Karlsruher Entwicklertag 2015:
20. bis 22. Mai 2015

300 Jahre Karlsruhe, 10 Jahre Karlsruher Entwicklertag. Innovation in der Software - Innovation durch Software

Design REST Services with CXF JAX-RS implementation: best practices and lessons learned

Andrei Shakirin, Talend

ashakirin@talend.com
ashakirin.blogspot.com

The Apache Software Foundation
http://www.apache.org/

Agenda

- REST architectural style
- Design of REST API for Syncope domain
- Practical aspects of using CXF JAX-RS

About Me

- Software architect in Talend Team
- PMC and committer in Apache CXF and commiter in Apache Syncope projects

Representational State Transfer

The Apache Software Foundation
http://www.apache.org/

- Set of principals and restrictions
- HTTP is one instantiation of the REST
- The scope of REST architectural style: loosely coupled application

REST Principles

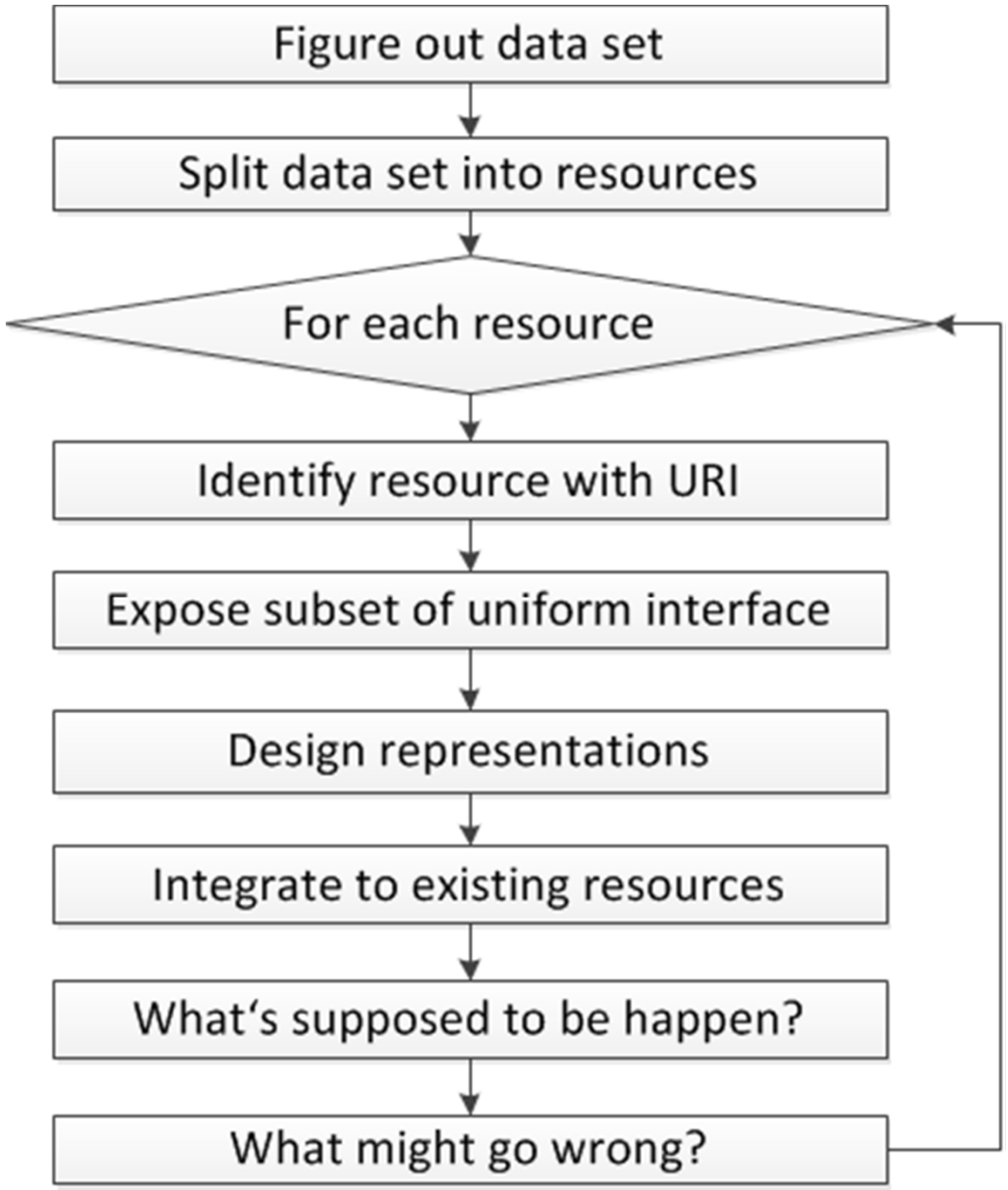
1. Everything has an ID
2. Using IDs to link things together: hypermedia and HATEOAS
3. Uniform interface
4. Interaction with resources through the representation
5. Communication is stateless

