МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Информационные технологии»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

ДЛЯ ВЫПОЛНЕНИЯ КОНТРОЛЬНОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ «СОВРЕМЕННЫЕ ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ» ДЛЯ СТУДЕНТОВ ЗАОЧНОЙ И ОЧНО-ЗАОЧНОЙ ФОРМ ОБУЧЕНИЯ

(Семестр 2)

Ростов-на-Дону
ДГТУ
2023

УДК 372.8:004
Составители: М. В. Привалов

Методические указания для выполнения контрольной работы по дисциплине «Современные технологии программирования» для студентов заочной и очно-заочной форм обучения. (Семестр 2) - Ростов-на-Дону: Донской гос. техн. ун-т, 2023. - 11с.

Рассматриваются современные подходы и технологии разработки программного обеспечения для распределённых информационных систем на открытой платформе Java.

Предназначены для студентов направления 09.04.02 «Информационные технологии» заочной и очно-заочной форм обучения.

УДК 372.8:004

Печатается по решению редакционно-издательского совета
Донского государственного технического университета

Ответственный за выпуск зав. кафедрой «Информационные технологии»,
д-р техн. наук, профессор Б.В. Соболь

В печать ___.___. 20___г.
Формат 60×84/16. Объем___усл.п.л.
Тираж ___экз. Заказ №___.

Издательский центр ДГТУ
Адрес университета и полиграфического предприятия:
344000, г. Ростов-на-Дону, пл. Гагарина,1

Оглавление

Проектирование и создание Web-служб с использованием архитектурного паттерна REST 4

Основные теоретические сведения. 4

Способы реализации RESTful служб в Java. Структура проектов и управление зависимостями. 5

Порядок выполнения контрольной работы 9

Варианты индивидуальных заданий 10

ЛИТЕРАТУРА 11

Проектирование и создание Web-служб с использованием архитектурного паттерна REST

Цель работы: научиться реализовывать алгоритмы параллельной обработки данных

Основные теоретические сведения.

Основным недостатком SOAP XML служб является использование XML в качестве формата сообщений, передаваемых между клиентом и службой. Данный формат имеет очень высокие накладные расходы, связанные с наличием, помимо данных, открывающих и закрывающих тегов. Разбор этих данных требует заметного количества ресурсов процессора и объёмов ОЗУ. Службы WCF тоже не лишены данного недостатка: эта технология значительно более продвинута, по сравнению со старой ASP .NET, при этом она также может использовать для представления данных JSON помимо XML формата. Тем не менее, стандартные коммуникации опираются на тот же SOAP. Плюс, технология является проприетарной.

Альтернативой, которая очень популярна в настоящее время, является применение архитектурного паттерна REST. Он подразумевает применение стандартизированного кодирования URL ресурсов и применение JSON в качестве формата передачи данных.

Примеры URL:
/user/?id – запрос пользователя с идентификатором id
/user/list – запрос полного списка пользователей
/user/search – запрос списка пользователей по критерию
/user/save – сохранение пользователя (подразумевает данные в теле POST)

Реализовать такую службу на Java можно несколькими способами. Самые очевидные из них:

1. Реализация без использования сторонних фреймворков.
2. Реализация с использованием Jersey.

3. Реализация на основе Spring Framework.

Способы реализации RESTful служб в Java. Структура проектов и управление зависимостями.

1. Реализация RESTful службы стандартными средствами платформы Java

Предположим, что мы хотим реализовать службу, поддерживающую методы GET и POST, а также способную возвращать список пользователей по запросу /user/list

1. Создадим сервер и контекст. Для этого воспользуемся классом HttpServer.

```
HttpServer httpServer = HttpServer.create(new InetSocketAddress(PORT), 0);
httpServer.createContext("/user/list", httpExchange -> {
   // код обработки запроса
});
```

2. Проверим тип запроса. Для GET реализуем вывод списка пользователей, а в случае POST выбросим клиенту ошибку с кодом 405 (Method not allowed)

```
if ("GET".equals(httpExchange.getRequestMethod())) {
    // обработка GET
} else {
  httpExchange.sendResponseHeaders(NOT_ALLOWED, -1);
}
```

