Schedule system for substitute teachers

A very similar system to your assignment!

A web application for a substitute teacher schedule

Let’s say a company offers substitute teachers to schools. They need a web service to keep track of what teacher is assigned to what school at what time.

The application should be able to answer the following requests:

- All assignments (all teachers, all dates)
- All assignments at given date (YYYY-mm-dd)
- All assignments for a given teacher (substitute_id - the id of a substitute teacher)
- All assignments for a given substitute_id at a given date

Accessing the web application

You access the web application using a URL with GET parameters.

The following GET parameters exist in this web API:

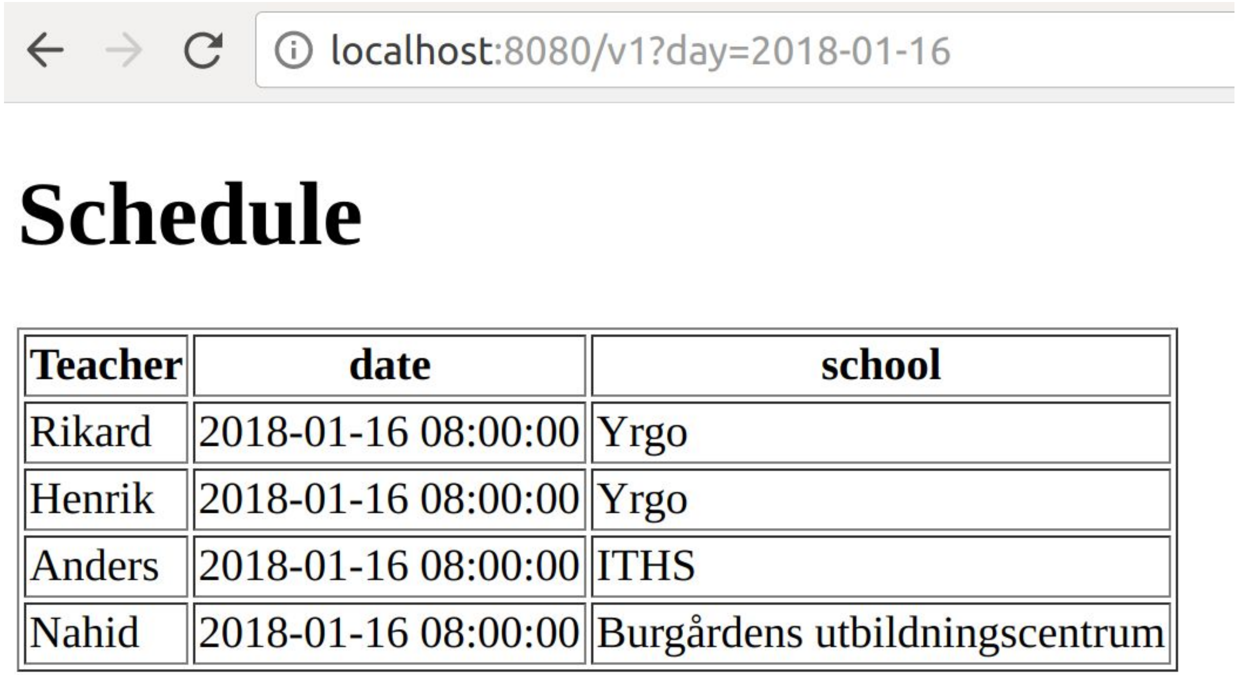
- `day` - the date of the schedule data - A String with format `YYYY-mm-dd` (optional)
- `substitute_id` - The substitute teachers employee number (an integer) (optional)

Leaving out `day` will list all dates of the schedule

Leaving out `substitute_id`, will list all schedules for all teachers

Leaving out both `day` and `substitute_id` will list all schedules at all dates for all teachers.

