Summary

How to create a CloudIA-Service. Here we will document how a new REST service is created in HIP demonstrated with Resteasy.

Table of Contents

1 Header

Responsible Developer	HeikoRath
Involved Persons	-
Architect	-
Developer	HipBeTeam#Struppi
Quality Manager, Tester	-
IT-Administrator	-

2 Introduction

We will demonstrate samples with our HipRegistry as a reference project. We support the Hosting REST Guidelines, this includes e.g. Hypermedia.
Samples and Annotations are meant to be rendered and used with Resteasy.

3 General

Here you will find adoptions to the Hosting REST Guidelines with samples within our HIP-Services.

3.1 Generic REST Resource Rules

3.1.1 Root Resource speaks Hypermedia

The root resource will return a list of Links to itself and other available resources on this service (child resources), which we usually link as (rel="list")

- e.g. a call on `curl vm1786.development.lan:8180/hip-registry` will return

```
{
    "links": [
        {
            "rel": "self",
            "href": "http://vm1786.development.lan:8180/hip-registry"
        },
        {
            "rel": "list",
```

```
                "href": "http://vm1786.development.lan:8180/hip-registry/bundles"
            }
        ]
    }
```

Currently this is made via a own `RootResource` Object which contains a List of Links. This shall be rendered dynamically in the future when we have the correct toolset for it.

3.1.1.1 Reference of rel attributes in HIP

Sources:

- *http://www.w3.org/TR/1999/REC-html401-19991224/struct/links.html#h-12.1.2*
- *http://www.iana.org/assignments/link-relations/link-relations.xml*
- *http://www.w3.org/TR/1999/REC-html401-19991224/types.html#type-links*

Link Type	HIP Description	HIP Sample	Source Description	State
Alternate	Alternate Version of the current Resource	.../vnd.*-v1 / ...vnd.*-v2	Designates substitute versions for the document in which the link occurs. When used together with the lang attribute, it implies a translated version of the document. When used together with the media attribute, it implies a version designed for a different medium (or media).	is it possible to relate to alternatives? e.g. getting v3 instead of current v4 via URI
`collection`	single Resource points to its collection	`listResource/{resource} => listResource`	The target IRI points to a resource which represents the collection resource for the context IRI.	other samples: `bundles/bundle => bundles, sessions/session => N.A.`

Link Type	HIP Description	HIP Sample	Source Description	State
current	Latest Version of a Resource, if applicable		A URI that, when dereferenced, returns a feed document containing the most recent entries in the feed.	see Alternate, it is not possible to link to the latest without a header
first			An IRI that refers to the furthest preceding resource in a series of resources.	first and last can be iterated via index of paging / next link
last			An IRI that refers to the furthest following resource in a series of resources.	
`next`	Forward iteration of a list resource	`...listResource?page=3&per_page=10"`	Indicates that the link's context is a part of a series, and that the next in the series is the link target.	
`prev`	Backward Iteration of a list resource	`...listResource?page=1&per_page=10"`	Indicates that the link's context is a part of a series, and that the previous in the series is the link target.	

Link Type	HIP Description	HIP Sample	Source Description	State
related	e.g. provisioningItemsresource e.g. by contract, customer, etc	`...context1/provItems => ...context2/provItems`	Identifies a related resource.	if there are related resources available, maybe the architecture isnt that good?
`search`	Provides information how to filter a collection resource	`listResource?matrixParamName={someValue}`	Refers to a resource that can be used to search through the link's context and related resources.	
`self`			Conveys an identifier for the link's context.	
`up`	link to the next available parent resource	`sessions/session/namespaces/namespace => sessions/session`	Refers to a parent document in a hierarchy of documents.	

