Week 1 Day 5

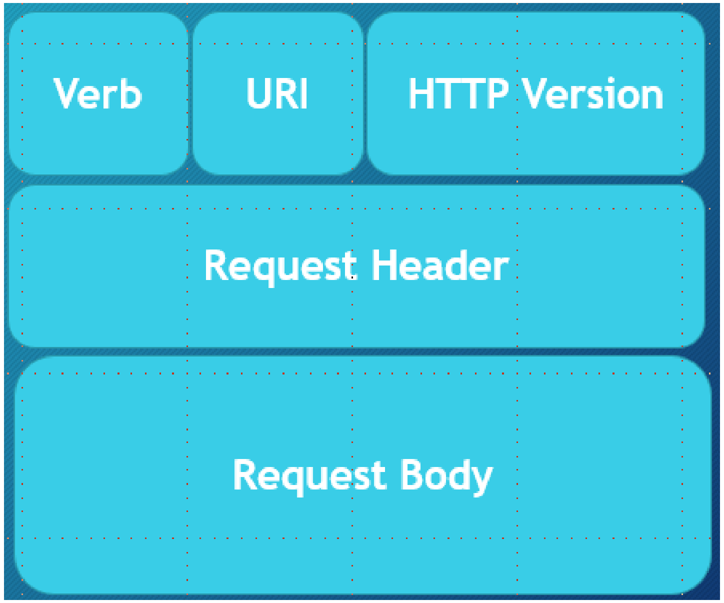
Java Web API

REVATURE

Introduction to HTTP

Hypertext Transfer Protocol

- Send information between clients and servers
- HTTP Request
- HTTP Response



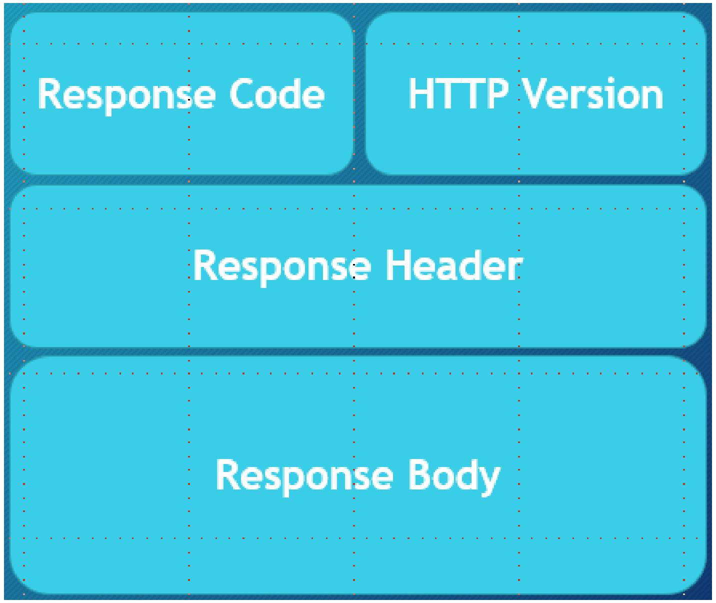
Verb

URI

HTTP Version

Request Header

Request Body



Response Code

HTTP Version

Response Header

Response Body

HTTP: Verbs/Methods

- Idempotent Methods:
  - Repeated requests return the same result
- Safe
  - Request does not alter state
- Four most used methods follow CRUD
  - Create, Read, Update, Delete

- GET
  - Retrieve Information
- POST
  - Send Information

- PUT
  - Update Information
- DELETE
  - Delete Information

HTTP: Status Codes

- Gives information about the request

- Informational: 100-199
- Successful: 200-299
  - 200 OK
  - 201 Created
- Redirected: 300-399
  - 301 Moved Permanently
  - 302 Found

- Client Error: 400-499
  - 401 Unauthorized
  - 404 Not Found
- Server Error: 500-519
  - 500 Internal Server Error
  - 503 Service Unavailable