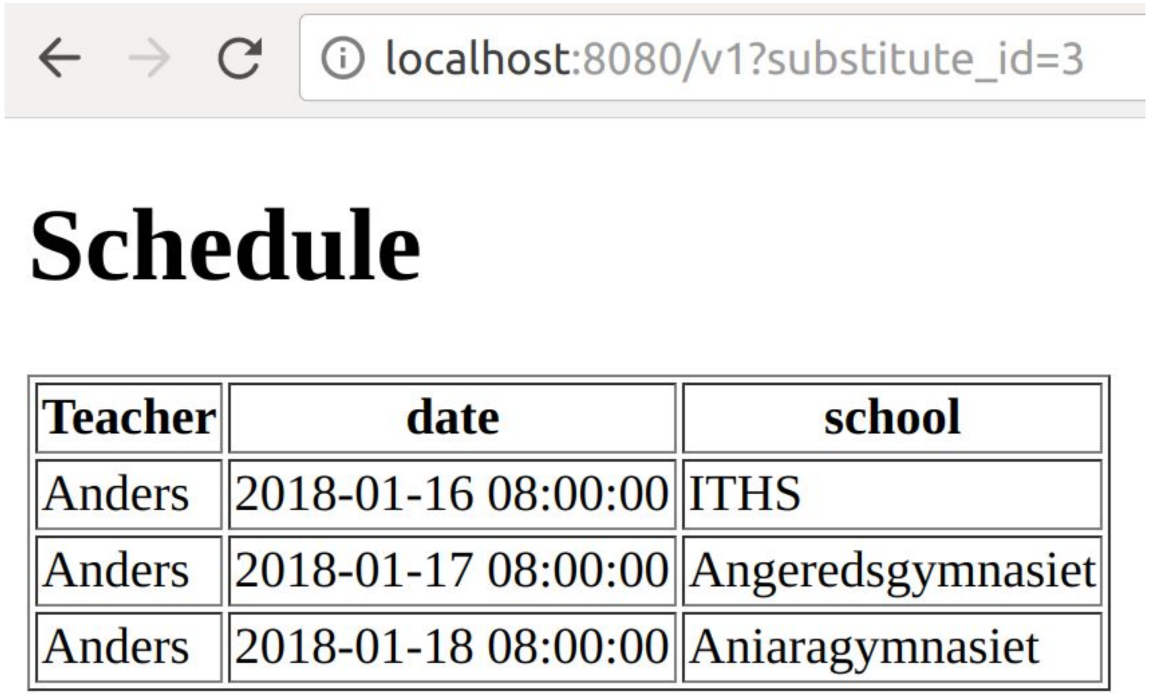
Example queries 1 - a certain day


localhost:8080/v1?day=2018-01-16

Schedule

Teacher	date	school
Rikard	2018-01-16 08:00:00	Yrgo
Henrik	2018-01-16 08:00:00	Yrgo
Anders	2018-01-16 08:00:00	ITHS
Nahid	2018-01-16 08:00:00	Burgårdens utbildningscentrum