Additionally: As it is impossible to iterate through childResources they can be named as the resource/artifiact sees it necessary. E.g a Bundle resource can link to a list of services via the

`rel="services"`

or

`rel="serviceList"`

3.1.2 Resources are notated in plural:

- Correct, list Resource: `/hip-registry/bundles` (not correct: `/hip-registry/bundle`)
- Correct, single Resource: `/hip-registry/bundles/{bundleId}` (not correct: `/hip-registry/bundle/{bundleId}`)
- Correct, action Resource: `/hip-testservice/tesresources/{testid}/dosomeactions`

This is basically Annotated with the `@Path` Annotation

For Responses of Type List, e.g. `List` you need to add a `@Wrapped` Annotation in Resteasy, otherwise the Representation in XML will be surrounded by the Element Collection

3.1.3 Headers

Headers which have to be supported are:

- `Accept` The Client wants a Resource Representation in a specific Structure (Usually `json` or `xml`)
  - Annotated by `@Produces`
  - Sample for the Hip Bundle List: `@Produces({HIP_REGISTRY_BUNDLE_LIST_JSON, HIP_REGISTRY_BUNDLE_LIST_XML, MediaType?.APPLICATION_JSON, MediaType?.APPLICATION_XML})`

- `Content-Type` The Client wants to store something (Entity-Body) on the Server (PUT/POST)
  - Annotated by `@Consumes` The Server accepts this List of Vendor Types as Body
- `Vary` The Server can return the Resource Representation in different ways, and the client can choose via these values
  - e.g. `Accept` (via the `Accept` Header the client can choose between `JSON` and `XML`)
  - In Resteasy we wrote a MessageBodyWriterInterceptor[?] called `VaryHeaderWriter` which injects this into the Response Headers, check the `com.oneandone.hosting.hip.registry.framework` Package within the Hip-Registry for Sources.

3.1.3.1 Vendor Types

- All Resources have a versioned, own Vendor Type and support `XML` and `JSON`
- All Resources support the Default `application/json` and `application/xml` formats **Sample for the List of Bundles Resource:**

```
application/vnd.oneandone.hip.registry.bundle.list-v1+json
application/vnd.oneandone.hip.registry.bundle.list-v1+xml
application/json
application/xml
```

3.2 Response Handling

As all RESTful services shall respond in similar ways, some things should be kept in mind.
e.g.:

- If a Resource is not found return HTTP 404 (Not Found), sample: `/hip-registry/bundles/NotExistentBundleId`
- If a Resource is not found on a List Resource (usually using filters via matrixParams) return a HTTP 200 (OK) with an empty List, sample `/hip-registry/bundles;matrixFilterKey=matrixValue`

The Resteasy Framework provides us with some ExceptionMappers[?] so we dont have to write them ourself. E.g the `NotFoundException` should be thrown when the Resource is found via its URL, but not within the system.

Attention: DO NOT catch ALL Exceptions otherwise the Resteasy Framework cannot help you with its full potential, e.g. the MethodNotAllowed[?] Exception with its correct Header will not work anymore

3.2.1 ExceptionMapper[?]

Before writing a custom Exception and ExceptionMapper[?] check this offical list: http://docs.jboss.org/resteasy/docs/2.3.1.GA/userguide/html_single/index.html#ExceptionHandling of Exceptions provided by RESTeasy. These are provided and handled by RESTeasy out of the box.

For Exceptionmapping to special Response-Codes you can simply create ExceptionMapper[?] by implementing the ExceptionMapper[?]-Class and annotate your ExceptionMapper[?]-class with Spring's @Provider. It will be found because of the include-filter in the spring configuration (**springramework:context:component-scan**).

Here is a list of Mappers which RESTEasy cannot handle out of the box:

- Illegal Arguments (as this is no REST problem, but a Java one)

```
@Provider
public class IllegalArgumentExceptionMapper implements ExceptionMapper<IllegalArgumentException> {

    /**
     * {@inheritDoc}
     */
    @Override
    public Response toResponse(IllegalArgumentException e) {

        ResponseBuilder responseBuilder = Response.status(Response.Status.BAD_REQUEST.getStatusCode());
        responseBuilder.header("Content-Type", "text/plain; charset=utf-8");
```

```
            return responseBuilder.entity(e.getMessage()).build();

        }
    }
```

- OAuth2 specific: AccessDeniedException
- Oauth2 specific: InvalidTokenException
- RemoteInfrastructureException (this one is a custom HIP exception, to reflect that a required Service, called by HIP is not working as specified, e.g. Comet had a internal error)

4 RESTEasy specifics

4.1 Dependencies

In order to use RESTeasy, add the hip-commons-resteasy dependency:

```
<dependency>
    <groupId>com.oneandone.hosting.hip</groupId>
    <artifactId>hip-commons-resteasy</artifactId>
</dependency>
```

hip-commons-resteasy contains all hip custom exception mappers and all hip MIME type providers.

