CS544 EA

Integration

REST: Spring RestTemplate

RestTemplate

- Spring provides RestTemplate for REST clients

Method	Description
`.getForObject()`	Sends a GET, returns a Java Object based on response body
`.getForEntity()`	Sends a GET, returns ResponseEntity (status/header/body)
`.postForLocation()`	Sends a POST, returns a URI (Location header from redirect)
`.postForObject()`	Sends a POST, returns a Java Object based on response body
`.postForEntity()`	Sends a POST, returns a ResponseEntity (status/header/body)
`.put()`	Creates or updates a resource using PUT
`.patchForObject()`	Sends a PATCH, returns Java Object
`.delete()`	Deletes the resource at the URI using DELETE
`.exchange()`	General / flexible method using RequestEntity / ReponseEntity

Needs extra HTTP lib to work

Lowerlevel .execute() method is also present, which we won't cover

Console Client Config

```java
@SpringBootApplication
public class ConsoleConfig {
	@Bean
	public RestTemplate getRestTemplate() {
		return new RestTemplate();
	}

	public static void main(String[] args) {
		SpringApplication.run(ConsoleConfig.class, args);
	}
}
```

Create the RestTemplate bean

Spring Boot executes all beans that implement CommandLineRunner

```java
@Component
public class ConsoleApp implements CommandLineRunner {
	@Autowired
	private PersonService personService;

	@Override
	public void run(String... args) {
		Person p = personService.get(1L);
		personService.add(new Person("Hello", 22));
		System.out.println(personService.getAll());
		p.setAge(33);
		personService.update(p);
		System.out.println(personService.getAll());
		personService.delete(2L);
		System.out.println(personService.getAll());
		p = personService.getAll().get(0);
		System.out.println(p.getName());
	}
}
```

pom.xml dependencies

```xml
<dependency>
	<groupId>org.springframework.boot</groupId>
	<artifactId>spring-boot-starter</artifactId>
</dependency>
<dependency>
	<groupId>org.springframework</groupId>
	<artifactId>spring-web</artifactId>
</dependency>
<dependency>
	<groupId>com.fasterxml.jackson.core</groupId>
	<artifactId>jackson-databind</artifactId>
</dependency>
```

application.properties

```
spring.main.web-application-type=NONE
```

Example

```java
@Service
public class PersonServiceProxy implements PersonService {
	@Autowired
	private RestTemplate restTemplate;
	private final String personUrl = "http://localhost:8080/person/{id}";
	private final String pplUrl = "http://localhost:8080/person/";

	@Override
	public Person get(Long id) {
		return restTemplate.getForObject(personUrl, Person.class, id);
	}
	@Override
	public List<Person> getAll() {
		ResponseEntity<List<Person>> response =
				restTemplate.exchange(pplUrl, HttpMethod.GET, null,
					new ParameterizedTypeReference<List<Person>>() {});
		return response.getBody();
	}
	@Override
	public Long add(Person p) {
		URI uri = restTemplate.postForLocation(pplUrl, p);
		if (uri == null) { return null; }
		Matcher m = Pattern.compile(".*/person/(\\d+)").matcher(uri.getPath());
		m.matches();
		return Long.parseLong(m.group(1));
	}
	@Override
	public void update(Person p) {
		restTemplate.put(personUrl, p, p.getId());
	}
	@Override
	public void delete(Long id) {
		restTemplate.delete(personUrl, id);
	}
}
```

Same PersonService interface

.getForObject(url, return type, ...path params)

Have to use .exchange() because .getForObject() cannot specify List<Person> as return type! (only List)

.exchange(url, method, reqData, respType)

.postForLocation(url, reqData)

.put(url, reqData, ...path params)

.delete(url, ...path params)