Web service feature

needs resource based uris

http methods

response http status codes

message headers

Hateous

To provide usesul links to resources in the api response

Richar Maturity Level

0 no rest

1 individual resource uri

2 http methods and status codes

3 Hateous

Jax Rs

Common api to work with Rest Web Services (Interfaces and Annotations)

Jersey, Rest Easy implementations of Jax Rs

Resource

A resource is an object with a type, associated data, relationships to other resources, and a set of methods that operate on it.

Config of jersey

There is a servlet configure, listening to a path and then choose the correct resource to handle the request.



@Path

Allows to specify the uri for a resource to execute

@Get

Associates a method to the http get method.

@Produces

Return content type which will be return by the resource

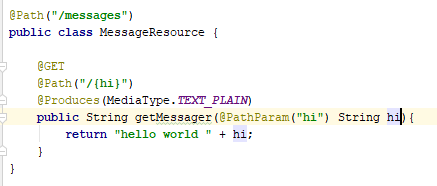
Resource Example



@PathParam

Allows to take a path param and use it in the resource





RequestBody receives the data and mapped into a java object



Message Body Writer Interface

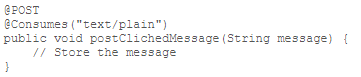
Necessary libs to convert from java to other format(xml, json). Ex convert from java message to json

Message Body Reader interface

Necessary libs to convert from any given format to java

@Consumes

is used to specify which MIME media types of representations a resource can accept



@QueryParam

Allows to take parameters sent and use them in the resource





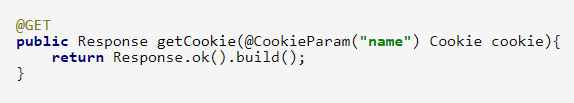
@HeaderParam

We can fetch header parameters

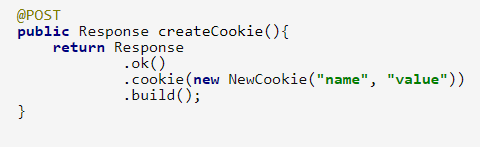


@CookieParam

We can retrieve the entire cookie by injecting the Cookie with the @CookieParam JAX-RS annotation



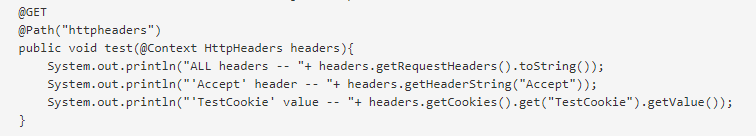
Create cookie on the response

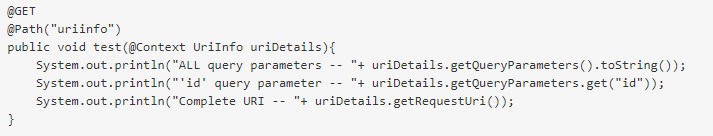


@Context

inject a variety of resources in your RESTful services. Some of the most commonly injected components are HTTP headers, HTTP URI related information.

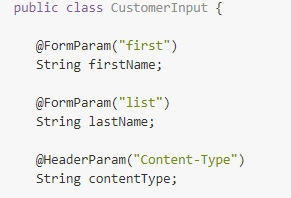
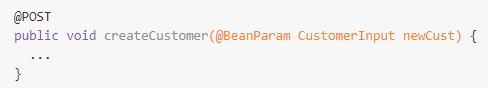






@BeanParam

Allows to create a bean, replace a whole lot of the other param annotations

(will receive form params first and list and header param content type, with that creates a CustomerInput object with the @BeanParam

Subresource

RootResource delegate the responsibility to a sub resource.

Subresource gets all parameters from the root resource

Root Resource

@Path(“{messageId}/comments”)

Public CommentResource getCommentResource(){

Return new CommentResource();}

Sub resource

@Path(“/”) //its already defined in the root

Public class CommentResource(){

@Get

Public String test(){

Return “test”;}

@XmlTransient

Allows to ignore a property from that resource

Return Response

Instead of returning an object, we can return a response object which can include headers, status code, entities, etc.



Return Response.created(uri)

.entity(newMessage).build();

@Provider

Registers the class to Jax rs for them to know its there and can be used

An application can extend JAX-RS runtime by implementing one or more interfaces as specified by the JSR-339. These implementations may be annotated with @Provider for automatic discovery during scanning phase

Web application Exception

Jax-rs set of exceptions, maps based on status in the response

Accept header

Can be added in the request, this is the content type it needs. If the class doesn’t support it then it will throw an exception

Param converters

Jersey includes them, to convert String to primitive

Param Converter Provider

Says to jax rs to use that paramconveter for a custom data type

Param Converter

When you need to convert a String to a custom datatype, it is need to create the class which implement param converter and register it on the param converter provider

@Provider

Public class MyDateConverterProvider implements ParamConverterProvider{

@Override

Public <T> ParamConverter<T> getConverter(….

….

MessageBodyReader, MessageBodyWriter

Convert the raw content from the request to a java type

Param Converter

Convert params to custom data types

Filters

For cross cutting concerns on the resources, for header information and metadata only

ContainerRequestFilter, before the request goes to the resource

ContainerResponseFilter, after the request finish on the resource

Needs to implement what you need or both

@Provider

Public class PoweredByFilter implements ContainterResponseFilter{

@Override

Public void filter(….

….}

Filter executes no matter if it was a success or error request.

restApi Stateless

do not maintain information from previous request

Basic Auth

Digest Access Authentication

Asymmetric

OAuth

Interceptors

Similar to filters but to manipulate input and output streams (filter only headers and metadata)

ReaderInterceptor modify request

WriterInterceptor modify response