как ResponseBody, что является поводом для преобразования возвращаемого объекта в JSON библиотекой jackson (или в любой другой медиа-тип, обозначенный в параметре produces аннотации RequestMapping). Если же не будет найдено аннотации RestController у класса, или ResponseBody у результата функции, то DispatcherServlet, по умолчанию действующий в соответствии с шаблоном MVC постарается обратиться к стандартной реализации ViewResolver и найти представление, сооответствующее возвращаемому String, Model, или ModelAndView значению. Как можно заметить из описания, сервлет-диспетчер действительно говорящее название для данного класса.

### Взаимодействие с базой данных

Приложение по умолчанию пытается найти базу данных в контексте ресурсов сервера по имени «java:/PostgresDS» (технология JNDI). Для того чтобы изменить способ подключения необходимо в конфигурационном файле «applicationContext.xml» указать для параметра «jndiName» бина (bean) «dataSource» иное JNDI имя, или создать новый бин класса, реализующего интерфейс DataSource, сконфигурировать его и добавить в качестве значения свойства «dataSource» в «entityManagerFactory».

Для взаимодействия с базой данных используется технология ORM (Object-Relation Mapping – объектно-реляционное отображение), позволяющая работать с записями в базе данных как с объектами Java. Для работы с базой данных по этой технологии в контексте данного приложения используется JPA (Java Persistence API) и реализация этой спецификации Hibernate.

Spring Data JPA – часть Spring Framework отвечающая за удобное взаимодействие с базой данных с помощью JPA. Spring Data JPA позволяет быстро и легко создать репозитории со стандартными CRUD-операциями. Для этого необходимо создать интерфейс и расширить с помощью него интерфейс JpaRepository, указав при этом класс сущности, с которой будет произоводиться взаимодействие и тип первичного ключа (Id). Кроме того, можно описывать собственные специфичные запросы к БД различными способами, такими как:

* Описание запроса в названии функции в интерфейсе
* Описание запроса на языке HQL в аннотации Query
* Описание запроса на языке SQL в аннотации Query (для этого необходимо установить значение аттрибута nativeQuery как true)

### Взаимодействие с сервером исходящей почты

Взаимодействие приложения с сервером исходящей почты производится по протоколу SMTP с помощью реализации интерфейса JavaMailSender – класса org.springframework.mail.javamail.JavaMailSenderImpl.

По умолчанию используется бин с идентификатором «mailSender», настроенный на взаимодействие с почтовым сервером Gmail (smtp.gmail.com). Изменить это и другие настройки можно в конфигурационном файле приложения «applicationContext.xml».

Запрос

GET localhost/api/myapi

Приложение 2 [/demo]

Приложение 1 [/api]

Контекст развертывания

Контроллер

Сервисный слой

Доменный слой

Слой базы данных

Сервер приложений

[localhost]

Ответ

{

“message” : “Hello world!”

}

### Структурная схема взаимодействия с API

Рис 4. Сервер определяет приложение, которому необходимо направить запрос исходя из строки запроса, в то время как приложение, принимающее запрос проводит его через слои. Каждый слой может обратиться к слою под собой, или вернуть результат в случае ошибки во времени выполнения.

# ПРАКТИЧЕСКАЯ ЧАСТЬ

## Обоснование выбора языка программирования

При выборе языка программирования были учтены следующие факторы:

1. Статическая типизация – многие ошибки исключаются на стадии компиляции
2. Скорость
3. Кроссплатформенность – переносимость на другие платформы

Таким образом если для первого и второго пункта подходила платформа .NET с языком C# и Java со спецификацией Java EE (различные PHP и подобные были сразу же исключены), то для третьего пункта победителем стал язык Java.

Кроме того, на фоне иных малоизвестных языков программирования Java обладает огромным сообществом, что, несомненно, положительно влияет на скорость решения проблем при разработке приложений.