3. Для реализации выдачи списка пользователей требуется выполнить 3 действия:
    a. Задать заголовки (как минимум, указать Content-Type). Для этого надо получить заголовки методом getResponseHeaders() и задать требуемые, используя методы интерфейса Map. После этого надо выдать их с кодом возврата сервера (если нормально, то это код 200), используя метод sendResponseHeaders(200, 0).
    b. Получить представление списка пользователей в формате JSON (подразумевается, что у вас есть POJO класс, представляющий пользователя, допустим, User, а также список пользователей). Для этого можно использовать JSON simple, но лучше – библиотеку

Jackson.

c. JSON в виде массива символов записать в тело ответа. Тело ответа является потоком вывода, поэтому проблем с этим быть не должно: его нужно получить методом getResponseBody(), записать результат и закрыть.
d. Указать серверу пустой исполнитель с помощью setExecutor(null); и запустить сервер методом start().

Некоторые пояснения.
Чтобы подключить библиотеку Jackson, требуется в pom.xml добавить зависимость:

```
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.11.3</version>
</dependency>
```

Версия может отличаться.

Для конвертирования списка в JSON потребуется объект класса ObjectMapper из этой библиотеки.
В случае успешного запуска должен быть открыт сокет на порту PORT (см. пример создания сервера выше), а тестовый REST-клиент по запросу /user/list должен выдать фрагмент JSON (рис. 1).

Рис. 1. Результат тестирования службы

Неудобство данного метода: в случае наличия параметров службы, их придётся вручную разбирать, анализируя запрос, который можно получить посредством getRequestURI().getRawQuery().

2. Реализация RESTful службы с помощью проекта Jersey

Более продвинутый проект для реализации конечных точек REST – это проект Jersey. Его преимущества:

- отображение классов и методов на пути в URI
- отображение параметров запроса на аргументы методов класса
- управление путями, параметрами и рядом заголовков с помощью аннотаций

Подключение Jersey к проекту.

Для подключения к проекту нам понадобится добавить следующие зависимости в pom.xml:

<dependency>

```
    <groupId>org.glassfish.jersey.containers</groupId>
    <artifactId>jersey-container-servlet</artifactId>
    <version>2.33</version>
</dependency>
<dependency>
    <groupId>org.glassfish.jersey.inject</groupId>
    <artifactId>jersey-hk2</artifactId>
    <version>2.26</version>
</dependency>
<dependency>
    <groupId>org.glassfish.jersey.media</groupId>
    <artifactId>jersey-media-json-jackson</artifactId>
    <version>2.33</version>
</dependency>
```

Первая обеспечивает доступ к контейнеру, вторая – инъекцию зависимостей, третья – сериализацию и десериализацию данных в JSON.

Далее пишем код нашей RESTful службы. Простейший пример показан ниже:

```
@Path("/message")
public class JerseyService
{
   @GET
   public String getMsg()
   {
       return "Hello World !! - Jersey 2";
   }
}
```

Но чтобы служба заработала, она должна быть частью Web-приложения. При этом Jersey должен быть обработчиком URI, путь к которому определяет доступ к конечным точкам REST. Для этого требуется в web.xml добавить следующую конфигурацию:

```
<servlet>
    <servlet-name>jersey-serlvet</servlet-name>
    <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
    <init-param>
        <param-name>jersey.config.server.provider.packages</param-name>
```

```
            <param-value>com.evil.service</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>jersey-serlvet</servlet-name>
        <url-pattern>/rest/*</url-pattern>
    </servlet-mapping>
```

В этой конфигурации com.evil.service – это пакет Java, содержащий классы, предоставляющие конечные точки REST для взаимодействия. Путь /rest/ - это путь в контексте приложения к этим конечным точкам. Так, в нашем примере, путь к ресурсу в контексте приложения будет /rest/message/

Приложение развёртываем в Tomcat, после чего конечные точки становятся доступны.

Аннотации @Produces и @Consumes позволяют управлять тем, что выдаёт служба (обычный текст, JSON, XML и т.п.).

Для отображения параметров на аргументы используйте их имена в фигурных скобках:

```
@Path("/fetch/{id}")
public Response fetch(@PathParam("id") String id) {
}
```

Особенности возврата generic-коллекций.