JAX-RS

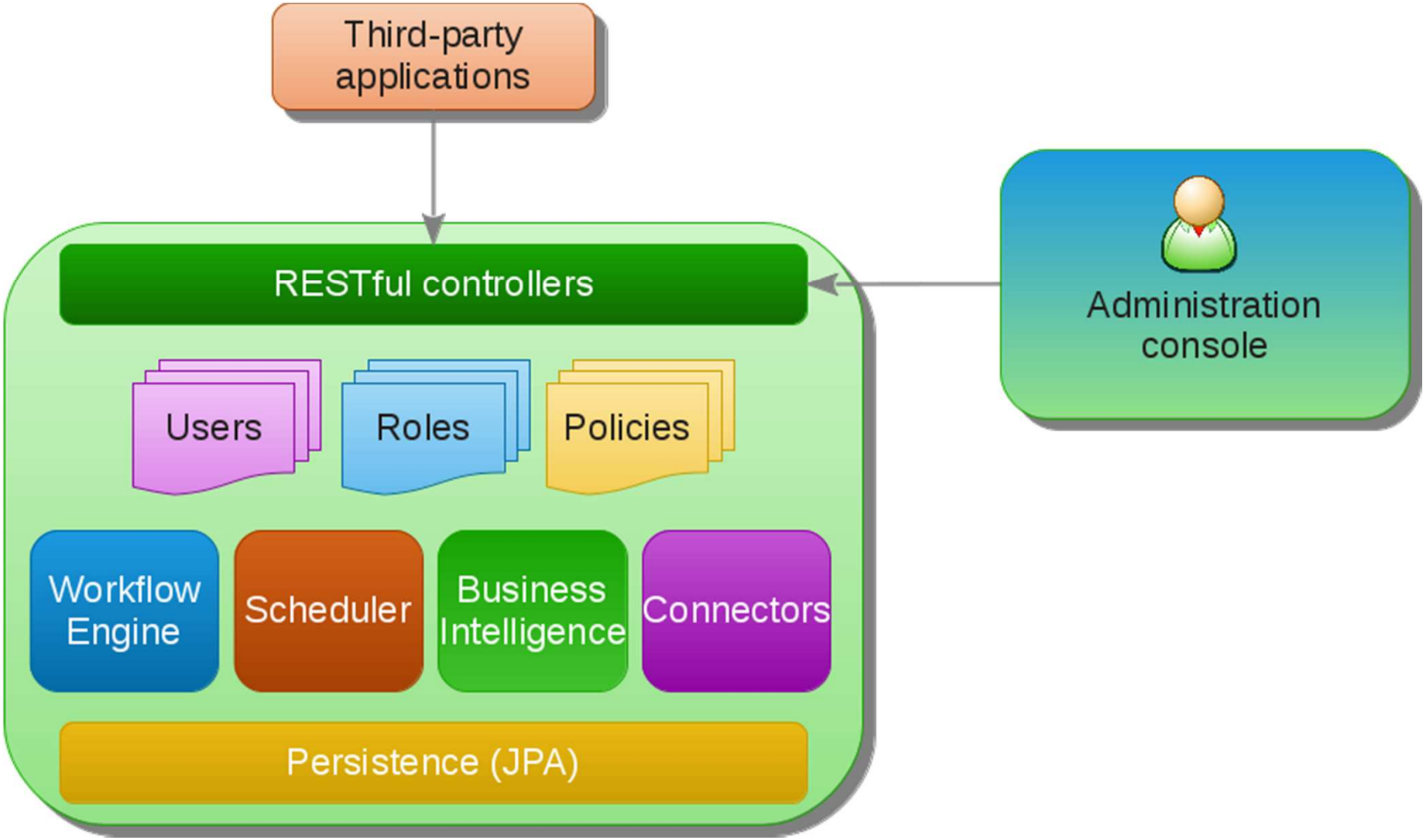
- Specification and Java API (JCP: JSR 311, 339, 370)
- Support in exposing a resource Java class (a POJO) as a web resource
- Versions 1.0, 1.1, 2.0 (client API, asynchronous API, filters and interceptors), 2.1 (in work)
- Implementations: Jersey, Apache CXF, Resteasy, Restlet …

