



CS544 EA

# Integration

REST: sending / receiving XML

# Send / Receive XML

- Enabling XML is simply adding a dependency

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

- With this added our webservice will be able to do both JSON and XML

# Content Negotiation

- HTTP headers are used to indicate what the incoming and outgoing data types should be
  - **Content-Type** indicates what the browser sends
  - **Accept** indicates what the browser wants
- The mime type for XML is:
  - application/xml

# Controller

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

    @GetMapping("/person/")
    public List<Person> getAll() {
        return personService.getAll();
    }

    @GetMapping("/person/{id}")
    public Person get(@PathVariable long id) {
        return personService.get(id);
    }

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

    @PutMapping("/person/{id}")
    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);
    }
}
```

By not specifying produces / consumes  
a controller method can be used for  
both XML and JSON

# Example Receiving

The screenshot displays the Postman application window. The top menu bar includes File, Edit, View, and Help. Below the menu 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 red circle, a person, a bell, a heart, and a 'Sign In' button.

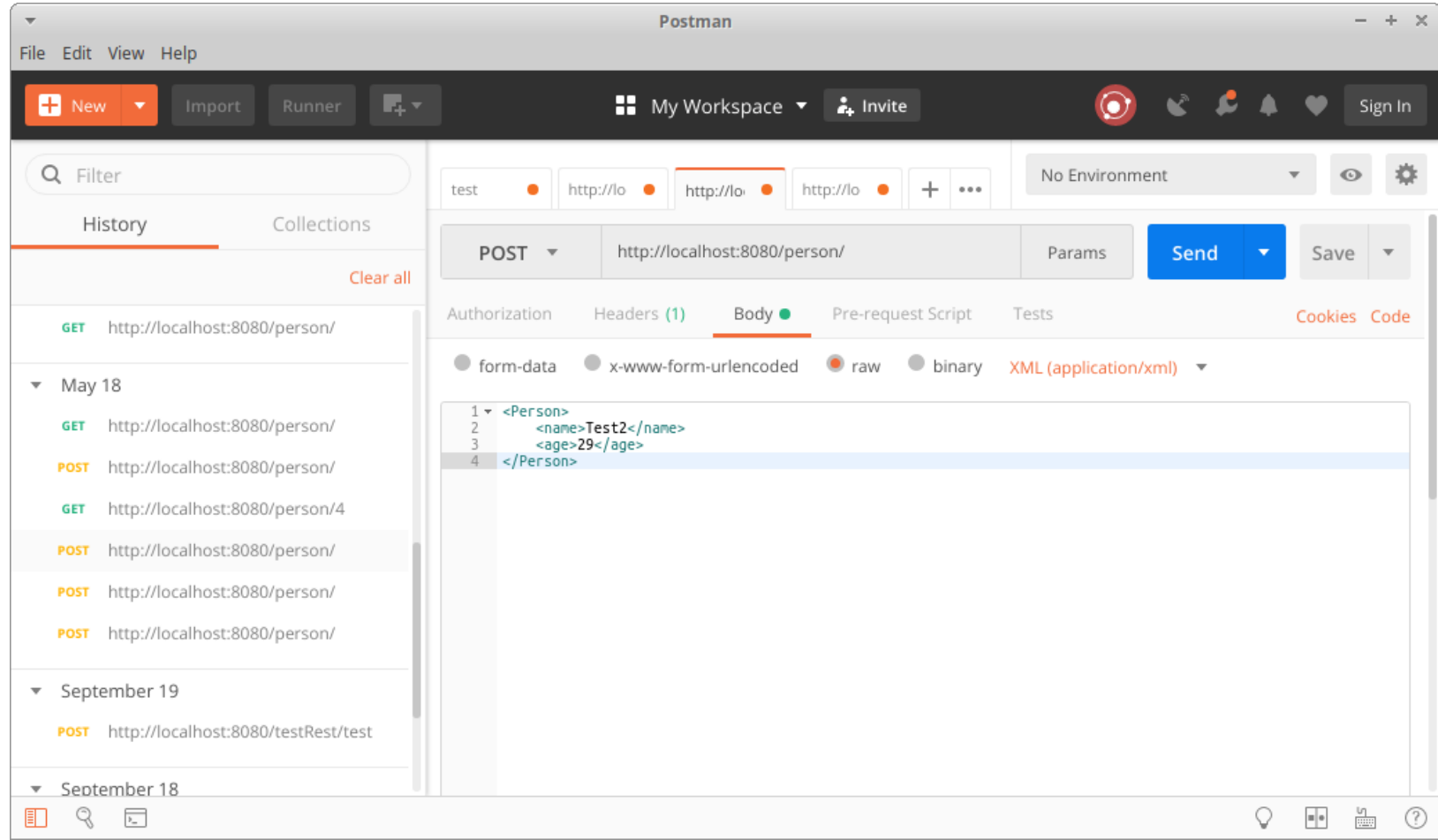
The left sidebar shows a 'History' tab with a search filter and a 'Clear all' button. It lists several requests from May 18 and September 19, including GET and POST requests to various endpoints like http://localhost:8080/person/ and http://localhost:8080/testRest/test.

The main workspace is configured for a GET request to http://localhost:8080/person/1. The 'Headers' tab is active, showing two headers: 'Content-Type' and 'Accept', both set to 'application/xml'. The 'Body' tab is also visible, showing a JSON response in 'Pretty' format: 

```
<Person>
  <id>1</id>
  <name>Test</name>
  <age>28</age>
</Person>
```

 The status bar at the bottom indicates a successful response with status 200 OK, time 26 ms, and size 186 B.

# Example Sending



# JAXB Mapping

- The Object to XML Mapping (OXM) defaults to:
  - Class get its own tag
  - Every property gets its own tag inside it
  - An object inside an object is nested
- If you want to change the name of a tag
  - Or use an XML attribute
  - Map them with annotations

# Basic OXM Mapping

```
@JacksonXmlElement(localName = "Customer")
public class Person {
    @JacksonXmlProperty(isAttribute = true)
    private Long id;
    @JacksonXmlProperty(localName = "firstName")
    private String name;
    private int age;
```

The screenshot shows a REST client interface with the following components:

- Request Bar:** Method **GET**, URL **http://localhost:8080/person/1**, and a **Send** button.
- Headers Tab:** Contains a table with headers:

	KEY	VALUE	DESCRIPTION	...
<input checked="" type="checkbox"/>	Content-Type	application/xml		
<input checked="" type="checkbox"/>	Accept	application/xml		
	Key	Value	Description	
- Body Tab:** Shows the response body in XML format:

```
1 <Customer id="1">
2   <age>28</age>
3   <firstName>Test</firstName>
4 </Customer>
```
- Status Bar:** Shows **Status: 200 OK** and **Time: 35 ms**.