Как правило, для нормальной разработки потребуется возвращать списки сущностей, то есть, пользоваться generic-коллекциями (List, Set, Map и т.п.). Но из-за особенностей Java при сериализации мы потеряем информацию о параметре типа и получим ошибку в Jersey. Чтобы этого не было, требуется воспользоваться обёрткой GenericEntity перед тем, как возвращать результат клиенту:

```
GenericEntity<List<User>> entity = new GenericEntity<List<User>>(users) {};
return Response.ok(entity).build();
```

Порядок выполнения контрольной работы

В соответствии с индивидуальным заданием требуется:

1. Определить функциональные границы Web-службы.
2. Определить контракт данных и функциональный контракт службы.

3. Создать REST-сервис с контрактом для обмена информацией сервера и клиента с применением обратной связи.
4. Продемонстрировать работу службы.

С помощью стандартных средств платформы или фреймворка Jersey реализовать вышеуказанный сервис и развернуть его в Web-контейнере. Обязательно предусмотреть в службе следующие 3 метода:

a) Возвращающий полный список сущностей (например, пользователей, клиентов, воздушных судов, заказов и т.п., согласно варивнту) по запросу GET
b) Возвращающий одну сущность по её Id, переданному запросом GET
c) Добавляющий одну сущность по запросу POST и возвращающий её Id

Продемонстрировать работу сервиса с помощью тестового REST-клиента.

Варианты индивидуальных заданий

1. Спроектировать компьютерную подсистему учета Интернет-услуг
2. Спроектировать подсистему учета ресурса воздушных судов
3. Спроектировать компьютерную подсистему начисления и учета страховых выплат клиентам в условиях страховой компании
4. Спроектировать подсистему учета и анализа экономической деятельности в условиях агентства недвижимости
5. Спроектировать подсистему формирования и обработки договоров добровольного страхования наземного автотранспорта
6. Спроектировать подсистему обработки торговых агентов
7. Спроектировать подсистему планирования и учета чартеров
8. Спроектировать подсистему формирования и учета счетов-заказов для туристических агентств
9. Спроектировать подсистему бронирования билетов в кинотеатр
10. Спроектировать подсистему расчета себестоимости для рыбного хозяйства
11. Спроектировать подсистему учета потребительских кредитов
12. Спроектировать подсистему реализации и движения горюче-смазочных материалов в условиях сети АЗС
13. Спроектировать подсистему учета продаж товаров
14. Спроектировать подсистему учета сбора и реализации зерновых культур
15. Спроектировать подсистему учета убытков автогражданской

ответственности

16. Спроектировать подсистему учета выплат за услуги газоснабжения для населения
17. Спроектировать подсистему расчета арендной платы в условиях коммунального предприятия
18. Спроектировать подсистему учета движения грузового подвижного железнодорожного транспорта в условиях металлургического завода
19. Спроектировать подсистему учета сырья и материалов
20. Спроектировать подсистему приема электронных коммунальных платежей в условиях банка
21. Спроектировать подсистему учета и планирования ремонтных работ
22. Спроектировать подсистему учета технического состояния компьютерного оборудования

ЛИТЕРАТУРА

1. Гуськова, О.И. Объектно-ориентированное программирование в Java : учебное пособие / О. И. Гуськова. - Москва : МПГУ, 2018. - 240 с. - ISBN 978-5-4263-0648-6. - Текст : электронный. - URL: https://znanium.com/catalog/product/1020593

2. Прохоренок, Н. А. Основы Java: Самоучитель Учебное пособие / Прохоренок Н.А. - СПб:БХВ-Петербург, 2017. - 704 с. ISBN 978-5-9775-3785-8. - Текст : электронный. - URL: https://znanium.com/catalog/product/978545

3. Будилов, В. А. Интернет-программирование на Java: Пособие / Будилов В.А. - СПб:БХВ-Петербург, 2014. - 698 с. ISBN 978-5-9775-1931-1. - Текст : электронный. - URL: https://znanium.com/catalog/product/940239

4. Создаем RESTful web service на Java с использованием Eclipse + Jersey + Glassfish3 /Хабр. – Текст: электронный. – URL: https://habr.com/ru/post/150176/

5. Lars Vogel, REST with Java (JAX-RS) using Jersey - Tutorial / Vogella GmbH. – Текст: электронный. – URL: https://www.vogella.com/tutorials/REST/article.html