1. Java Web 应用开发基础知识

1. 使用浏览器访问计算机工程学院的主页，服务器发送的 HTTP 响应报文中的 Server 字段的值是\_\_\_\_\_\_（1）Tomcat:Apache-Coyote\_\_\_\_\_\_\_\_\_。

2. 将浏览器的语言设置成英文以后，访问江苏海洋大学的官方网页https://www.jou.edu.cn，响应报文中的Content-Language字段的值是\_\_\_\_（2）zh-cn\_\_\_\_。因此江苏海洋大学的官网没有英文版主页。

3. 浏览https://lib.jou.edu.cn/，使用页面上方的搜索框搜索关键字CSCS，表单的提交方法是\_\_\_（3）POST\_\_\_\_\_(字母使用大写)，HTTP响应的状态码是\_\_\_\_\_（4）200\_\_\_\_\_。

4. 本课程学习使用的构建系统是Gradle，运行时默认使用的构建脚本文件名是 \_\_\_\_\_build.gradle\_\_\_\_\_\_。

5. 包含十字图标生成的 Java Web 应用程序使用 Servlet 技术实现, Context Path 为 problem1，开发服务器使用地址 http://localhost:8080/，该 Servlet 路由地址是 crosslogo，并且支持查询参数（n 十字图标的大小，如不指定，默认大小为 3)和 fillcolor（格点颜色，使用#RRGGBB 的格式，也就是以#字符开始的 16 进制颜色值(字母大写），默认颜色值#FF0000）。则生成大小为 30的红色十字图标的最短查询字符串是 \_\_(6) n=30&fillcolor=#FF000\_\_（仅填写?字符后面的内容），如果使用网址 http://localhost:8080/problem/crosslog?fillcolor=#FF0000&fillcolor=#00FF00&N=5 访问，则生成的十字图标的大小是\_\_\_\_（7）5\_\_\_\_\_。

6. Spring 框架的核心功能是 DI 和 AOP，Spring 框架的 Autowiring 功能是\_\_\_（8）DI\_\_\_\_（填首字母缩写语）的一种常用形式。

7. Spring Boot 2.7.11 中添加依赖 JDBC 之后，HikariCP 的版本是是\_\_\_\_（9）2.5.1\_\_\_\_\_\_(填.分隔的 3 个数字)。

8. 在 Spring Initializr 中选择依赖 H2 Database 则在生成的构建代码中添加的依赖坐标是\_\_\_\_\_\_\_\_\_\_\_\_\_\_（10）\_\_\_\_\_\_\_\_\_\_\_com.h2database:h2\_\_\_\_\_\_\_\_\_\_（格式为 groupId:artifactId)。

1. Servlet编程

使用 Servlet 验证余数排列定理，根据最佳的编程实践，业务逻辑与显示分离的原则，使用 Thymeleaf 模板引擎显示验证的表格。核心代码如下：

TemplateEngine buildTemplateEngine() {

final ClassLoaderTemplateResolver templateResolver =

new ClassLoaderTemplateResolver();

templateResolver.setTemplateMode(TemplateMode.HTML);

templateResolver.\_\_\_\_\_\_(1) setPrefix\_\_\_\_\_\_\_\_\_\_("/templates/");

templateResolver.setSuffix("\_\_\_\_(2) .html\_\_\_\_\_");

final TemplateEngine templateEngine = new TemplateEngine();

templateEngine.setTemplateResolver(templateResolver);

return templateEngine;

}

private void renderTableModulo10Template(TemplateEngine engine, PrintWriter pw) throws IOException {

Context ctx = new Context();

int[] nums = IntStream.range(1, 10).toArray();

Integer[] invs = new Integer[nums.length];

ctx.setVariable("\_\_\_\_\_(3) nums\_\_\_\_\_\_", nums);

int[] xy = new int[2];

for (int i = 0; i < nums.length; i++) {

if (exgcd(nums[i], 10, xy) == 1) {

\_\_\_\_(4)invs\_\_\_\_\_[i] = (xy[0] + 10) % 10;

}

}

ctx.setVariable("invs", invs);

engine.process("\_\_\_\_\_\_\_(4) table\_modulo\_10\_\_\_\_\_\_\_\_\_", ctx, pw);

}

<!-- resources/templates/table\_modulo\_10.tmpl -->

<!DOCTYPE html>

<html xmlns:\_\_\_\_(5)th\_\_\_\_\_="http://www.thymeleaf.org">

<head>

<meta charset="UTF-8">

<title>Table - Modulo 10</title>

<style>

td.border {

border: 1px solid black;

