



OTUS

ОНЛАЙН ОБРАЗОВАНИЕ

Онлайн-образование



Меня хорошо видно && слышно?

Ставьте +, если все хорошо

Напишите в чат, если есть проблемы

Проверить, идет ли запись!



The background of the slide is an aerial photograph of a dense city skyline, likely New York City, with numerous skyscrapers. The image is overlaid with a semi-transparent blue layer. On the left side of this layer, there is a network diagram consisting of white dots connected by thin white lines, forming a complex web. The title text is centered on this blue layer.

Contract testing with Spring

Свиридов Дмитрий

Правила вебинара



Активно участвуем



Задаем вопросы в чат или голосом



Off-topic обсуждаем в Slack



Вопросы вижу в чате, могу ответить не сразу

Цели вебинара

1

Вспомним
протоколы
взаимодействия
сервисов

2

Поговорим про REST

3

Что такое Contract

4

Попрактикуем
тестирование
контракта

5

Как работает Contract
testing в Spring



Протоколы

SOAP

SOAP (*Simple Object Access Protocol* — простой протокол доступа к объектам) — [протокол](#) обмена структурированными сообщениями

Чаще всего SOAP используется поверх HTTP.

Сообщение SOAP выглядит так:

- **Envelope** — корневой элемент, который определяет сообщение и пространство имен, использованное в документе.
- **Header** — содержит атрибуты сообщения, например: информация о безопасности или о сетевой маршрутизации.
- **Body** — содержит сообщение, которым обмениваются приложения.
- **Fault** — необязательный элемент, который предоставляет информацию об ошибках, которые произошли при обработке сообщений.

Пример SOAP конверта

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getProductDetailsResponse xmlns="http://warehouse.example.com/ws">
      <getProductDetailsResult>
        <productID>12345</productID>
        <productName>Стакан граненый</productName>
        <description>Стакан граненый. 250 мл.</description>
        <price>9.95</price>
        <currency>
          <code>840</code>
          <alpha3>USD</alpha3>
          <sign>$</sign>
          <name>US dollar</name>
          <accuracy>2</accuracy>
        </currency>
        <inStock>true</inStock>
      </getProductDetailsResult>
    </getProductDetailsResponse>
  </soap:Body>
</soap:Envelope>
```

Архитектура REST

Системы, поддерживающие REST, называются RESTful-системами.

Признаки:

1 – Использование протокола HTTP

2 – Взаимодействие Client - Server

3 – Использование методов

GET /book/ — получить список всех книг

GET /book/3/ — получить книгу номер 3

PUT /book/ — добавить книгу (данные в теле запроса)

POST /book/3 — изменить книгу (данные в теле запроса)

DELETE /book/3 — удалить книгу

4 – Передача сообщений в виде JSON

Преимущества REST

Преимущества, которые дает REST

- масштабируемость;
- прозрачность системы взаимодействия;
- простота интерфейсов;
- портативность компонентов;
- лёгкость внесения изменений;
- способность эволюционировать, приспосабливаясь к новым требованиям.

The image features a background of a dense city skyline, likely New York City, viewed from an elevated perspective. The entire image is tinted with a blue and teal color palette. A network of thin, light blue lines connects various points across the image, creating a digital or technological overlay. The word "Контракт" is centered in the middle of the image, overlaid on a semi-transparent dark blue horizontal band.

Контракт

Контракт для SOAP

WSDL (*Web Services Description Language*) — язык описания [веб-сервисов](#) и доступа к ним, основанный на языке [XML](#)

Каждый документ WSDL можно разбить на следующие логические части:

- 1.определение типов данных (types) — определение вида отправляемых и получаемых сервисом XML-сообщений
- 2.элементы данных (message) — сообщения, используемые web-сервисом
- 3.абстрактные операции (portType) — список операций, которые могут быть выполнены с сообщениями
- 4.связывание сервисов (binding) — способ, которым сообщение будет доставлено

XML Schema (XSD) — язык описания структуры [XML](#)-документа.

Пример WSDL

Пример WSDL:

```
<message name="getTermRequest">  
  <part name="term" type="xs:string"/>  
</message>
```

```
<message name="getTermResponse">  
  <part name="value" type="xs:string"/>  
</message>
```

```
<portType name="glossaryTerms">  
  <operation name="getTerm">  
    <input message="getTermRequest"/>  
    <output message="getTermResponse"/>  
  </operation>  
</portType>
```


Пример XML Schema

Простой пример схемы на XML Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="country">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="country_name" type="xs:string"/>
        <xs:element name="population" type="xs:decimal"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Пример документа, соответствующего этой схеме:

