Thinking in Java (at least trying to!)
Write Once, Read Forever ;-)

Using @Context in JAX-RS [part 1]

Posted on May 3, 2015

JAX-RS provides the ***@Context*** annotation to inject a variety of resources in your RESTful services. Some of the most commonly injected components are HTTP headers, HTTP URI related information. Here is a complete list (in no specific order)

- HTTP headers
- HTTP URI details
- Security Context
- Resource Context
- Request
- Configuration
- Application
- Providers

Lets look at these one by one with the help of examples

HTTP headers

Although HTTP headers can be injected using the @HeaderParam annotation, JAX-RS also provides the facility of injecting an instance of the ***HttpHeaders*** interface (as an instance variable or method parameter). This is useful when you want to iterate over all possible headers rather than injecting a specific header value by name

```
@Path("testinject")
public class InjectURIDetails{
    //localhost:8080/<root-context>/testinject/httpheaders
    @GET
    @Path("httpheaders")
    public void test(@Context HttpHeaders headers){
        System.out.println("ALL headers -- "+ headers.getRequestHeaders().toString());
        System.out.println("'Accept' header -- "+ headers.getHeaderString("Accept"));
        System.out.println("'TestCookie' value -- "+ headers.getCookies().get("TestCookie").getValue());
    }
}

```

InjectHeaders.java hosted with ❤ by **GitHub** **view raw**

Follow

HTTP URI details

UriInfo is another interface whos y JAX-RS (as an instance variable or method parameter). Use this instance to ed to the request URI and its parameters (query, path)

Follow "Thinking in Java (at least trying to!)"

Get every new post delivered to your Inbox.

Join 295 other followers

Enter your email address

Sign me up

Build a website with WordPress.com

```
@Path("testinject")
public class InjectURIDetails{
  //localhost:8080/<root-conte
  @GET
  @Path("uriinfo")
  public void test(@Context Ur
      System.out.println("ALL                                   s.getQueryParameters().toString());
      System.out.println("'id' query parameter -- "+ uriDetails.getQueryParameters.get("id"));
      System.out.println("Complete URI -- "+ uriDetails.getRequestUri());
  }
}
```

InjectURIDetails.java hosted with ❤ by GitHub view raw

Providers

An instance of the Providers interface can be injected using @Context. One needs to be aware of the fact that this is only valid within an existing provider. A Providers instance enables the current Provider to search for other registered providers in the current JAX-RS container.

Note: Please do not get confused between Provider and Providers.

Provider

- A JAX-RS Provider is a is generic term for any class which supplements/extends the JAX-RS features by implementing standard interfaces exposed by the JAX-RS specification
- It is annotated using the @Provider annotation for automatic discovery by the run time
- Examples of JAX-RS providers are – Message Body Reader, Message Body Writer, Exception Mapper and Context Providers.

Providers

Refers to the (injectable) javax.ws.rs.ext.Providers interface which was discussed in this sub section

Security Context

Inject an instance of the javax.ws.rs.core.SecurityContext interface (as an instance variable or method parameter) if you want to gain more insight into identity of the entity invoking your RESTful service. This interface exposes the following information

- Instance of java.security.Principal representing the caller
- Whether or not the user if a part of a specific role
- Which authentication scheme is being used (BASIC/FORM/DIGEST/CERT)
- Whether or not the request invoked over HTTPS

```
@Path("testinject")
public class InjectSecurityContext{
  //localhost:8080/<root-context>/testinject/securitycontext
  @GET
  @Path("securitycontext")
  public void test(@Context SecurityContext secContext){
      System.out.println("Caller -- "+ secContext.getUserPrincipal()getName());
      System.out.println("Authentication Scheme -- "+ secContext.getAuthenticationScheme());
      System.out.println("Over HTTPS ? -- "+ secContext.isSecure());
      System.out.println("Belongs to 'admin' role? -- "+ secContext.isUserInRole("admin");
  }
}
```

InjectSecurityContext.java hosted with ❤ by GitHub view raw

That's all for this part. Rest of the injectables will be covered in the next iteration.