Javalin

Webservice Framework

REVATURE

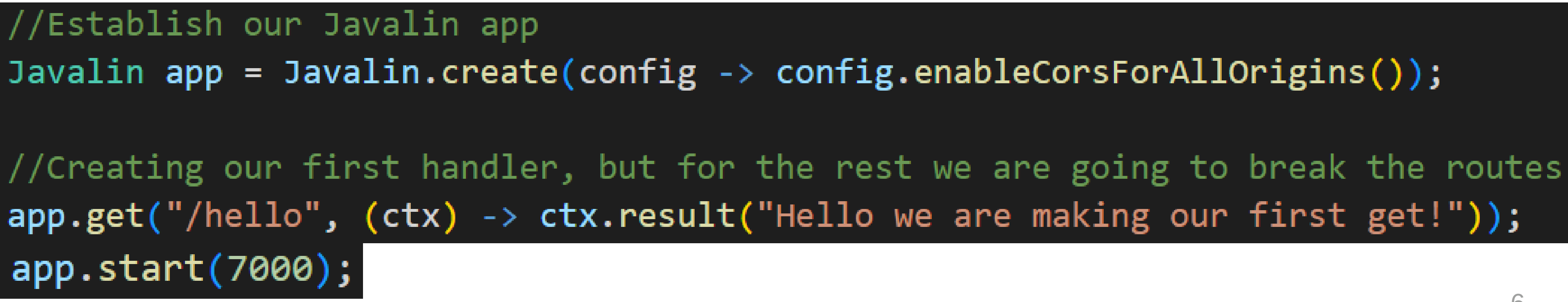
Javalin Overview

Lightweight Java web framework

- Used to created java webservices
- Runs on an embedded Jetty Server
- Needed Jar Dependencies
  - Javalin
  - Slf4j
  - Jackson

Javalin: Configuration and HTTP Handlers

- Create a new Javalin object and call the start method
- Register API endpoints with Http Handlers
  - Include the resource URL
  - Include the implementation for the handler
- Handlers receive a context object
  - Includes information about the request for your use

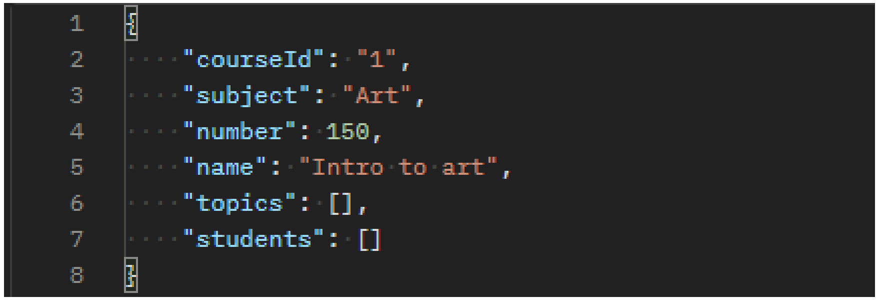


```
//Establish our Javalin app
Javalin app = Javalin.create(config -> config.enableCorsForAllOrigins());

//Creating our first handler, but for the rest we are going to break the routes
app.get("/hello", (ctx) -> ctx.result("Hello we are making our first get!"));
app.start(7000);
```

Jackson API

- API to convert Java objects to JSON objects to send in HTTP Requests
- JSON:
  - Javascript Object Notation
  - Data exchange format
  - String format of objects