Example queries, 2 - a certain teacher

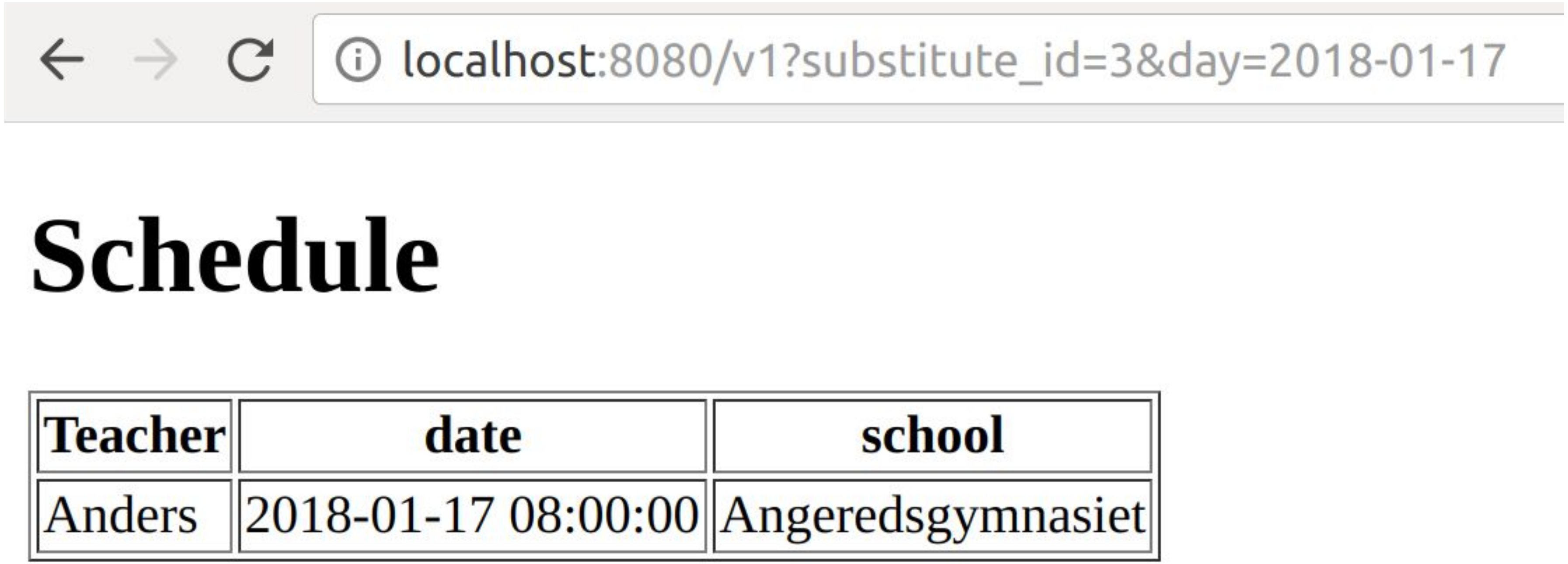


localhost:8080/v1?substitute_id=3

Schedule

Teacher	date	school
Anders	2018-01-16 08:00:00	ITHS
Anders	2018-01-17 08:00:00	Angeredsgymnasiet
Anders	2018-01-18 08:00:00	Aniaragymnasiet

Example queries, 3 - certain teacher, a certain day

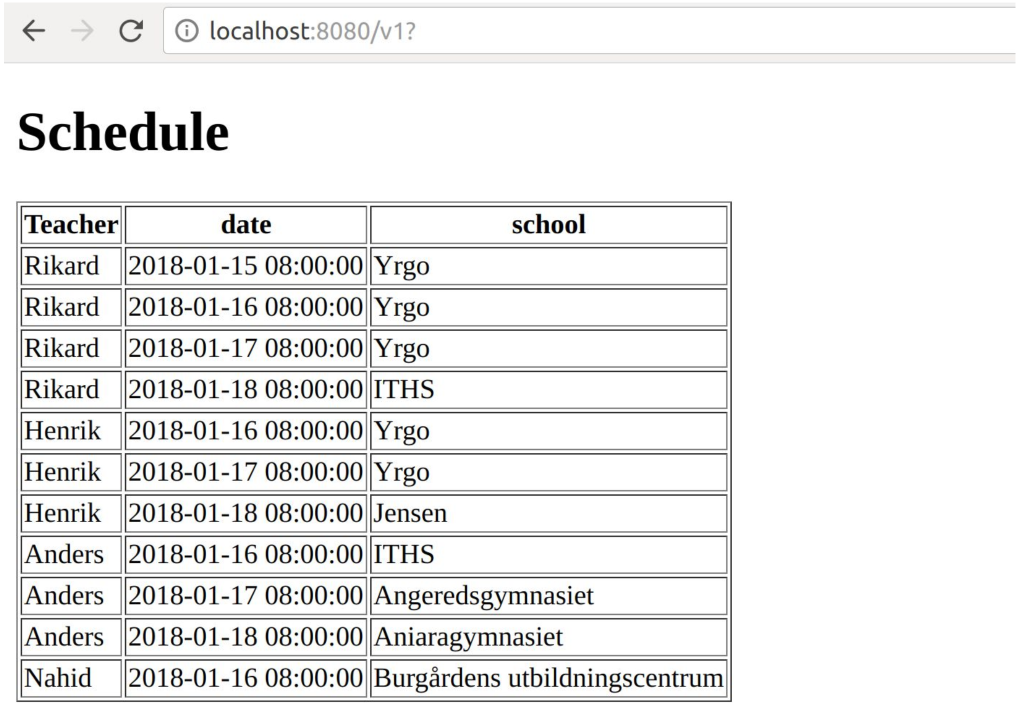


localhost:8080/v1?substitute_id=3&day=2018-01-17

Schedule

Teacher	date	school
Anders	2018-01-17 08:00:00	Angeredsgymnasiet

Example queries, 4 - Everyone, all the time!



Schedule

Teacher	date	school
Rikard	2018-01-15 08:00:00	Yrgo
Rikard	2018-01-16 08:00:00	Yrgo
Rikard	2018-01-17 08:00:00	Yrgo
Rikard	2018-01-18 08:00:00	ITHS
Henrik	2018-01-16 08:00:00	Yrgo
Henrik	2018-01-17 08:00:00	Yrgo
Henrik	2018-01-18 08:00:00	Jensen
Anders	2018-01-16 08:00:00	ITHS
Anders	2018-01-17 08:00:00	Angeredsgymnasiet
Anders	2018-01-18 08:00:00	Aniaragymnasiet
Nahid	2018-01-16 08:00:00	Burgårdens utbildningscentrum

