**More About Root Resource Class**

* A Root Resource Class
  + Can be Standalone/Independent Class

(i.e dosen’t extend/implements any other class/interface)

* + can “implements” One/More Interfaces
  + can “extends” an another Class
  + can “extends” an another Class and “implements” One/More Interfaces.
* So far, we’ve applied JAX-RS annotations directly on the “Independent Root Resource Class”
* In JAX-RS, we are also allowed to define a Java interface that contains all out JAX-RS annotation metadata instead of applying all our annotations to our implementation class.

EX:

import…

public interface FileDownloadService {

@GET

@Path(“/image”)

@Produces(“image/png”)

public Response getImageFile();

}

* With an interface, we can define web service methods that gives high level picture on how our RESTful services looks like without exposing implementation details.
* Interfaces are a great way to isolate all this metatdata into one logicall and readable construct
* Hence they helps us to achieve readability because business logic isn’t “polluted” with so many annotations.
* The implementation class looks like

@Path(“/file/download”)

public class FileDownloadServiceImpl implements FileDownloadService {

public Response getImageFile(){

…

}

}

NOTE:

* @Path SHOULD be present at class level not at interface level
* Hence, Interfaces helps us to design our services based on our need (with more than one implementation class with different @Path)
* As we can see, except @Path, no other JAX-RS annotations are needed within the implementing class. All our metadata is confined to FileDownloadService interface.
* If we want to, we can override the metadata defined in our interface by reapplying annotations within our implementation class.

@Path(“/file/download”)

public class FileDownloadServiceImpl implements FileDownloadService

{

@GET

@Path(“/img”)

@Produces(“image/png”)

public Response getImageFile(){

…

}

}

* In the above example, we are overriding the metadata defined in an interface for one specific method. “When overriding metadata for a method, you must specify all the annotation metadata for that method even if you are changing only one small thing”
* However, it is not recommend to override metadata in implementation class.
* The whole point of using an interface to apply our JAX-RS metadata is to isolate the information and define it in one place. If annotations are scattered about between implementation class & interface, then code becomes a lot harder to read and understand
* It’s also possible that root resource class can have super class (Abstract & Concrete)
* If Super Class is an Abstract Class then it SHOULD NOT have @Path at class level.