hip-commons-resteasy uses the following dependencies:

where `${resteasy.version}` is currently 2.3.2.Final

```
<dependency>
      <groupId>org.jboss.resteasy</groupId>
      <artifactId>resteasy-spring</artifactId>
      <version>${resteasy.version}</version>
      <exclusions>
        <exclusion>
          <groupId>org.jboss.resteasy</groupId>
          <artifactId>resteasy-jettison-provider</artifactId>
        </exclusion>
      </exclusions>
    </dependency>
    <dependency>
      <groupId>org.jboss.resteasy</groupId>
      <artifactId>resteasy-jackson-provider</artifactId>
      <version>${resteasy.version}</version>
    </dependency>
    <dependency>
      <groupId>org.jboss.resteasy</groupId>
      <artifactId>resteasy-jaxb-provider</artifactId>
      <version>${resteasy.version}</version>
    </dependency>
```

We don't use Jettison as a Json representation format provider, because it doesn't returns the needed Json representation format! Instead Jackson is used, so we have to exclude the Jettison provider explicitly!

If you need Jettison e.g. for parsing Json strings, never use the resteasy-jettison-provider. Instead add the Jettison dependency directly, which doesn't act as a Json representation format provider:

```
<dependency>
```

```
   <groupId>org.codehaus.jettison</groupId>
   <artifactId>jettison</artifactId>
</dependency>
```

Using the Jettison provider and the Jackson provider leads to not predictable Json representation formats, because you never no which one is used!

4.2 web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
                             http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

  <display-name>hip-registry</display-name>

    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:applicationContext.xml</param-value>
    </context-param>

    <listener>
        <listener-class>
            org.jboss.resteasy.plugins.server.servlet.ResteasyBootstrap
                </listener-class>
    </listener>

    <listener>
        <listener-class>
                        org.jboss.resteasy.plugins.spring.SpringContextLoaderListener
                </listener-class>
    </listener>

  <servlet>
    <servlet-name>Resteasy</servlet-name>
    <servlet-class>
      org.jboss.resteasy.plugins.server.servlet.HttpServletDispatcher
    </servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>Resteasy</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>


  <servlet>
    <servlet-name>ServerCheck</servlet-name>
    <servlet-class>de.web.common.monitoring.check.servlet.SpringServerCheckServlet
    </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ServerCheck</servlet-name>
    <url-pattern>/-system/check</url-pattern>
  </servlet-mapping>
```

```
</web-app>
```

4.3 applicationContext.xml

```
...
    <context:component-scan base-package="com.oneandone.hosting.hip.registry.*" >
        <context:include-filter type="annotation" expression="javax.ws.rs.ext.Provider"/>
    </context:component-scan>
...
```

4.4 RestEasy?-Impl class

Simply annotate your RestEasy? class with Spring's @Component and then inject your beans using Spring's @Autowired. Don't forget to include the annotation-config and component-scan elements in your spring configuration.

5 Research / Work in progress

1. Current:

```
{
  "Bundle" :
    {
      "bundleId" :
       {
          "bundleId" : "sampleId"
       },
      "componentList" : "",
      "contact" : "sampleContact",
      "description" : "sampleDescription",
      "monitoringList" : "",
      "responsible" : "sampleResponsible",
      "serviceList" : ""
    }
}
```

2. Goal:

```
{
  "links" : {
      "self" : { "href" : "LinkToMyself" },
      "monitorings" : { "href" : "LinkToMyself/monitorings" },
      "services" : { "href" : "LinkToMyself/services" },
      "components" : { "href" : "LinkToMyself/components" }
   },

  "Bundle" :
    {
      "bundleId" : "sampleId",
      "componentList" : "",
      "contact" : "sampleContact",
      "description" : "sampleDescription",
```

