MTAT.03.229
Enterprise System Integration

Lecture 6: Hypermedia REST

Marlon Dumas
University of Tartu

Based on slides by Luciano García-Bañuelos
Recap from last week

- CRUD REST services
- Multiple URIs, multiple HTTP verbs, and HTTP Status Codes
- HTTP verbs
  - GET, POST, PUT, DELETE
- HTTP Status Codes
  - 1xx - Metadata
  - 2xx – Everything’s fine
  - 3xx – Redirection
  - 4xx – Client did something wrong
  - 5xx – Server did a bad thing
Hypermedia within our application?

• HATEOAS: Hypermedia as the engine of application state

• Roy Fielding’s view:

  The simultaneous presentation of information and controls such that the information becomes the affordance through which the user obtains choices and selects actions.

• A simpler wording:

  Include links into the resource representation. Let the user (human or program) continue the execution by following one of those links.
Hypermedia in the Web

Lazy loading (the user must explicit follow the link)

Eager loading (the browser downloads the image without asking anything to the user)
Hypermedia in the Web

<form method="get">
  <label>Search term:</label>
  <input name="query" type="text" value=""/>
  <input type="submit"/>
</form>

GET /books/?query=rest HTTP/1.1
Host: www.example.org

POST /keywords HTTP/1.1
Host: www.example.org
Content-Type: application/x-www-form-urlencoded

keywords=SOAP+REST+WS

<form method="post" action="http://www.example.org/keywords">
  <label>Keywords:</label>
  <input name="keywords" type="text" value=""/>
  <input type="submit"/>
</form>
Designing Hypermedia APIs
Example: PO creation (BuildIt)

HTTP/1.1 201 Created
Location: /orders/1253

 POST /orders/  

http://rentit.com/rest

}<purchaseOrder>
  <start>7-10-2016</start>
  <end>11-10-2016</end>
  <plant>
    <name>Excavator</name>
  <plant>
</purchaseOrder>
Example: PO retrieval

GET /orders/1253

HTTP/1.1 200 Ok

```
<purchaseOrder>
  <startDate>7-10-2016</startDate>
  <endDate>11-10-2016</endDate>
  <cost>1500.00</cost>
  <plant>
    <sku>exc1253ab98</sku>
    <name>Excavator</name>
  </plant>
  <links>
    <link rel="accept" href="/orders/1253/accept" method="POST"/>
    <link rel="reject" href="/orders/1253/accept" method="DELETE"/>
  </links>
</purchaseOrder>
```
Our example application

```
Our example application
```

```
PurchaseOrderRestController

POExtRequestResource

PurchaseOrderResource

PlantInvEntryResource

PurchaseOrderService

createPurchaseOrder()
acceptPurchaseOrder()
rejectPurchaseOrder()
resubmitPurcherOrder()
extendRentalPeriod()
acceptRentalPeriodExtension()
rejectRentalPeriodExtension()

<<Enumeration>>
POStatus

PENDING
OPEN
PENDING_EXTENSION
CLOSED
REJECTED

<<Enumeration>>
ExtStatus

OPEN
ACCEPTED
REJECTED

PurchaseOrder

startDate: Date
endDate: Date
total: Money
status: POStatus

Customer

PlantReservation

plant

1

customer 0..*

reservations

PlantInvEntry

name: String
description: String
price: Money

POExtRequestResource

endDate: Date
status: ExtStatus

PurchaseOrderRestController

```

```
Customer PlantInvEntry
name: String
description: String
price: Money
PurchaseOrder
startDate: Date
endDate: Date
total: Money
status: POStatus
PlantInvEntryResource
name: String
description: String
price: Money
POExtRequestResource
endDate: Date
status: ExtStatus
PurchaseOrderRestController
```
Our example application

Integration layer (REST adapter)

Application

Domain model
Resource life cycle: Purchase order

PurchaseOrderService
- createPurchaseOrder()
- acceptPurchaseOrder()
- rejectPurchaseOrder()
- resubmitPurchaseOrder()
- extendRentalPeriod()
- acceptRentalPeriodExtension()
- rejectRentalPeriodExtension()

- createPO
- pending confirmation
- acceptPO
- open
- rejectRPEXtension
- extendRentalPeriod
- pending extension
- closePO
- rejected
- resubmitPO
- closed
Hypermedia API