REST API Design

The Apache Software Foundation
http://www.apache.org/



Apache Syncope



Syncope Domain Model

- Users (name, password, dates, attributes)
- Roles (name, owners, attributes)
- Entitlements (TASK_DELETE, ROLE_CREATE)
- Connectors (ConnID bundle: DatabaseTable, SOAP)
- External Resources (name, connector, mode, mapping)
- Tasks (type, resource, action, status)

Resources and URIs: Rules

- Resource is anything to be referenced
- Normally resources are the nouns
- Resources are coarse grained

- URIs: descriptive and well structured
- URIs: scoping information

Design Attempt

/users/addUser

/users/a1b2c3/verifyPassword

/roles/s8g3j8/updateRole

/tasks/submitTask

Don‘t do that!

Resources Types

1. Predefined top-level resources
2. Resource for collection of objects
3. Resource for every exposed objects
4. Resource representing results of algorithms

Top Level Resources

- Entry point to the API
- Home page or list of root entities (collections)

URIs:

`http(s)://api.syncope.org/administration`

vs

`http(s)://api.syncope.org/administration/rest/jaxrs/cxf`

Collection Resources

/users

/roles

/connectors

/resources

/tasks

Instance Resources

/users/user123
/users/user123/status
/roles/roleA
/roles/roleA/parent
/connectors/ldap
/connectors/ldap/bundles

No Hierarchy?

/relationships/user123,roleA

/color-blends/red;blue

Algorithm Resources

/users?failedLogin=true

/tasks?type=propagation&status=success

FIQL (Feed Item Query Language):

/tasks?_s=date=lt=2014-10-31;date=gt=2014-10-01;(type==sync)

Subset of Uniform Interface

Will the client will fetch resource of this type?

```
GET /users
GET /users/a1b2c3
```

Subset of Uniform Interface

Will the client delete resource of this type?

```
DELETE /users/a1b2c3
```

Subset of Uniform Interface

Will the client modify resource of this type?

```
PUT /users/a1b2c3
```

Subset of Uniform Interface

Will the client create resource of this type?

Who is in charge to determine URI: server / client?

Server:

```
POST /users
201 Created, Location=/users/a1b2c3
```

Client:

```
PUT /users/myuser
```

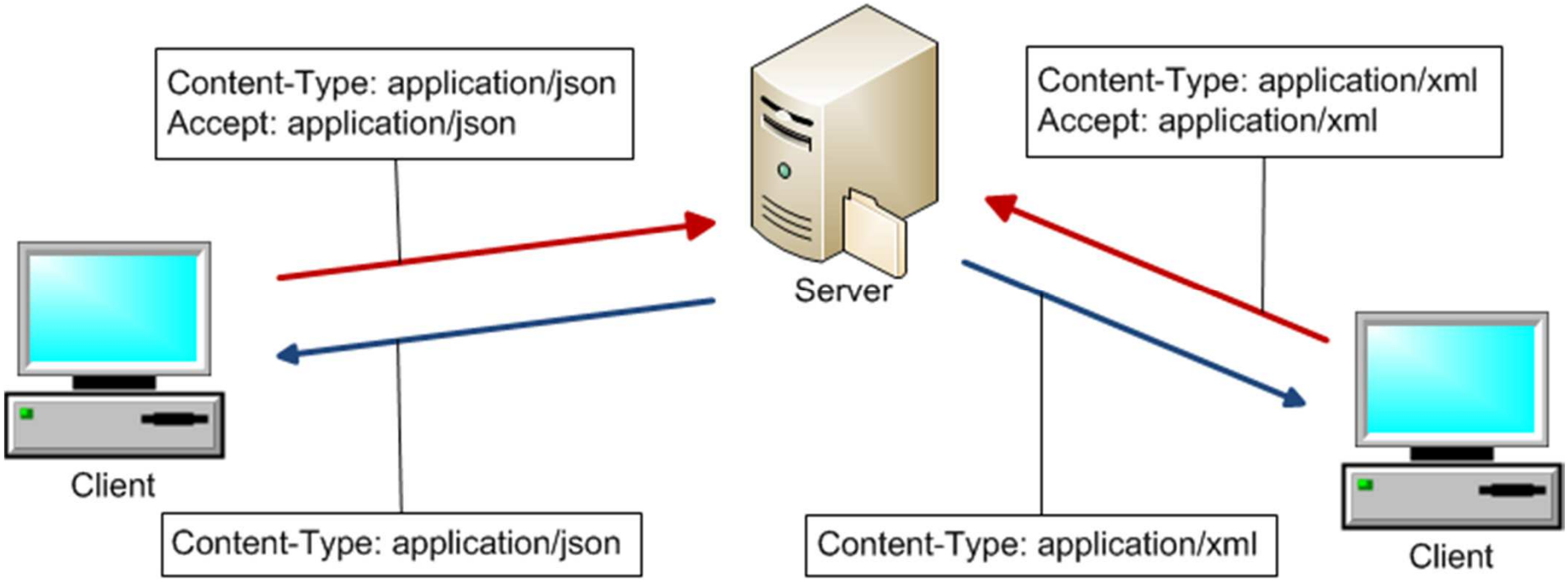
The Apache Software Foundation
http://www.apache.org/