```
<?xml version="1.0" encoding="utf-8"?>
<country>
  <country_name>France</country_name>
  <population>59.7</population>
</country>
```

Контракт для REST

Методы GET, PUT, POST, DELETE

URL: host:8080/book/3

JSON Schema — один из языков описания структуры JSON-документа. Использует синтаксис JSON

JSON:

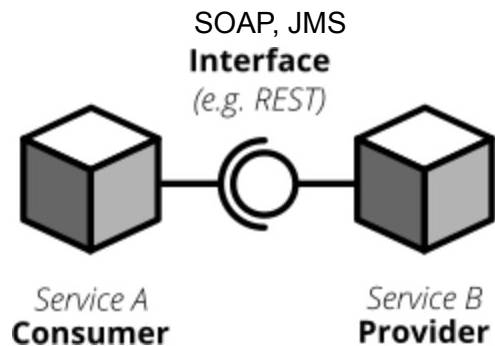
- **запись** — это неупорядоченное множество пар **ключ:значение**, заключённое в фигурные скобки «{ }»
- **массив** (одномерный) — это упорядоченное множество **значений**. Массив заключается в квадратные скобки «[]».
- **число** (целое или вещественное).
- **литералы** *true* ([логическое значение](#) «истина»), *false* ([логическое значение](#) «ложь») и *null*.
- **строка** — это упорядоченное множество из нуля или более символов [юникода](#), заключённое в двойные кавычки.

Пример JSON

```
{  
  "firstName": "Иван",  
  "lastName": "Иванов",  
  "address": {  
    "streetAddress": "Московское ш., 101, кв.101",  
    "city": "Ленинград",  
    "postalCode": 101101  
    "isAvailable": true  
  },  
  "phoneNumbers": [  
    "812 123-1234",  
    "916 123-4567"  
  ]  
}
```


Contract Tests

Проверяют, что реализации на стороне потребителя и поставщика всё ещё придерживаются определённого контракта. Они выступают хорошим набором регрессионных тестов и гарантируют раннее обнаружение отклонения от контракта.



Поставщик предоставляет данные потребителям. **Потребитель** обрабатывает данные, полученные от поставщика. Поставщик создаёт REST API со всеми необходимыми конечными точками, а потребитель обращается к этому REST API, чтобы получить данные или инициировать изменения в другой службе.

Spring Cloud Contract

Spring cloud contract – это файл, в котором на yaml или groovyDSL описано, как должны выглядеть запрос и ответ. По умолчанию все контракты лежат в папке `/src/test/resources/contracts/`.

Для примера протестируем простейший GET-endpoint

```
@GetMapping("/bets/{userId}")
public ResponseEntity<List<Bet>> getBets(@PathVariable("userId") String userId) {
    List<Bet> bets = service.getByUserId(userId);
    if (bets.isEmpty()) {
        return ResponseEntity.noContent().build();
    }
    return ResponseEntity.ok(bets);
}
```

Spring Cloud Contract

Описание контракта:

```
org.springframework.cloud.contract.spec.Contract.make {  
  request {  
    method 'GET'  
    urlPath '/bets/2'  
  }  
  response {  
    status 200  
    headers {  
      header('Content-Type', 'application/json')  
    }  
    body("""  
      {  
        "sport": "football",  
        "amount": 1  
      }  
      """)  
  }  
}
```

Далее из этого файла с помощью maven или gradle плагина генерируются юнит-тесты и json'ы для [wiremock](https://wiremock.org/).

*язык yaml

Spring Cloud Contract

```
package contracts.hello  
import org.springframework.cloud.contract.spec.Contract
```

```
Contract.make {  
    description "should return person by id=1"
```

```
    request {  
        url "/person/1"  
        method GET()  
    }
```

```
    response {  
        status OK()  
        headers {  
            contentType applicationJson()  
        }
```

```
        body (  
            id: 1,  
            name: "foo",  
            surname: "bee"  
        )  
    }  
}
```

```
*see my previous
```

Spring Cloud Contract

Ниже показан сгенерированный тест. По умолчанию использована библиотека RestAssured.

