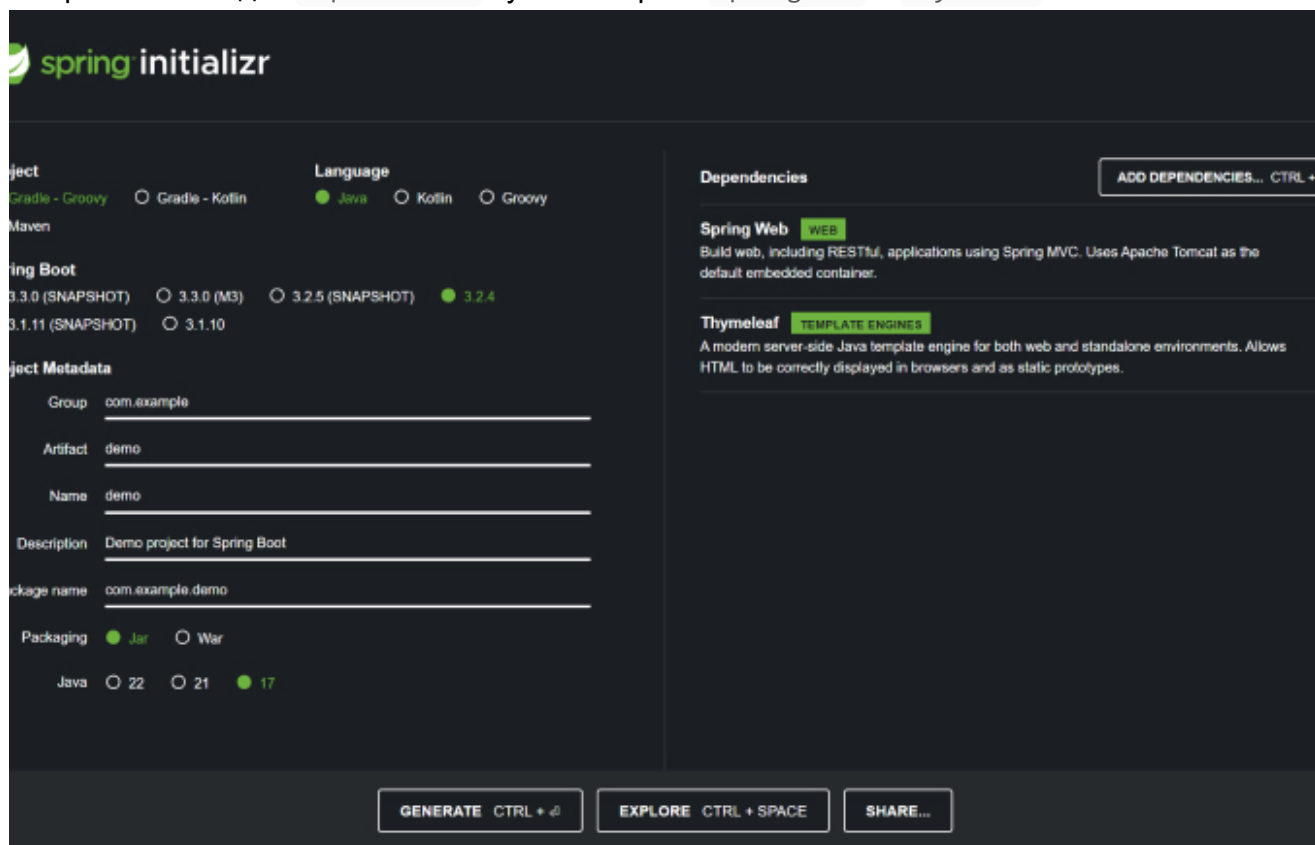


Spring-boot

Подготовка

Чтобы начать работу в java с html кодом и создавать сайты, нужно скачать пакет по [сайту](#), и выбрать во вкладке Dependencies нужно выбрать Spring Web и Thymeleaf