Resource API: UserService

Representations: Media Types

The Apache Software Foundation
http://www.apache.org/

Data Format + Parsing rules



Representations: Media Types

- Standard:

  ```
  text/html
  application/json
  application/xml
  application/xhtml+xml
  application/x-www-form-urlencoded
  ```

- Custom:

  ```
  application/user+json
  application/vnd.mycompany-myformat
  ```

- Versioned:

  ```
  application/user+json&v2
  ```

Representations: JAX-RS

```
@Path("users")
@Consumes("application/json", "application/xml")
@Produces("application/json", "application/xml")
public interface UserService {

  @GET
  @Produces("application/json;qs=1.0", "application/xml;qs=0.75")
  Collection<UserTO> list();

  @POST
  @Consumes("application/json;q=1.0", "application/xml;q=0.25")
  Response create(UserTO userTO);
…
}
```

CXF: Entity Providers

The Apache Software Foundation
http://www.apache.org/

Data	CXF Provider / Java Type	Media Type
XML	JAXBElementProvider, SourceProvider / JAXBElement, XMLStreamReader(Writer)	`application/xml, application/*+xml, text/xml`
JSON	JSONProvider(Jettison), Jenkins	`application/json, application/*+json`
Multipart	MultipartProvider / Attachments, MultipartBody, Map	`multipart/mixed, multipart/related, …`
Binary	Attachment, InputStream, byte[], long[]	`application/octet-stream`
Form	FormEncodingProvider / MultiValuedMap	`application/octet-stream`
XSLT	XSLTJaxbProvider	`application/xml, application/*+xml, text/xml`
Atom	AtomFeedProvider / Feed (Apache Abdera)	`application/atom+xml, application/atom+xml;type=feed`

JSON: Jettison vs Jackson

The Apache Software Foundation
http://www.apache.org/

CXF JSONProvider (based on Jettison)

- Adopt XML structures to JSON (mapped, BadgerFish)
- STaX API (XMLStreamWriter, XMLStreamReader)
- Flexible and simple (prefixes, root elements, array serialization, unwrapping, XSLT transformations)
- Using: for small payloads, if flexibility required

```
<root><child>test</child><child>test</child></root>

{ "root" : { child : [ "test", "test" ] } }
```

JSON: Jettison vs Jackson

Jackson

- Not XML oriented
- Supports streaming, tree and data binding models
- Supports Jackson specific annotations
- Using: middle and large payloads, sophisticated class hierarchy

```
@JsonTypeInfo(use=Id.CLASS, include=As.PROPERTY,
property="class")
```

Representation: Links

- HTML/XHTML

```
<LINK rel="user" href="/users/a1b2c3">
<a rel="user" href="/users/a1b2c3">User</a>
```

- XML

```
<user xlink:href="/users/a1b2c3">User</user>
```

- JSON: ?