}

</style>

</head>

<body>

<table>

<tr>

<td>&times;</td>

<td \_\_\_(6)th:each\_\_\_\_\_\_="a : ${nums}" th:text="\_\_\_(7) ${a}\_\_\_\_\_\_"></td>

<td></td>

</tr>

<tr th:each="i : ${nums}">

<td th:text="${i}"></td>

<td class="\_\_\_(8)border\_" th:each="j : ${nums}" \_\_(9)th:text\_="${i \* j % 10}"></td>

<td th:text="${\_\_\_\_\_(10)invs\_\_\_\_\_\_[i - 1]}"></td>

</tr>

</table>

</body>

</html>

1. Spring 框架基础

使用 Spring 编写程序输出 main 方法中注释的输出结果。

// 所有代码均位于beans包中

@Component

public class Parrot {

private String name = "JuJu";

// Omitted getters and setters

@Override

public String toString() {

return "\_\_\_(1) Parrot:\_\_\_" + name;

}

}

@\_\_\_\_\_\_(2) Component\_\_\_\_\_\_\_\_

public class Person {

private String name = "Ella";

private final Parrot parrot;

public Person(@Qualifier("\_\_\_\_\_(3) parrotMiki\_\_\_\_\_\_") Parrot parrot) {

this.parrot = parrot;

}

// Omitted getters and setters

}

@\_\_\_\_\_\_(4) Configuration\_\_\_\_\_\_\_\_

@ComponentScan(\_\_\_\_\_\_(5) basePackages\_\_\_\_\_\_\_\_\_\_ = "beans")

public class ProjectConfig {

\_\_\_(6) @Bean("parrotKoko")\_\_\_\_\_\_\_\_

public Parrot parrotKoko() {

Parrot p = new Parrot();

p.setName("Koko");

return p;

}

@Bean("parrotMiki")

public Parrot parrot2() {

Parrot p = new Parrot();

p.setName("Miki");

return p;

}

}

public class Main {

public static void main(String[] args) {

var context = new

AnnotationConfigApplicationContext(ProjectConfig.\_\_\_\_(7) class\_\_\_\_\_);

Person p = context.\_\_\_\_\_\_(8) getBean\_\_\_\_\_\_\_(Person.class);

// 输出结果：

// Person's name: Ella

// Person's parrot: Parrot:Miki

// Parrot:JuJu

// Parrot:Koko

System.out.println("Person's name: " + p.getName());

System.out.println("Person's parrot: " + p.getParrot());

System.out.println(context.getBean(\_\_\_\_\_\_\_\_(9) parrotMiki\_\_\_\_\_\_\_));

System.out.println(context.getBean(\_\_\_\_\_\_\_\_(10) parrotKoko\_\_\_\_\_\_\_));

}

}

1. Spring 应用架构

使用 Spring Boot 和 Spring MVC 开发 Web 应用程序，实现用户的登录、退出及所有用户的登录次数功能。(使用Spring Boot默认配置)

<!-- index.html -->

<!DOCTYPE html>

<html lang="en" xmlns:th="http://www.thymeleaf.org">

<head>

<meta charset="UTF-8">

<title>Login</title>

</head>

<body>

<form action="/" method="\_\_\_\_\_(1) POST\_\_\_\_\_\_\_">

Username: <input type="text" name="username" /><br />

Password: <input type="password" name="pwd" /><br />

<button type="submit">Log in</button>

</form>

<p th:text="${message}"></p>

</body>

</html>

<!-- main.html -->

<!DOCTYPE html>

<html lang="en" xmlns:th="http://www.thymeleaf.org">

<head>

<meta charset="UTF-8">

<title>Login</title>

</head>

<body>

<h1>Welcome, <span th:text="${username}"></span>!</h1>

<h2>Your login number is <span th:text="${loginCount}"></span></h2>

<a href="/main?logout">Log out</a>

</body>

</html>

@Controller

public class LoginController {

private final LoginProcessor loginProcessor;

public LoginController(LoginProcessor loginProcessor) {

this.loginProcessor = loginProcessor;

}

@GetMapping("/")

public String loginGet() {

return "login.html";

}

@\_\_\_\_\_\_\_（2）​PostMapping\_\_\_\_\_\_\_\_\_\_("/")

public String loginPost(

@\_\_\_\_（3）RequestParam\_\_\_\_\_\_String username,

@RequestParam(\_\_\_(4) "password"\_\_\_\_\_\_) String password,

Model model

) {

loginProcessor.setUsername(username);

loginProcessor.setPassword(password);

boolean loggedIn = loginProcessor.login();

if (loggedIn) {

return "redirect:/main";

}

model.addAttribute("\_\_\_\_\_(5) message\_\_\_\_\_\_\_\_", "Login failed!");

return "login.html";

