**HttpHeaders (or) Programmatic Access to Headers and Cookies (or) Raw Headers and Cookies:**

Using @HeaderParam or @CookieParam we can access only one heeder param or one cookie param and if we want we can write multiple @HeaderParam or @CookieParam annotations with multiple method params, instead we can use HttpHeaders to get all multiple headers.

Sometimes we need programmatic access to view all headers within the incoming request. For instance, we may want to log them. The JAX-RS specification provides the javax.ws.rs.core.HttpHeaders interface for such scenarios.

HttpHeaders is an interface that is provide by the JAX-RS API to get more info about Headers and Cookies.

We are calling programmatic approach as Raw Headers or Raw Cookies approach bcz using HttpHeaders we inject as String type only but not in the business data types.

We can obtain a map of all cookies sent by the client by injecting a reference to javax.ws.rs.core.HttpHeaders.

interface HttpHeaders {

public List<String> getRequestHeader(String name);

public MultivaluedMap<String, String> getRequestHeaders();

public Map<String, Cookie> getCookies();

...

}

getRequestHeader(String) method allows you to get access to one particular header.

getRequestHeaders() gives you a map that represents all headers.

getRequestHeaders()-used to get all the ReqHeaders (including headers and Cookies).

getCookies()-used to get only all the cookies.

If we need more information about the cookie other than its base value, then instead of @Cookie we can inject a javax.ws.rs.core.Cookie object. The Cookie class has additional contextual information about the cookie beyond its name and value.

package javax.ws.rs.core;

public class Cookie {

public String getName() {...}

public String getValue() {...}

public int getVersion() {...}

public String getDomain() {...}

public String getPath() {...}

...

}

getName() and getValue() methods correspond to the string name and value of the cookie you are injecting.

getVersion() method defines the format of the cookie header—specifically, which version of the cookie specification the header follows.

getDomain() method specifies the DNS name that the cookie matched.

getPath() method corresponds to the URI path that was used to match the cookie to the incoming request. All these attributes are defined in detail by the IETF cookie specification.

We can use the @Context annotation to get access to HttpHeaders.

@Path("/cab-mgmt")

public class CabResource {

@GET

@Produces(MediaType.TEXT\_PLAIN)

@Path("/dynamictrips")

public String getTrips(@Context HttpHeaders httpHeaders) {

// Get All the ReqHeaders which contains headers and cookies

MultivaluedMap<String, String> allHeaders =

headers.getRequestHeaders();

// Accessing one particular header

String referer = headers.getRequestHeader("Referer").get(0);

// Get Only All the cookies separately

Map<String, Cookie> allcookies = headers.getCookies();

// Alternatively get the Only All the Cookies Separately

for (String name : httpHeaders.getCookies().keySet()) {

Cookie cookie = httpHeaders.getCookies().get(name);

String cookieName=cookie.getName();

String cookieValue=cookie.getValue();

}

}

}

Access the application:

http://localhost:8080/12.6HttpHeaders/resources/cab-mgmt/dynamictrips

Select ADD HEADERS option and send multiple headers as follows

cabID=C121

driverMobile=9929292929

driverName=john

Response:

All the ReqHeaders (Headers & Cookies)...

host=localhost:8080;connection=keep-alive;

drivermobile=9929292929;

drivername=john;

user-agent=Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.110 Safari/537.36;cabid=C121;accept=\*/\*;

accept-encoding=gzip, deflate, sdch;

accept-language=en-US,en;q=0.8;cookie=driverID=DR143

Getting only Cookies ....driverID=DR143