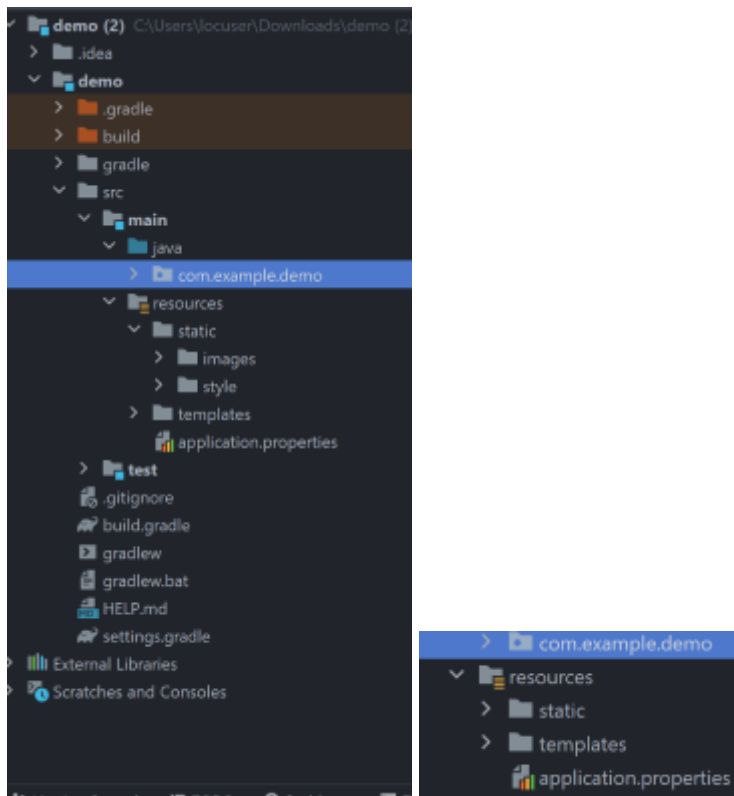


The screenshot shows the Spring Initializr web application interface. It features a dark theme with a green logo at the top left. The main area is divided into several sections: 'Project' with options for 'Gradle - Groovy', 'Gradle - Kotlin', and 'Language' (Java, Kotlin, Groovy); 'Spring Boot' with version selection (3.3.0, 3.3.0 (M3), 3.2.5 (SNAPSHOT), 3.2.4); 'Project Metadata' with fields for Group, Artifact, Name, Description, and Package name; and 'Dependencies' with a list of selected dependencies including 'Spring Web' and 'Thymeleaf'. At the bottom, there are three buttons: 'GENERATE', 'EXPLORE', and 'SHARE...'. The 'GENERATE' button has a tooltip that says 'CTRL + G'.

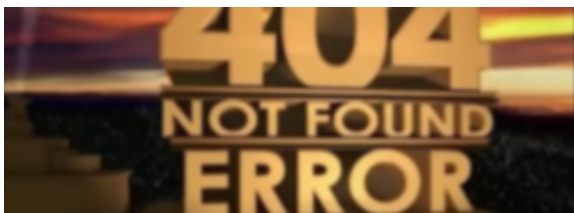
после мы нажимаем кнопку `generate`, распаковываем, и через IDEA находим нашу папку и открываем проект

Прежде чем начать писать код, нужно выстроить архитектуру нашего сервера, ведь от этого будут зависеть пути к нашим файлам.

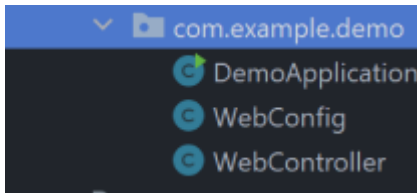
Наша архитектура довольно проста, просто создаем папки, в папке `resources` должны быть сделаны 2 папки `static` и `templates`, в папке `templates` будут находиться нашим html странички, а в папке `static` нужно сделать еще 2 папки `images` и `style`, в них будут находиться наши изображения и наши стили.



После создания архитектуры нужно создать пару файлов, один `WebConfig` и `WebController`, первый отвечает за то чтобы корректно отображались картинки на сайте и не вылетала



ошибка, а второй же будет отвечать за работу сайта, перехода по ссылкам и в принципе функционала.



WebConfig

Код `WebConfig`:

```
package com.example.demo;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
@EnableWebMvc
public class WebConfig implements WebMvcConfigurer {

    @Override
```

```

public void addResourceHandlers(ResourceHandlerRegistry registry) {
    registry.addResourceHandler("/images/**")
        .addResourceLocations("classpath:/static/images/");

    registry.addResourceHandler("/style/**")
        .addResourceLocations("classpath:/static/style/");

    registry.addResourceHandler("/scripts/**")
        .addResourceLocations("classpath:/static/scripts/");
}
}

```

В классе `addResourceHandlers` мы для `thymeleaf` (он отвечает за обработку вывода и корректного отображения наших картинок) "прокладываем" путь до наших файлов, чтобы не словить ошибку 404. Позже я расскажу зачем это необходимо.

WebController

Код `WebController`:

```

package com.example.demo;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class WebController {
    @GetMapping("/")
    public String First(){
        return "index";
    }
}

```

Что такое `@GetMapping("/")` это аннотация класса который мы помечаем, и если при запуске нашего проекта мы в строку браузера введем `http://localhost:8080/` то у нас откроется сделанный нами файл `index.html`

HTML

А теперь, что же находится в нашем `index`?

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.w3.org/1999/xhtml">
<head>
    <meta content="UTF-8">
    <title>попытка</title>
    <link rel="stylesheet" th:href="@{/style/style.css}">
</head>

```

```
<body>


</body>
</html>
```

Чтобы вывести картинку нужно написать код похожий на код из html

```

```

Как вы могли заметить, в классе `WebConfig` была в качестве параметров передана строка `"/images/**"` так вот, эти `**` отвечают за то что в примере `/images/logo.png` и эти звезды отвечают за файл и его расширение, и сам класс `WebConfig` хранит в себе более расширенные пути, и по факту облегчают нам работу как разработчикам

Что такое `alt` и зачем оно нужно?

Это обозначение сработает если у нас картинка не выведется и в этом случае выведется сообщение

Что такое `width` ?

`Width` отвечает за сжатие картинки например `width="100%"` это отображение картинки в натуральную величину, без сжатия, в примере у меня картинка выводится в 30% от натуральной величины.

А как сделать фоновую картинку для сайта?

Необходимо в файле `style` (если вы его конечно сделали в пути `/static/style/style.css`) прописать некоторые строки которые будут относиться к `body` **МНОГО BODY В ПРОЕКТЕ ЭТО БОЛЬ** поэтому оно должно быть одно.

```
body {
    background-image: url('../images/background.jpg');
    background-repeat: no-repeat;
    background-size: cover;
}
```

Что такое `background-image` ?

Здесь мы прописываем путь до нашей картинки, почему не так как мы делали это раньше? Потому что это `css` и он сам прекрасно видит все изображения, поэтому ему не нужен `themeleaf`

Что такое `background-repeat` ?

При помощи этой команды у нас изображение на фоне всегда будет одно и оно не будет двигаться при скроле.

Что такое `background-size` ?

При помощи этой команды мы делаем так чтобы наша картинка покрывала весь экран
ВОЗМОЖНО ЧТО ПОСТРАДАЕТ САМО ИЗОБРАЖЕНИЕ