}

}

@Controller

public class MainController {

private final LoggedUserManagementService loggedUserManagementService;

private final LoginCountService loginCountService;

public MainController(LoggedUserManagementService loggedUserManagementService, LoginCountService loginCountService) {

this.loggedUserManagementService = loggedUserManagementService;

this.loginCountService = loginCountService;

}

@GetMapping("/main")

public String home(

@RequestParam(required = false) String logout,

Model model

) {

if (logout != null) {

loggedUserManagementService.setUsername(\_\_\_\_\_(6) ​null\_\_\_\_\_\_);

}

String username = loggedUserManagementService.getUsername();

int count = loginCountService.getCount();

if (username == null) {

return "\_\_\_\_\_\_\_\_(7) redirect:/\_\_\_\_\_\_\_";

}

model.addAttribute("username" , username);

model.addAttribute("loginCount", count);

return "main.html";

}

}

@Component

@\_\_\_\_\_\_\_(8) RequestScope\_\_\_\_\_\_\_\_\_

public class LoginProcessor {

private final LoggedUserManagementService loggedUserManagementService;

private final LoginCountService loginCountService;

private String username;

private String password;

public LoginProcessor(

LoggedUserManagementService loggedUserManagementService,

LoginCountService loginCountService) {

this.loggedUserManagementService = loggedUserManagementService;

this.loginCountService = loginCountService;

}

public boolean login() {

loginCountService.increment();

String username = this.getUsername();

String password = this.getPassword();

boolean loginResult = false;

if ("natalie".equals(username) && "password".equals(password)) {

loginResult = true;

loggedUserManagementService.setUsername(username);

}

return loginResult;

}

//省略的getters和setters

}

@Service

@\_\_\_\_\_\_(9) SessionScope\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

public class LoggedUserManagementService {

private String username;

//省略的getters和setters

}

@Service

@\_\_\_\_\_(10) ApplicationScope\_\_\_\_\_\_\_\_\_

public class LoginCountService {

private int count;

public void increment() {

count++;

}

public int getCount() {

return count;

}

}

1. Spring 应用测试

使用 Spring 集成测试功能对购物 REST API 进行测试。

@ RestController

@RequestMapping("/purchase")

public class PurchaseController {

private final PurchaseRepository purchaseRepository;

public PurchaseController(PurchaseRepository purchaseRepository) {

this.purchaseRepository = purchaseRepository;

}

@PostMapping

public void storePurchase(\_\_\_\_\_\_\_\_(1)@RequestBody\_\_\_\_\_\_\_\_\_ Purchase purchase) {

purchaseRepository.storePurchase(purchase);

}

@GetMapping

public \_\_\_\_\_\_\_(2)List<Purchase>\_\_\_\_\_\_ findPurchases() {

return purchaseRepository.findAllPurchases();

}

}

@\_\_\_\_\_\_\_\_(3)WebMvcTest\_\_\_\_\_\_\_\_\_

@AutoConfigureMockMvc

class MainTests {

@Autowired

private MockMvc mockMvc;

@\_\_\_\_\_(4) MockBean\_\_\_\_\_\_

private PurchaseRepository purchaseRepository;

@Test

void storePurchaseTest() throws Exception {

ObjectMapper mapper = new ObjectMapper();

Purchase p = new Purchase();

p.setProduct("Spring Quickly");

p.setPrice(BigDecimal.TEN);

mockMvc.\_\_\_\_\_(5) perform\_\_\_\_\_\_\_(post("/purchase")

.contentType(MediaType.\_\_\_\_\_\_(6) application/json\_\_\_\_\_\_\_\_\_\_\_)

.content(mapper.writeValueAsString(p))

).\_\_\_\_\_\_\_(7) andExpect\_\_\_\_\_\_\_\_\_(status().isOk());

\_\_\_\_\_\_\_\_\_(8) verify\_\_\_\_\_\_\_\_\_\_\_(purchaseRepository).storePurchase(p);

}

@Test

void getPurchases() throws Exception {

ObjectMapper mapper = new ObjectMapper();

Purchase p = new Purchase();

p.setProduct("Spring Quickly");

p.setPrice(BigDecimal.TEN);

List<Purchase> purchases = List.of(p);

when(purchaseRepository.findAllPurchases()).\_\_\_\_\_(9) thenReturn\_\_\_\_\_\_\_\_(purchases);

mockMvc.perform(get("/purchase"))

.andExpect(status().isOk())

.andExpect(content().\_\_\_\_\_\_(10) json\_\_\_\_\_\_\_\_(mapper.writeValueAsString(purchases)));

}

}