



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

@RestController is a composite of @Controller and @ResponseBody

Controller

```
@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}

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

RedirectView
“redirect: ” String
won’t work!

Person & Person Service

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

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

```
@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 webservice

The screenshot displays the Postman application window. The top menu bar includes File, Edit, View, and Help. Below it is a toolbar with buttons for New, Import, Runner, and a workspace selector set to 'My Workspace'. On the right of the toolbar are icons for a test runner, a sync icon, a share icon, a heart icon, and a Sign In button.

The left sidebar contains a search bar labeled 'Filter' and two tabs: 'History' and 'Collections'. Under the 'History' tab, there is a list of requests under the heading 'Today'. The list shows several GET requests to 'http://localhost:8080/person/1' and 'http://localhost:8080/person/4'. A 'Clear all' button is located above the list.

The main panel shows a selected request named 'test' with the method 'GET' and the URL 'http://localhost:8080/person/1'. The 'Headers' tab is active, displaying a table with two headers: 'Content-Type' and 'Accept', both set to 'application/json'. Below the headers, there is a 'Body' tab showing a JSON response in 'Pretty' format:

```
{  "id": 1,  "name": "Test",  "age": 28}
```

. The status bar at the bottom indicates 'Status: 200 OK', 'Time: 14 ms', and 'Size: 161 B'.