Links in JSON

- JSON HAL (Mike Kelly, IETF draft)
- Siren (Kevin Swiber)
- Collection + JSON (Mike Amudsen)
- Custom format

JSON HAL

The Apache Software Foundation
http://www.apache.org/

GET /tasks

Accept: application/hal+json

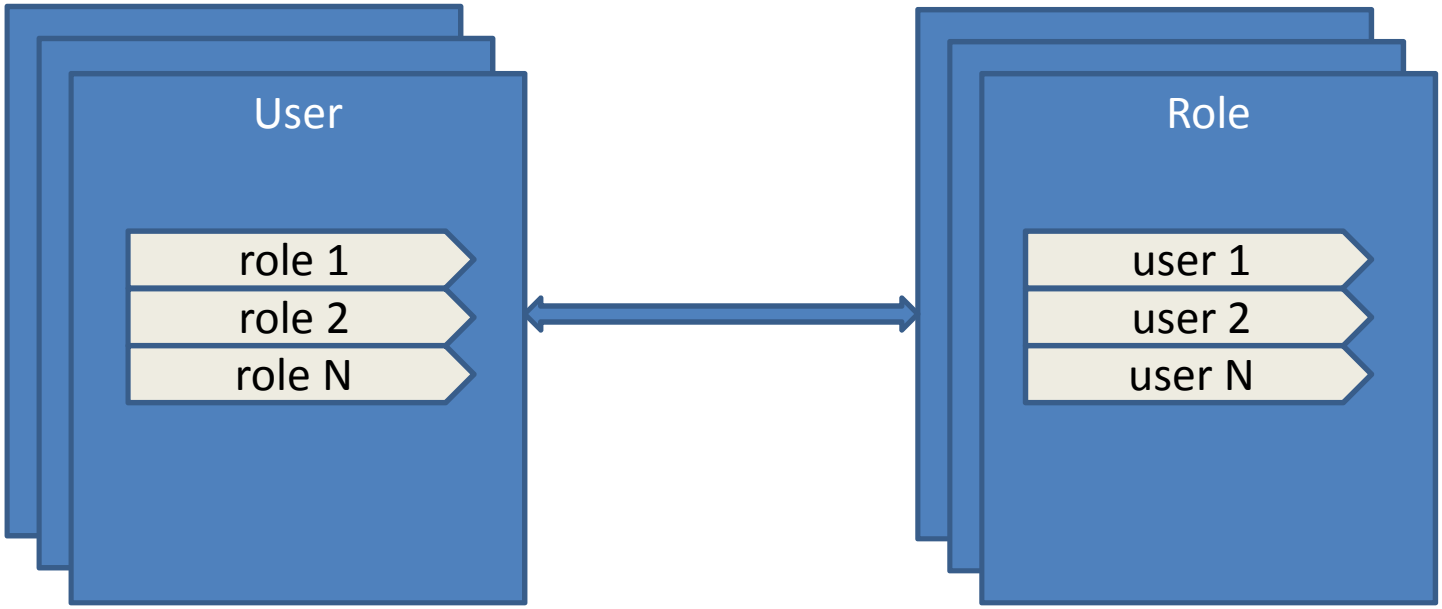
Links

Embedded objects

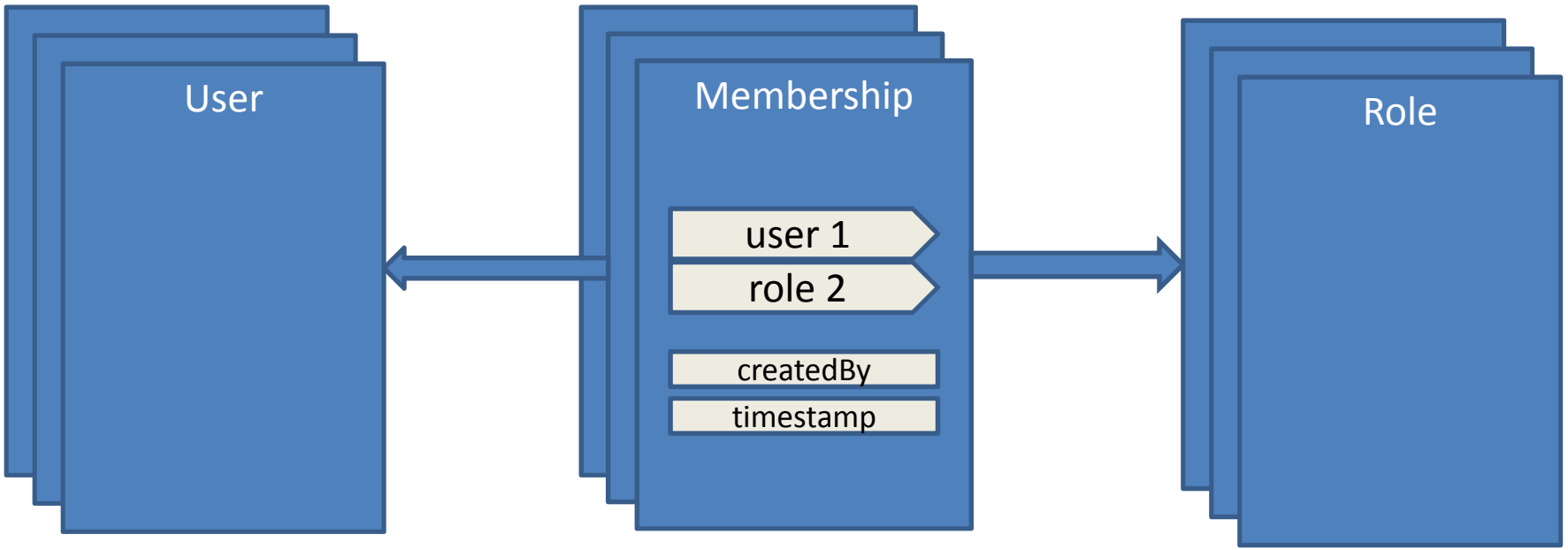
State

```
{
  "_links": {
     "self": { "href": "/tasks"},
     "next": { "href": "/tasks?page=2"},
     "find": { "href": "/tasks?user={user_id}", "templated"="true"},
  },
  "_embedded": {
     "tasks": [{
         "_links": {
            "self": { "href": "/tasks/g4h3v5"},
            "resource": { "href": "/resources/472538"},
            "user": {"href": "/users/a1b2c3"}
         },
         "propagationMode": "ONE_PHASE",
         "subjectType": "role",
         "startTime": "2014-04-10T08:13:23:331Z",
         "endTime": "2014-04-10T08:24:03:445Z",
         "status": "finished",
      },
      ...
     ]
  },
  "finished": 15,
  "failed": 0
}
```

