CS544 EA

Integration

REST: Spring MVC

Spring MVC

- Spring MVC built-in support for RESTful web services

```
<dependency>
     <groupId>com.fasterxml.jackson.core</groupId>
     <artifactId>jackson-databind</artifactId>
</dependency>
```

- When the Jackson library is on the classpath
  - Automatically configures the bean: `MappingJackson2HTTPMessageConverter`
    - Able to convert Java Objects to JSON text
    - Able to convert JSON text to Java Objects

Controller

@RestController is a composite of @Controller and @ResponseBody

```
@RestController
public class PersonController {
	@Autowired
	private PersonService personService;

	@GetMapping(value="/person/", produces="application/json")
	public List<Person> getAll() {
		return personService.getAll();
	}

	@GetMapping(value= "/person/{id}", produces = "application/json")
	public Person get(@PathVariable long id) {
		return personService.get(id);
	}

	@PostMapping(value="/person/", consumes = "application/json")
	public RedirectView post(@RequestBody Person person) {
		long id = personService.add(person);
		return new RedirectView("/person/" + id);
	}

	@PutMapping(value= "/person/{id}", consumes = "application/json")
	public void put(@PathVariable long id, @RequestBody Person person) {
		if (id != person.getId()) { throw new IllegalArgumentException(); }
		personService.update(person);
	}

	@DeleteMapping("/person/{id}")
	public void delete(@PathVariable long id) {
		personService.delete(id);
	}
}
```

Produces / Consumes optional

ResponseBody becomes like:
{“id”: 1, “name”: “Test”, “age”: 28}

RedirectView “redirect: ” String won’t work!

RequestBody expected like:
{“name”: “Other”, “age”: 27}

Person & Person Service

```java
public class Person {
	private Long id;
	private String name;
	private int age;

	...
}
```

```java
public interface PersonService {
	Person get(Long id);
	List<Person> getAll();
	Long add(Person p);
	void update(Person p);
	void delete(Long id);
}
```

```java
@Service
public class MockPersonService implements PersonService {
	private Map<Long, Person> ppl = new HashMap<>();
	private long next = 1;

	public MockPersonService() {
		add(new Person("Test", 28));
	}
	@Override
	public Person get(Long id) {
		return ppl.get(id);
	}
	@Override
	public List<Person> getAll() {
		return new ArrayList<Person>(ppl.values());
	}
	@Override
	public Long add(Person p) {
		long id = next++;
		p.setId(id);
		ppl.put(id, p);
		return id;
	}
	@Override
	public void update(Person p) {
		ppl.put(p.getId(), p);
	}

	@Override
	public void delete(Long id) {
		ppl.remove(id);
	}
}
```

Testing

Postman is popular for testing webservices