```
    "monitoringList" : "",
    "responsible" : "sampleResponsible",
    "serviceList" : [
      {
          "bundleId" : "sampleId",
          "serviceName": "sampleServiceName",
          "intEndpoint": "http://vm1786.development.lan:8080/hip-sessionstore-1.0",
          "extEndpoint": "http://vm1786.development.lan:8080/hip-sessionstore-1.0",
          "namespace": "hip-sessionstore",
          "metadataEntries": {
              "metadataKey": "responsible",
              "metadataValue": "HCS"
          }
      },
      {
          "bundleId" : "sampleId",
          "serviceName": "hip-authorizationservice",
          "intEndpoint": "http:\/\/vm1786.development.lan:8080\/hip-authorizationservice-0.7",
          "extEndpoint": "http:\/\/vm1786.development.lan:8080\/hip-authorizationservice-0.7",
          "namespace": "hip-authorizationservice",
          "metadataEntries": {
              "metadataKey": "responsible",
              "metadataValue": "HCS"
          }
      }
    ]
  }
}
```


6 Mapping of Exceptions to HTTP ErrorCodes?

The following list of business cases show how we map Exceptions to HTTP error codes

6.1 Business Case to ErrorCodes? Concrete samples

6.1.1 HIP-SessionStore

Business Case	Exception	HTTP Error Code	Abstract REST case	Comment
DELETE a non-existing key		404	DELETE on missing resource	
DELETE key from a non-existing namespace	NamespaceNotExistsException?	404	DELETE on missing resource	currently 412
DELETE key/value from a non-existing session	SessionNotExistsException?	404	DELETE on missing resource	currently 412
GET with malformed KeyEntryList? (payload)		400	Request is syntactically wrong	

Business Case	Exception	HTTP Error Code	Abstract REST case	Comment
POST malformed EntryList? (payload)		400	Request is syntactically wrong	
GET KeyEntryList? contained missing key (returned less results than requested)		206	Only subset of requested resources found on server	
GET on namespace without KeyEntryList?		400	Request is missing critical information	
PUT into non-existing namespace		404	Parent resource in URI was not found	
PUT into non-existing session		404	Parent resource in URI was not found	

6.1.2 HIP-Registry

BusinessCase	Exception	HTTP Error Code	Abstract REST case	Comment
GET Bundle with non-existent bundleId in URI		404	Parent resource in URI was not found	
GET services/components/monitoring with non-existent bundleId in URI		404	Parent resource in URI was not found	
POST Bundle with BundleID? in payload		400	Including tech. ID where server generates ID	
POST services/components/monitoring into non-existing bundleId		404	Parent resource in URI was not found	
PUT Bundle without BundleID? or other mandatory field in payload		400	Request is missing critical information	
PUT service/component/monitoring into non-existing BundleId?		404	Parent resource in URI was not found	
DELETE services/components/monitoring from non-existing Bundle		404	Parent resource in URI was not found	
PUT service/component/monitoring with incorrect BundleId? in payload (missmatch from URI)		400	URI and transmitted payload mismatch	
POST service/component/monitoring with relevant s/c/m Id in payload		400	Including ID in payload where server is responsible to generates it	

6.1.3 PIM

BusinessCase	Exception	HTTP Error Code	Abstract REST case	Comment
GET provisioning-items with non-existent contractId		404	Parent resource in URI was not found	

BusinessCase	Exception	HTTP Error Code	Abstract REST case	Comment
GET provisioning-items with non-existent customerId		404	Parent resource in URI was not found	
GET provisioning-items with non-existent accountId		404	Parent resource in URI was not found	

6.2 REST Mapping Business Cases to HTTP Responses

BusinessCase	HTTP Error Code	Generic	Comment	Concrete
The given resource in URI was not found	404	Resource not found		`/staticResource/{parentIdNotFound}/staticChildResource/{requestedId}`
Parent resource in URI was not found	404	Resource not found		`/staticResource/{requestedId}`
The given resource differs from values in the payload	400	data missmatch		`/staticResource/{requestedId} payload: {object: {id: anotherRequestedId}}`
The given resource differs from values in the payload	400	data missmatch	e.g POST	`/staticResource payload: {object: {id: willBeGeneratedId}}`
The payload does not fulfill the requirements	400	Mandatory data is missing		`payload: {object: {id: null}}`

7 List of sources

This How to is based on following sources:

- http://developers.1and1.com/hosting/guidelines/rest/
- settings.xml: settings.xml

Attachment	Action	Size	Date	Who	Comment
settings.xml	manage	8.3 K	18 Jul 2012 - 12:42	FrankStiller	

This topic: Devhost > WebHome > HostingIntegrationPlatform > HipBackEnd > HipBeHowTo > HowToRestService
History: r21 - 14 Aug 2012 - 05:50:06 - KaiHofstetter