LECTURE 7: WEB SERVICES - RESTFUL API – CRUD OPERATIONS

MOHAMAD GHARIB
UNIVERSITY OF TARTU
Representational state transfer (REST) is a software architectural style designed for distributed hypermedia, which defines a set of constraints to be used for creating Web services.
REST - recap

HTTP GET request

HTTP response
HTML, JSON, etc.

HTTP GET request

HTTP response
HTML, JSON, etc.

HTTP DELETE request

HTTP response
XML, JSON, etc.

Web server

Plants list

Read/Query

Result

Read/Query

Result

Delete

Booking list

Loaded/updated depending on GET
REST Architecture
REST Architecture

Controller-Service-Repository pattern
A **Controller** maps **clients** requests to corresponding **services/functionalities** (Java methods).
A **Controller** maps **clients' requests** to corresponding **services/functionalities** (Java methods).
A **Controller** maps **clients' requests** (Java methods) to corresponding **services/functionalities**.
A **Controller** maps **clients' requests** to corresponding **services/functionalities** (Java methods).
A Controller maps clients requests to corresponding services/functionalities (Java methods).
A Service (a method) implements the business logic underlining the service
A Service (a method) implements the business logic underlining the service.
A Repository is responsible for retrieving, storing, updating and deleting some set of data.
Java relies on the object-orient paradigm, and **databases** store data as rows and columns in a series of tables.
Java relies on the object-orient paradigm, and databases store data as rows and columns in a series of tables.
Java relies on the object-orient paradigm, and databases store data as rows and columns in a series of tables.
Object–relational mapping (ORM) is a for converting data between incompatible type systems using object-oriented programming languages.
JPA provides Java developers with an object/relational mapping facility for managing relational data in Java applications.
## REST Architecture - Repository

### Spring Boot Server

#### Plant catalog

<table>
<thead>
<tr>
<th>Method</th>
<th>Endpoint</th>
<th>CRUD Operation</th>
<th>Spring Boot Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/plants</td>
<td>fetchAllPlants()</td>
<td>plantsRepository.findAll()</td>
</tr>
<tr>
<td>GET</td>
<td>/plants/id</td>
<td>fetchPlanById(id)</td>
<td>plantsRepository.findById(id)</td>
</tr>
<tr>
<td>POST</td>
<td>/plants</td>
<td>createPlant(plant)</td>
<td>plantsRepository.save(plant)</td>
</tr>
<tr>
<td>PUT</td>
<td>/plants/id</td>
<td>updatePlant(id, plant)</td>
<td>plantsRepository.save(plant)</td>
</tr>
<tr>
<td>DELETE</td>
<td>/plants/id</td>
<td>deletePlant(id)</td>
<td>plantsRepository.deleteById(id)</td>
</tr>
</tbody>
</table>

**JPA & CRUD Operations**
The Controller-Service-Repository (CSR) pattern - practical example

You will not be at this point unless you have already identified key entities, their relationships and required operations.

In this example, we have two entities:

- Plant
- Reservation
The CSR pattern - practical example

You will not be at this point unless you have already identified key entities, their relationships and required operations.

In this example, we have two entities:
- Plant
- Reservation

You deal with entities one by one, starting from the most important.
The CSR pattern - practical example

com.~.plants
- Plant (Entity)
- PlantController
- PlantService
- PlantRepository

com.~.reservations
- Reservation (Entity)
- ReservationController
- ReservationService
- ReservationRepository
The CSR pattern - practical example

com.~.plants
- Plant (Entity)
- PlantController
- PlantService
- PlantRepository

com.~.reservations
- Reservation (Entity)
- ReservationController
- ReservationService
- ReservationRepository
The CSR pattern - practical example

**com.~.plants**
- Plant (Entity)
- PlantController
- PlantService (hardcoded resource)
- PlantRepository

**com .~. reservations**
- Reservation (Entity)
- ReservationController
- ReservationService
- ReservationRepository
The CSR pattern - practical example

com.~.plants
- Plant (Entity)
- PlantController
- PlantService
- PlantRepository

com.~.reservations
- Reservation (Entity)
- ReservationController
- ReservationService
- ReservationRepository
The CSR pattern - practical example

com.~.plants

- Plant (Entity)
- PlantController
- PlantService
- PlantRepository

com .~. reservations (Entity Rel.)

- Reservation (Entity)
- ReservationController
- ReservationService
- ReservationRepository
Thank You for your attention

Mohamad Gharib
mohamad.gharib@ut.ee

unitartu
tartuuniversity