<table>
<thead>
<tr>
<th>Verb</th>
<th>URI template</th>
<th>Relation</th>
<th>Current state</th>
<th>New state</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/orders</td>
<td>createPO</td>
<td></td>
<td>Pending confirmation</td>
<td>Submit partial representation</td>
</tr>
<tr>
<td>POST</td>
<td>/orders/{oid}/accept</td>
<td>acceptPO</td>
<td>Pending confirmation</td>
<td>Open</td>
<td>Use empty body</td>
</tr>
<tr>
<td>DELETE</td>
<td>/orders/{oid}/accept</td>
<td>rejectPO</td>
<td>Pending confirmation</td>
<td>Rejected</td>
<td></td>
</tr>
</tbody>
</table>
# Purchase Order API

<table>
<thead>
<tr>
<th>Method</th>
<th>URI template</th>
<th>Relation</th>
<th>Current state</th>
<th>New state</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/orders</td>
<td>createPO</td>
<td></td>
<td>Pending</td>
<td>Submit partial representation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>confirmation</td>
<td></td>
</tr>
<tr>
<td>POST</td>
<td>/orders/{oid}/accept</td>
<td>acceptPO</td>
<td>Pending</td>
<td>Open</td>
<td>Use empty body</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>confirmation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELETE</td>
<td>/orders/{oid}/accept</td>
<td>rejectPO</td>
<td>Pending</td>
<td>Rejected</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>confirmation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PUT</td>
<td>/orders/{oid}</td>
<td>resubmitPO</td>
<td>Rejected</td>
<td>Pending</td>
<td>Submit new full representation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>confirmation</td>
<td></td>
</tr>
<tr>
<td>DELETE</td>
<td>/orders/{oid}</td>
<td>closePO</td>
<td>Open</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>POST</td>
<td>/orders/{oid}/extensions</td>
<td>extendRentalPeriod</td>
<td>Open</td>
<td>Pending</td>
<td>Submit new end date</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>update</td>
<td></td>
</tr>
<tr>
<td>POST</td>
<td>/orders/{oid}/extensions/{eid}/accept</td>
<td>acceptRPExtension</td>
<td>Pending</td>
<td>Open</td>
<td>Use empty body</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>update</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELETE</td>
<td>/orders/{oid}/extensions/{eid}/accept</td>
<td>rejectRPExtension</td>
<td>Pending</td>
<td>Open</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>update</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Annotated State model

Events produced by BuildIt

POST /orders
createPO

POST /orders/{oid}/extensions/{eid}/accept
acceptRPExtension

POST /orders/{oid}/extensions/{eid}/accept
acceptPO

DELETE /orders/{oid}/accept
rejectPO

POST /orders/{oid}/accept
acceptPO

DELETE /orders/{oid}/accept
rejectPO

PUT /orders/{oid}
resubmitPO

POST /orders/{oid}/extensions
extendRentalPeriod

DELETE /orders/{oid}
closePO

POST /orders/{oid}/extensions/{eid}/accept
acceptRPExtension

DELETE /orders/{oid}/extensions/{eid}/accept
rejectRPExtension

pending confirmation

open

closed

rejected

HYPERMEDIA REST  LUCIANO GARCÍA-BAÑUELOS
Refinement: introducing plant reservations

- POST /orders/createPO
- POST /orders/{oid}/accept
- DELETE /orders/{oid}/accept
- rejectPO
- PUT /orders/{oid}
- resubmitPO
- POST /orders/{oid}/extensions/{eid}/accept
- acceptRPEXtension
- open
- closePO
- DELETE /orders/{oid}/extensions/{eid}/accept
- rejectRPEXtension
- pending extension
- POST /orders/{oid}/extensions
- extendRentalPeriod
- [plant available]
- reservePlant
- [plant unavailable]
Implementing Hypermedia APIs
Changes on resource classes

In this case, we will use the library "spring-hateoas". However, the classes required and their implementation are quite similar to those in our previous version.

Note that we will use JSON as the representation format.