Download the source code so you can follow

Start with cloning the following git hub repo:

https://github.com/progund/web-misc/

Change directory to `web-misc/nahid-sysint/`

Run `tree` to get a directory overview

Directory overview

```
. ← this is where you will be working - don’t leave this directory ;-)
├── clean.sh
├── download_sqlite_driver.sh
├── download_winstone.sh
├── compile_servlet_and_start_winstone.sh
├── winstone.jar
└── www ← this is the “web root” it can be named anything, but www is not uncommon
    └── WEB-INF ← this is the directory with stuff private to winstone; servlets, settings, resources
        ├── classes
        │   └── se
        │       └── yrgo
        │           └── schedule
        │               ├── AccessException.java
        │               ├── Assignment.java
        │               ├── AssignmentsFactory.java
        │               ├── Assignments.java
        │               ├── DatabaseAssignments.java
        │               ├── DBHelper.java
        │               ├── FormatterFactory.java
        │               ├── Formatter.java
        │               ├── HtmlFormatter.java
        │               ├── ParamParser.java
        │               └── ScheduleServlet.java ← This is the servlet! Note the package...
        ├── lib ← this is the directory to put third party libraries (JAR files) e.g. database, json etc
        │   └── sqlite-jdbc.jar
        ├── resources
        │   └── vikarie.db
        └── web.xml ← this is the settings file for winstone - defines servlets and maps them to URL patterns etc

8 directories, 19 files
```

Let’s start at looking at web.xml

```
<?xml version="1.0" encoding="utf-8"?>

<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
  <servlet>
    <servlet-name>schedule</servlet-name>
    <servlet-class>se.yrgo.schedule.ScheduleServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>schedule</servlet-name>
    <url-pattern>/v1/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Start Winstone and try out various URL

```
$ ./compile_servlet_and_start_winstone.sh

# Valid dates in the database: 2018-01-15 - 2018-01-18

# Valid substitute_ids:        1,2,3,4



#############################
########End part one!########
#############################
```

Part 2 - investigating the code

Next, we’ll look at the servlet code.

Let’s look at se.yrgo.schedule.ScheduleServlet

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
  // Read the request as UTF-8
  request.setCharacterEncoding(UTF_8.name());

  // Parse the arguments - see ParamParser class - you don’t have to write a parser but...
  ParamParser parser = new ParamParser(request);
  // Set the content type (using the parser)
  response.setContentType(parser.contentType());
  // To write the response, we're using a PrintWriter
  PrintWriter out =
    new PrintWriter(new OutputStreamWriter(response.getOutputStream(),
                                           UTF_8), true);
```

Let’s look at se.yrgo.schedule.ScheduleServlet

```
// Get access to the database, using a factory Assignments is an interface
Assignments db = AssignmentsFactory.getAssignments();
// Start with an empty list (makes code easier)
List<Assignment> assignments = new ArrayList<>();
// Call the correct method, depending on the parser's type value
  StringBuilder table;
  switch (parser.type()) { // What do they want? All teachers, or some particular teacher?
    case ALL:
      assignments = db.all();
      break;
    case TEACHER_ID_AND_DAY:
      assignments = db.forTeacherAt(parser.teacherId(), parser.day());
      break;
    case DAY:
      assignments = db.at(parser.day());
      break;
    case TEACHER_ID:
      assignments = db.forTeacher(parser.teacherId());
  }
}  // Note! try-catch not shown here for brevity!!!
```

Let’s look at se.yrgo.schedule.ScheduleServlet

```
      // Get a formatter, by asking the parser for the format (defaults to HTML)
      Formatter formatter = FormatterFactory.getFormatter(parser.format());
      // Format the result to the format according to the parser:
      String result = formatter.format(assignments);
      // Print the result and close the PrintWriter
      out.println(result);
      out.close();
    }

/*
Using the parser class (we wrote it ourselves!) eliminates the if-statements!
Without it we’d have to check the GET parameters and use IF-statements to understand
what to do.

In your assignment, you can use IF-statements or write a parser (you can borrow code from our
parser ;-) ) - it’s up to you.
*/
```

Without the parser....

Without the parser, we’d have had to do this:

```
// in doGet()...
String format = request.getParameter("format");

if (format == null) {
  // handle default format (HTML? something else?)
} else if (format.equals("xml")) {
  // create response in XML format...
} else if (format.equals("json")) {
  // create response in JSON format...
}
```

What’s wrong with this web API?

- It only answers in HTML format (a web page)
- HTML isn’t very easy for applications to parser (make sense of)

So what should we do? Nothing! You should do! :-)

Your assignment is to implement this web API so that it responds with either XML or JSON. See http://wiki.juneday.se/mediawiki/index.php/Assignment:SubstituteTeacherScheduler

Tasks 1 and 2 are about making the servlet care about formats.

What do we need to learn before the assignment?

The data about schedules, schools and teachers are in a database. So we need to learn how to access that from Java (JDBC).

We also need to learn about tiered architectures - how to move database stuff out of the servlet and into a specialized class (keep the servlet protected from knowing too much about low level stuff).

We also need to learn about how to create JSON and XML from Java.