Tutorial Rest

v. 0.1.9-SNAPSHOT

Project Documentation

UniKnow **2015-05-07**

Table Of Content

1 Introduction

RESTful Web services

Within this tutorial we will create a simple restful blogging system.

When providing services in a specific technology (jax-ws, jax-rs, …), it's wise to avoid mixing the business logic of your service and the technology you are using to expose it to the outside world. This separation facilitates easier adaption of newer technologies and makes it easier to maintain and test services.

The project is separated in 2 packages.

1. api - defining the interfaces that are exposed
2. impl - Implementation of the interfaces

The `api` package contains the business interface `BlogService` within the package `org.uniknow.agiledev.tutorial.rest.api`, and the technology specific interface `BlogRestService` within the ' `jaxrs` package containing JAX-RS annotations. The `jaxrs` package also contains the DTO (Data Transfer Objects) classes required to provide and/or consume the REST services.

1.1 Versioning

Versioning of the REST service will be done via custom `Accept` headers containing custom media types (such as `application/vnd.agiledev.blog.v1+xml`) instead of using one of the defined media types. Within the `BlogRestService` the custom consume and produce media types are defined

```
/**
 * Version 1 of Blog Rest Service
 */
package org.uniknow.agiledev.tutorial.rest.api.jaxrs.V1;

@Path("/blog")
@Consumes({ "application/agiledev.blog.v1+xml",
        "application/agiledev.blog.v1+json" })
@Produces({ "application/agiledev.blog.v1+xml",
        "application/agiledev.blog.v1+json" })
public interface BlogRestService {
    ...
}
```

When the entities change in a non compatible way (V2), a new version of the API can co-exist with the first version, given that the entity/DTO namespace is also changed. The newer API will have its own interface and use an updated media type:

```
/**
 * Version 2 of Blog Rest Service
 */
package org.uniknow.agiledev.tutorial.rest.api.jaxrs.V2;

@Path("/blog")
@Consumes({"application/agiledev.blog.v2+xml", "application/agiledev.blog.v2+json"
@Produces({"application/agiledev.blog.v2+xml", "application/agiledev.blog.v2+json"
```

and the namespace would become something like this (`package-info.java`):

```
@XmlSchema(namespace = "http://www.uniknow.org/rest/blog/api/v2",
    elementFormDefault = XmlNsForm.QUALIFIED, xmlns = { @XmlNs(prefix = "api",
        namespaceURI = "http://www.uniknow.org/rest/blog/api/v2") })
package org.uniknow.agiledev.tutorial.rest.api.jaxrs.V2;

import javax.xml.bind.annotation.*;
```

Now the client is required to specify an `Accept` header when using the API. Whe can remedy this by adding default media types to the latest version of our API. This way clients invorking the API without an `Accept` header will be working with the latest version.

@Path(“/blog”) @Consumes({ MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON, “application/agiledev.blog.v2+xml”, “application/agiledev.blog.v2+json” }) @Produces({ MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON, “application/agiledev.blog.v2+xml”, “application/agiledev.blog.v2+json” })

1.2 Running the example application

To build and run the application perform:

```
mvn clean install cargo:run
```

Once the application is running the Rest services can be invoked by using for example the URL http://localhost:8080/rest/api/blog/posts and include header param `Accept` with for example value `application/agiledev.blog.v2+json`. If used without `Accept` header the response will be of content type `application/xml`.

Resources:

- http://codebias.blogspot.cz/2013/03/how-to-restful-web-services-with-jax-rs.html
- http://codebias.blogspot.nl/2014/03/versioning-rest-apis-with-custom-accept.html
- http://esofthead.com/restful-application-spring-resteasy/