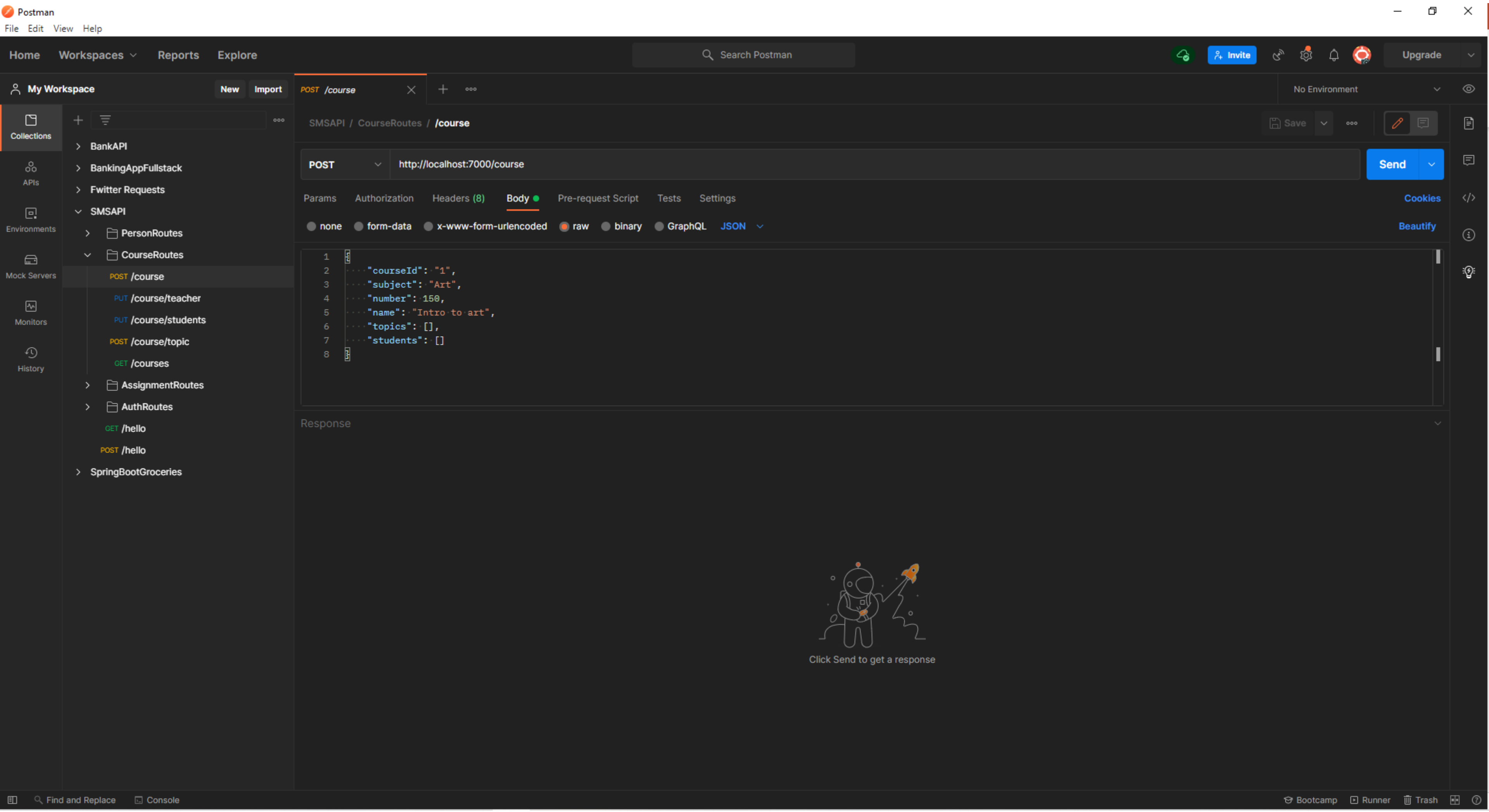


```
{
    "courseId": "1",
    "subject": "Art",
    "number": 150,
    "name": "Intro to art",
    "topics": [],
    "students": []
}
```

REVATURE

Java Setting Up Javalin Demo

Postman



POST http://localhost:7000/course

Body: ○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "courseId": "1",
    "subject": "Art",
    "number": 150,
    "name": "Intro to art",
    "topics": [],
    "students": []
}
```

Logging: Logback

- The act of recording events that occur in your program
  - Gives developers information
- Logback
  - Successor to Log4J
  - Implementation of slf4j
  - Allows us to log message and or errors in our Java programs
  - Configured in a pom.xml file in the java/resources folder

Logback: Logging Levels/Thresholds

- Describes if/when we should log

1. ALL: log everything
2. DEBUG: fine grained informational messages
3. INFO: coarse grained informational messages
4. WARN: possible problem
5. ERROR: exception/application issue
6. FATAL: sever error/exception
7. OFF: no logging

Java Logging Demo