Relations: Many To Many



Relations as Resources



Errors

- Choose appropriate HTTP status code
- Set short error code for automatic processing
- Provide informative and descriptive entity-bodies
- Use JAX-RS ExceptionMappers

Errors: Code Mapping

1. Decide is it client or server problem (4xx/5xx)
2. Look into HTTP status code spec and select approprite one:

- Entity Not Found -> 404 Not Found / 303 See Other
- Entity Already Exist -> 409 Conflict
- IllegalArgumentException -> 400 Bad Request
- Error (OutOfMemory) -> 503 Service Unavailable / 500

Errors: HTTP Response

404 Not found

X-Application-Error-Code: *EntityNotFound*

X-Application-Error-Info: entity=user,id=a1b2c3

{

A user ‘a1b2c3‘ is not found in Syncope storage. Check if user name is correct. Refer following link for the details: https://cwiki.apache.org/confluence/pages/viewpage.action?pageId=30751185/

}

Errors: ExceptionMappers


```
@Provider
public class RestServiceExceptionMapper implements
      ExceptionMapper<SyncopeClientException> {

    @Override
    public Response toResponse(final SyncopeClientException ex) {
        LOG.error("SyncopeClientException thrown by REST method: " +
                ex.getMessage(), ex);

        builder = ex.isComposite() ?
                getSyncopeClientCompositeExceptionResponse(ex.asComposite())
                : getSyncopeClientExceptionResponse(ex);

        return builder.build();
    }
}
```

Asynchronous Processing

- Model operations taking a long time
- Provide non-blocking calls on the client side
- Provide suspended responses on the server side

Asynchronous: Long Operations


```
POST /tasks HTTP/1.1
{
 "propagationMode": "TWO_PHASES",
 "resource": { "href": "/resources/98712" }
 "status": "NONE",
  …
}

202 Accepted
Location: /tasks/x7h3b4


GET tasks/x7h3b4
{
 "propagationMode": "TWO_PHASES",
 "resource": { "href": "/resources/98712" }
 "status": "IN_PROGRESS",
  …
}
```

Asynchronous: Client API

```
InvocationCallback<Response> callback =
     new InvocationCallback {

  public void completed(Response res) {
    System.out.println("Request success!");
  }

  public void failed(ClientException e) {
    System.out.println("Request failed!");
  }
};

client.target("http://mysyncope.org/tasks")
  .request()
  .async()
  .post(myEntity, callback);
```

Asynchronous: Server API

```
@Path("/connectors")
public class AsyncResource {
    @GET
    public void asyncGet(@Suspended final AsyncResponse asyncResponse) {

        new Thread(new Runnable() {
            @Override
            public void run() {
                String result = readConnectors();
                asyncResponse.resume(result);
            }

            private String readConnectors() {
                // ... very expensive operation
            }
        }).start();
    }
}
```

