1. Spring Boot 简介

- **Spring Boot Starter：它将常用的依赖分组进行整合，将其合并到一个以来中，这样就可以一次性添加道项目的Maven或Gradle构件中**
- **自动配置：Spring Boot的自动配置特性利用了Spring对条件话配置的支持，和力地推测了应用所需bean并自动化配置他们**
- **命令接口(command-line interface，CLI)：Spring Boot的CLI发挥了Groovy编程语言的优势，并结合自动配置进一步简化Spring应用的开发**
- **Actuator：它为Spring Boot应用添加了一定的管理特性**

2. 构建Spring Boot构建应用

2.1 处理请求

```
@Controller
@RequestMapping("/")
public class ContractController {
    private ContractRepository contractRepository;

    @Autowired
    public ContractController(ContractRepository contractRepository) {
        this.contractRepository = contractRepository;
    }

    @RequestMapping(method = RequestMethod.GET)
    public String home(Map<String, Object> model) {
        List<Contact> contacts = contractRepository.findAll();
        model.put("contacts", contacts);

        return "home";
    }

    @RequestMapping(method = RequestMethod.POST)
    public String submit(Contact contact) {
        contractRepository.save(contact);
        return "redirect:/";
    }
}
```

ContractController使用了@Controller注解，所以组件扫描会找到他。我们不需要在Spring应用上下文中明确声明为bean

```
@Data
public class Contact {
    private long id;
    private String firstName;
    private String lastName;
    private String phoneNum;
    private String emailAddress;
}
```

2.2 创建视图

使用Thymeleaf来定义视图

只要将Thymeleaf添加道项目的类路径下，就启用了Spring Boot的自动配置。当应员工运行时，Spring Boot将会探测到类路径中的Thymeleaf，然后会自动配置视图解析器、模板解析器和模板引擎，这些都是在Spring MVC使用Thymeleaf所需要的。因此，在应用中不需要是哟共显式Spring配置的方式来定义Thymeleaf

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Spring Boot Contacts</title>
    <link rel="stylesheet" th:href="@{/style.css}"/>
</head>

<body>
<h2>Spring Boot Contacts</h2>
<form method="POST">
    <label for="firstName">First Name:</label>
    <input type="text" name="firstName"></input><br/>
    <label for="lastName">Last Name:</label>
    <input type="text" name="lastName"></input><br/>
    <label for="phoneNumber">Phone #:</label>
    <input type="text" name="phoneNumber"></input><br/>
    <label for="emailAddress">Email:</label>
    <input type="text" name="emailAddress"></input><br/>
    <input type="submit"></input>
</form>

<ul th:each="contact : ${contacts}">
    <li>
        <span th:text="${contact.firstName}">First</span>
        <span th:text="${contact.lastName}">Last</span> :
        <span th:text="${contact.phoneNumber}">phoneNumber</span>,
        <span th:text="${contact.emailAddress}">emailAddress</span>
    </li>
</ul>
</body>
</html>
```

自动配置的模板解析器会爱指定的目录下查找Thymeleaf模板，这个项目就是是相对于根类路径下的templates目录下，因此需要将home.html放到“src/main/resources/templates”

这个模板还需要应员工名为style.css的样式表，因此需要将这个样式表放到项目中

```
body {
    background-color: #eeeeee;
    font-family: sans-serif;
}

label {
    display: inline-block;
    width: 120px;
    text-align: right;
}
```

2.3 添加静态内容

当采用Spring Boot的Web自动配置来定义Spring MVC bean时，这些bean会包含一个资源处理器，它会将“/**”映射到几个资源路径中

- /META-INF/resources/
- /resources/
- /static/
- /public/

在传统的基于Maven/Gradle构建项目中，通常会将静态呢日荣放在“src/main/webapp”目录中，这样在构建所生成war文件里面，这些内容就会位于war文件的根目录下。如果使用Spring Boot构建war文件的化，这依然是可选的方案。但是也可以将静态内容放在资源处理器所映射的上述四个路径下。

2.4 持久化数据

使用H数据库和JDBC(使用Spring 的 JdbcTemplate)

JDBC Starter一来回将 Spring JdbcTemplate需要的所有内容都引入进来

```
@Repository
public class ContactRepository {

    private JdbcTemplate jdbc;

    @Autowired
    public ContactRepository(JdbcTemplate jdbc) {
        this.jdbc = jdbc;
    }

    public List<Contact> findAll() {
        return jdbc.query(
                "select id, firstName, lastName, phoneNumber, emailAddress " +
                        "from contacts order by lastName",
                new RowMapper<Contact>() {
                    @Override
                    public Contact mapRow(ResultSet rs, int rowNum) throws
SQLException {
                        Contact contact = new Contact();
                        contact.setId(rs.getLong(1));
                        contact.setFirstName(rs.getString(2));
                        contact.setLastName(rs.getString(3));
                        contact.setPhoneNumber(rs.getString(4));
                        contact.setEmailAddress(rs.getString(5));
                        return contact;
                    }
                }
        );
    }

    public void save(Contact contact) {
        jdbc.update(
                "insert into contacts " +
                        "(firstName, lastName, phoneNumber, emailAddress) " +
                        "values (?, ?, ?, ?)",
                contact.getFirstName(), contact.getLastName(),
                contact.getPhoneNumber(), contact.getEmailAddress());
```

```
    }

}
```

ContactRepository使用了@Repository，因此组件扫描的时候，它会被发现并创建为Spring应员工上下文中共的bean

当Spring Boot 探测到Spring的JDBC模块和H2在路径下的时候，自动配置就会发挥作用，将自动配置JdbcTemplate bean和 H2DataSource bean

数据库模型

```
create table contacts
(
    id identity,
    firstName    varchar(30) not null,
    lastName     varchar(50) not null,
    phoneNumber  varchar(13),
    emailAddress varchar(30)
);
```

如果我们将这个文件命名为schema.sql并将其放在根路径下(src/main/resources),当启动的时候就会找打这个文件并进行数据加载。

如果需要构建部署war文件 需要在依赖中添加：

```
<packaging>war</packaging><!--打包成war 默认打包为jar-->
```

然后就可以部署到任意支持SErvlet的容器中

3 组合使用Groovy与Spring Boot CLI

通过使用Groovy语言，配合Spring框架，开发应用会更加简洁

page 524

4 通过Actuator获取了解应员工内部状况

Spring Boot Actuator所完成的主要功能就是为基于Spring Boot的应用添加多个有用的管理端点。包含以下内容

- **GET /autocofig：描述了Spring Boot在使用自动配置的时候，所作出的决策**
- **GET /beans：列出运行应用所配置的bean**
- **GET /configprops：列出应用中讷讷感狗用来配置bean的所有属性以及当前的值**
- **GET /dump：列出应用的线程，包括每个线程的栈跟踪信息**
- **GET /env：列出应用上下文中所有可用的环境变量和系统属性变量**
- **GET /env/{name}：展现某个特定环境变量和属性变量的值**
- **GET /health：展现当前应用的健康状况**
- **GET /metrics：列出应用相关的指标，包括请求特定端点的运行次数**
- **GET /metics/{name}：展现应用特定指标项的指标状况**
- **GET /shutdown：强制关闭应用**
- **GET /trace：列出应用最近请求相关的元数据，包括请求和响应头**