Maven dependency
Group id: org.springframework.hateoas
Artifact id: spring-hateoas
Version: 0.19.0.RELEASE
public class PlantResourceAssembler
extends ResourceAssemblerSupport<Plant, PlantResource> {

    public PlantResourceAssembler() {
        super(PlantRESTController.class, PlantResource.class);
    }

    @Override
    public PlantResource toResource(Plant plant) {
        PlantResource resource;
        resource = createResourceWithId(plant.getId(), plant);
        resource.setName(plant.getName());
        resource.setDescription(plant.getDescription());
        resource.setPrice(plant.getPrice());
        return resource;
    }
}
@RestController
@RequestMapping("/api/orders")
public class PurchaseOrderRESTController {

@Autowired
PurchaseOrderService service;

@RequestMapping(method = RequestMethod.POST, value = "")
public ResponseEntity<PurchaseOrderResource> createPO(@RequestBody PurchaseOrder po)
    throws PlantUnavailableException {
    po.setStatus(POStatus.PENDING_CONFIRMATION);
    service.createPO(po);

    HttpHeaders headers = new HttpHeaders();
    URI uri = linkTo(PurchaseOrderRESTController.class).slash(po.getId()).toUri();
    headers.setLocation(uri);
    PurchaseOrderResource poRs = assembler.toResource(po);
    ResponseEntity<PurchaseOrderResource> response = new ResponseEntity<>(poRs, headers, HttpStatus.CREATED);
    return response;
}
}
@Controller
@Controller
@RequestMapping("/api/orders")
public class PurchaseOrderRESTController {

@RequestMapping(method = RequestMethod.GET, value = "{id}" )
public ResponseEntity<PurchaseOrderResource> getPO(@PathVariable Long id) {
    PurchaseOrder po = PurchaseOrder.findPurchaseOrder(id);
    PurchaseOrderResourceAssembler assembler = new PurchaseOrderResourceAssembler();
    PurchaseOrderResource resource = assembler.toResource(po);
    switch (po.getStatus()) {
        case PENDING_CONFIRMATION:
            Method _acceptPO = PurchaseOrderRESTController.class.getMethod("acceptPO", Long.class);
            String acceptLink = linkTo(_acceptPO, po.getId()).toUri().toString();
            resource.add(new ExtendedLink(acceptLink, "acceptPO", "POST"));

            Method _rejectPO = PurchaseOrderRESTController.class.getMethod("rejectPO", Long.class);
            String rejectLink = linkTo(_rejectPO, po.getId()).toUri().toString();
            resource.add(new ExtendedLink(rejectLink, "rejectPO", "DELETE"));
            break;
        default: break;
    }
    ResponseEntity<PurchaseOrderResource> response = new ResponseEntity<>(resource, HttpStatus.OK);
    return response;
}
}
Exception Handling in REST (Hypermedia) APIs
Example

GET /orders/1253

HTTP/1.1 200 Ok
<purchaseOrder>...
</purchaseOrder>

HTTP/1.1 404 Not found

HTTP/1.1 401 Unauthorized

HTTP/1.1 500 Internal Server Error
Exception handling & HTTP status codes

- 400 – Bad Request
  - The client has PUT or POST a resource representation that is in the right format, but contains invalid information

- 401 – Unauthorized
  - Proper credentials to operate on a resource were not provided
  - Response `WWW-Authenticate` header contains the type of authentication the server expects
    - Basic, digest, WSSE, etc.

- 403 – Forbidden
  - The client request is OK, but the server doesn’t want to process it (e.g., Restricted by IP address)
Exception handling & HTTP status codes

• 404 – Not Found
  ◦ The standard catch-all response

• 405 – Method Not Allowed
  ◦ The resource does not support a given method. Use “Allow” header to list the verbs the resource understands (e.g., Allow: GET, POST, PUT)

• 409 – Conflict
  ◦ Tried to change the state of the resource to something the server will not allow
Exception handling & HTTP status codes

• **406 – Not Acceptable**
  ◦ The server cannot reply with the representation requested by the client
    ◦ HTTP header `Accept`

• **415 Unsupported Media Type**
  ◦ The server cannot consume the resource format used by the client
    ◦ HTTP header `Content-type`
Exception handling within our REST controller

```java
@RestController
@RequestMapping("/rest/orders")
public class PurchaseOrderRESTController {

    @ExceptionHandler({PlantUnavailableException.class,
                        InvalidHirePeriodException.class})
    public ResponseEntity<String> handleBadRequest(Exception ex) {
        return new ResponseEntity<>(ex.getMessage(), HttpStatus.CONFLICT);
    }

    @ExceptionHandler(EntityNotFoundException.class)
    public ResponseEntity<String> handleNotFoundException(Exception ex) {
        return new ResponseEntity<>(ex.getMessage(), HttpStatus.NOT_FOUND);
    }
}
```