Transactions

```
/tasks/f3g4n5
{
  “userFilter“: “age<=16“
}

/tasks/l8b3n7
{
 “userFilter“: “age>16“
}
```

Requirement: update age to 18 in both tasks in transaction

Transactional View

1. Create transaction:

```
POST /transactions/tasks-update
201 Created
Location: /transactions/tasks-update/89d3
```

2. Update transaction resources:

```
PUT /transactions/tasks-update/89d3/tasks/f3g4n5
{
  “userFilter“: “age<=18“
   …
}

PUT /transactions/tasks-update/l8b3n7/tasks/f3g4n5
{
  “userFilter“: “age>18“
   …
}
```

Committed Transaction

3. Commit transaction:

```
PUT /transactions/tasks-update/89d3
committed = true

200 OK
{
   “tasks“: [
       {“ref“:“/tasks/f3g4n5“}
       {“ref“:“/tasks/l8b3n7“}
   ]
}


GET /tasks/f3g4n5
{
  “userFilter“: “age<=18“
  …
}

GET /tasks/l8b3n7
{
 “userFilter“: “age>18“
  …
}
```

Bean Validation: JAX-RS

```
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;
…

@Path("users")
@Consumes("application/json", "application/xml")
@Produces("application/json", "application/xml")
public interface UserService {

 @GET
 PagedResult<UserTO> list(
            @NotNull @Min(1) @QueryParam(PARAM_PAGE) Integer page,
            @NotNull @Pattern(regexp = "\\d+") @QueryParam(PARAM_SIZE)
                                                    Integer size);

 @GET
 @Path("{email}")
 @Valid UserTO getUser(@Email @PathParam("email") String email);

…
}
```

Conclusion

- Try to follow REST and RESTful HTTP principles by design your application
- Consider using JAX-RS implementation for Java applications
- CXF is nice alternative with active, responsive and cooperative community

Links

- Apache CXF :
  http://cxf.apache.org/
  http://cxf.apache.org/docs/jax-rs.html

- Apache Syncope:
  http://syncope.apache.org/

- Blogs:
  http://sberyozkin.blogspot.com
  http://ashakirin.blogspot.de/
  http://aredko.blogspot.de/

JAX-RS 2.1

The Apache Software Foundation
http://www.apache.org/

❑ Filling the Gaps
- Server-Side Events
- MVC

❑ Performance
- Reactive Programming Model
- Java SE 8 Streams
- Non-blocking I/O

❑ JAVA EE
- Declarative Security Model
- CDI

Errors: Batch Operations

207 Multi-Status
X-Application-Error-Code: *Composite*

```
{
  “message“: “Multiple errors detected“
  “errors“: [
    {
      “statusCode“: 409
      “errorCode“: “EntityExists“
      “ errorMessage “ : “User ‘a1b2c3‘ already exists“
    }
    {
     “statusCode“: 404
      “errorCode“: “NotFound“
      “errorMessage“: “User ‘d4e5f6‘ not found“
    }
   …
  ]
}
```

Validation

- JAX-RS 2.0: Bean Validation 1.1 Specification
- Implementation: Hibernate Validator (or Apache BVal)
- Exception mapper maps:

a) Input parameter validation violation -> 400 Bad Request

b) Return value validation violation -> 500 Internal Server Error

Relations as Resources


```
GET users/a1b2c3

HTTP/1.1 200 OK
Content Type: application/linked+json

{
 "href": "/users/a1b2c3",
 "name": "testUser",
 …

 "memberships": {
  "href": "/memberships?userId=a1b2c3"
 }
}
```

OPTIONS

Returns communication options of target resource

```
OPTIONS /users

Response:
200 OK
Allow: OPTIONS,GET,POST
```

Bean Validation: CXF


```
<jaxrs:server address="/">
    <jaxrs:inInterceptors>
        <ref bean="validationInInterceptor" />
    </jaxrs:inInterceptors>

    <jaxrs:outInterceptors>
        <ref bean="validationOutInterceptor" />
    </jaxrs:outInterceptors>

    <jaxrs:serviceBeans>
    ...
    </jaxrs:serviceBeans>

    <jaxrs:providers>
        <ref bean="exceptionMapper"/>
    </jaxrs:providers>
</jaxrs:server>

<bean id="exceptionMapper"
class="org.apache.cxf.jaxrs.validation.ValidationExceptionMapper"/>
<bean id="validationProvider" class="org.apache.cxf.validation.BeanValidationProvider" />

<bean id="validationInInterceptor"
class="org.apache.cxf.jaxrs.validation.JAXRSBeanValidationInInterceptor">
    <property name="provider" ref="validationProvider" />
</bean>

<bean id="validationOutInterceptor"
class="org.apache.cxf.jaxrs.validation.JAXRSBeanValidationOutInterceptor">
    <property name="provider" ref="validationProvider" />
</bean>
```