```
public class UserControllerTest extends ContractBase {

    @Test
    public void validate_get_200() throws Exception {
        // given:
        MockMvcRequestSpecification request = given();

        // when:
        ResponseOptions response = given().spec(request)
            .get("/bets/2");

        // then:
        assertThat(response.statusCode()).isEqualTo(200);
        assertThat(response.header("Content-Type")).isEqualTo("application/json");
        // and:
        DocumentContext parsedJson = JsonPath.parse(response.getBody().asString());
        assertThatJson(parsedJson).field("[\"amount\"]").isEqualTo(1);
        assertThatJson(parsedJson).field("[\"sport\"]").isEqualTo("football");
    }
}
```



LIVE



Загружаем шаблон проекта Spring

1. Скачать шаблонный spring проект на start.spring.io
2. Открыть проект в среде разработки.

Домашнее задание

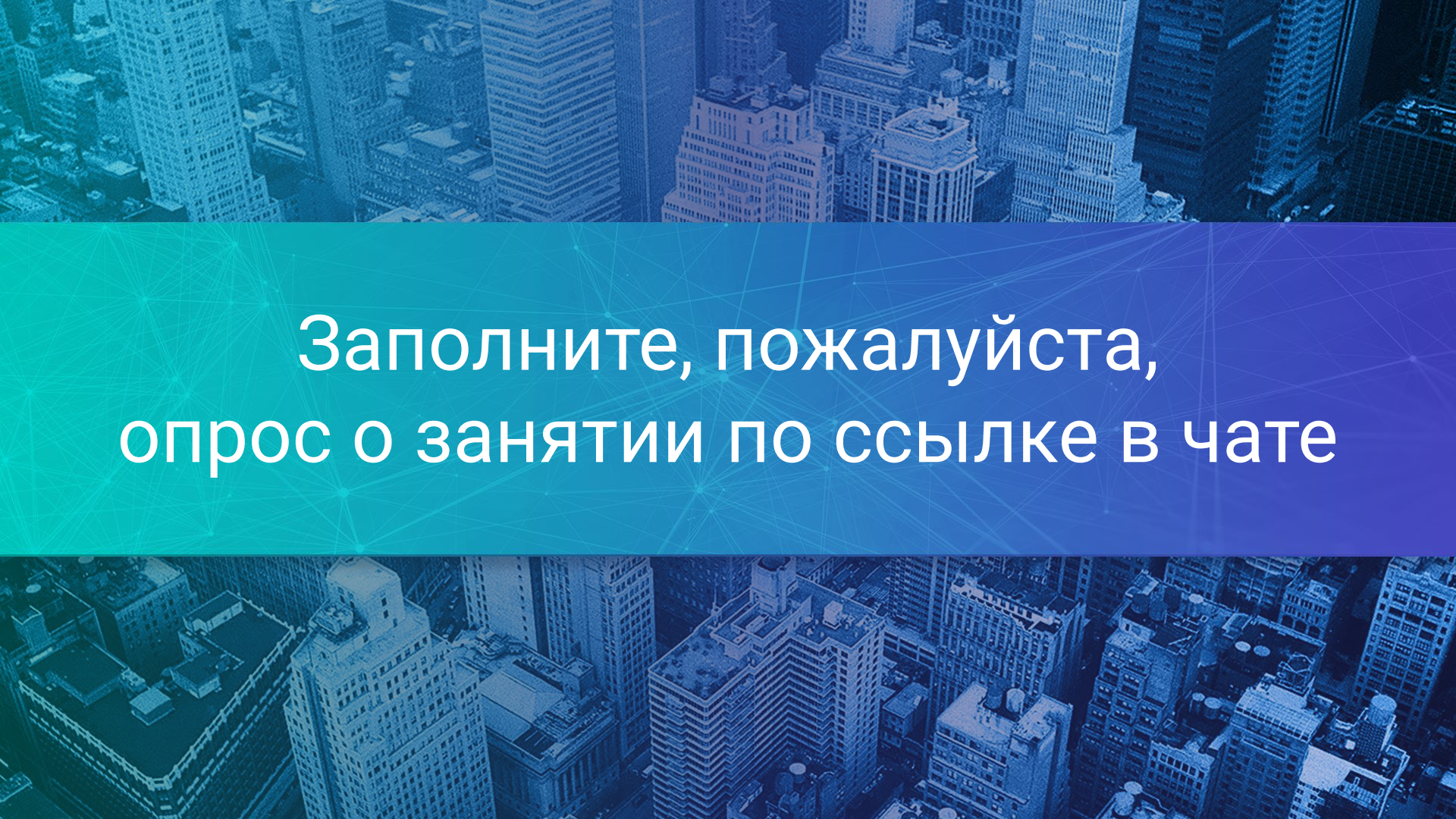
- 1 Протестировать 3-5 контрактов с сайта reqres.in.



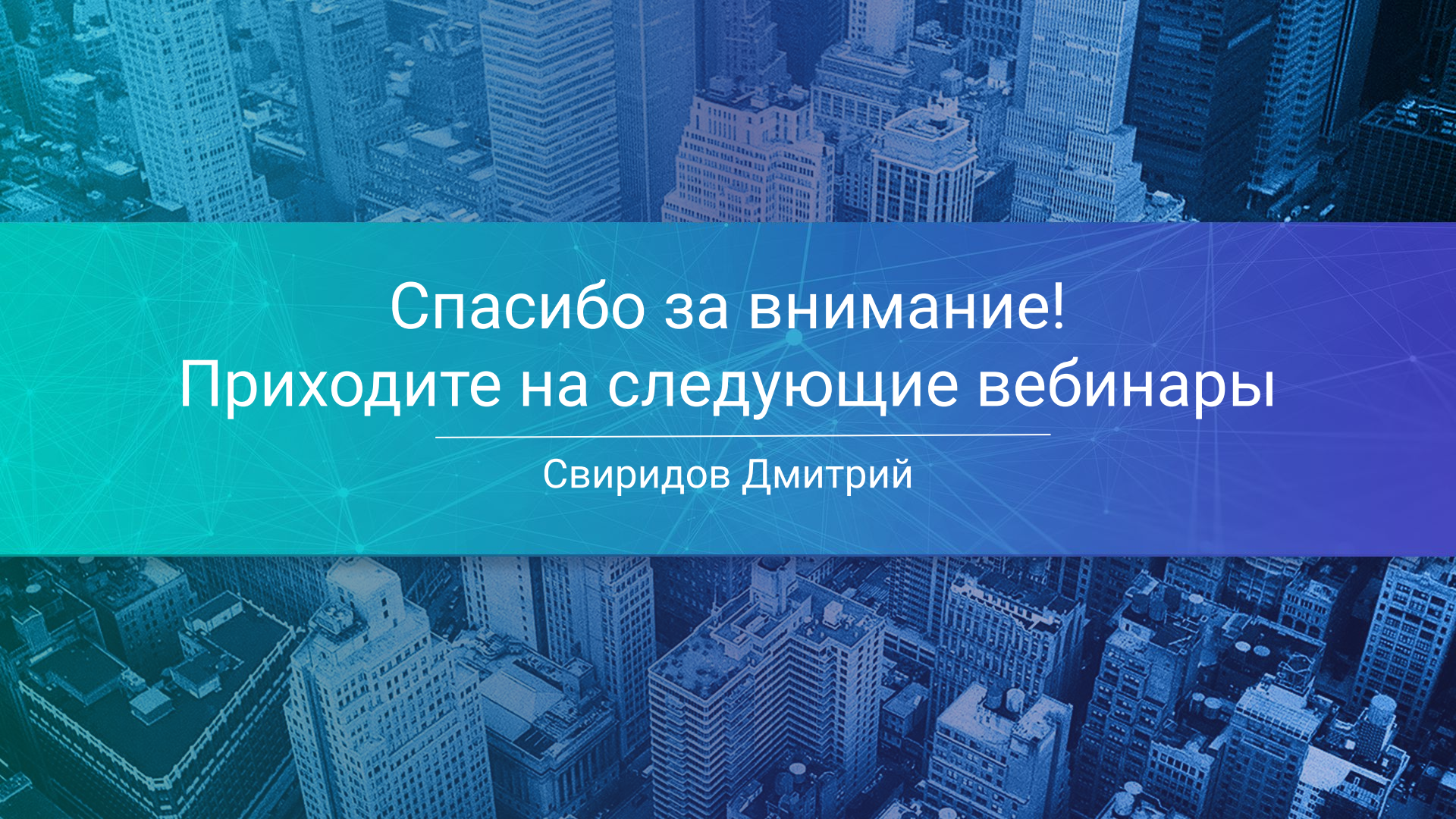
Срок:



Пройдите, пожалуйста, опрос в Чате с преподавателем после приёмки вашего ДЗ



Заполните, пожалуйста,
опрос о занятии по ссылке в чате



Спасибо за внимание!
Приходите на следующие вебинары

Свиридов Дмитрий