GET, HEAD

- READ semantic
- Must be safe and idempotent
- Cacheable
- Can be conditional or partional

```
GET /users/a1b2c3
```

DELETE

- DELETE semantic
- Not safe, Idempotent
- Resource doesn‘t have to removed immediately
- Can return the resource representation or other payload

```
DELETE /users/a1b2c3
```

PUT

- Can be used for UPDATE and for CREATE
- Not safe, idempotent
- Not for partial updates

```
PUT /users/a1b2c3
{
“username“:“testUser“
“status“:“active“
…
}
```

POST

- Can be used to do anything (normally create or update)
- Neither safe, no idempotent

```
POST /users
{
“username“:“testUser“
“status“:“active“
…
}
Response:
201 Created, Location=/users/a1b2c3
```

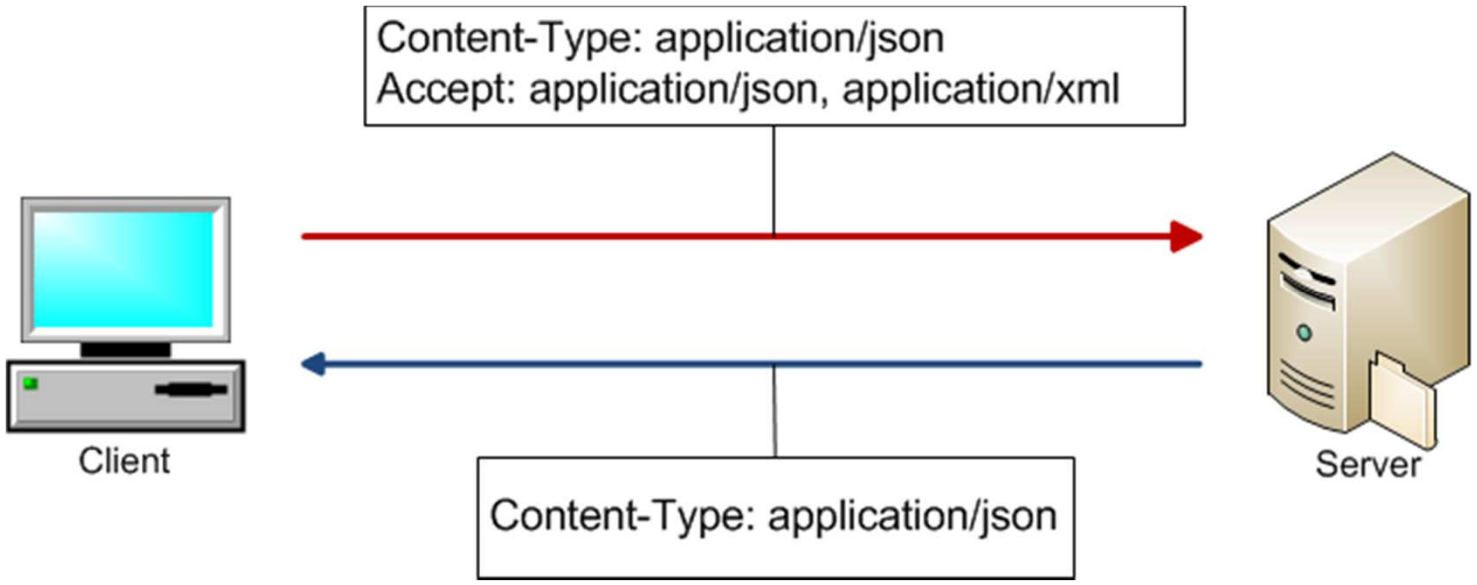
Resources and URIs: Rules

- Resource is anything to be referenced
- Normally resources are the nouns
- Resources are coarse grained

- URIs: descriptive and well structured
- URIs: scoping information

Representations: Media Types

Data Format + Parsing rules



Representations

Extension Mappings:

- /users/a1b2c3
- /users/a1b2c3.json
- /users/a1b2c3.xml