JAX-RS: getClasses vs getSingletons

I'm using JAX-RS (Jersey implementation) to build a web service.

The entry point to the web application is a class that extends `Application` and overrides the `getClasses` method. I understand that JAX-RS will create a new instance for each new request.

It is possible to override `getSingleton` instead. My understanding is that this will reuse the same instance for all requests. However, won't this approach (assuming I'm correct) destroy concurrency within the web application? That is, since the same instance is always used, incoming requests will be processed one at a time (FIFO?).

web-services web-applications jersey jax-rs

asked Aug 15 '13 at 14:12

Daniel
50 1 5

1 Answer

No, it won't. Multiple threads in JVM are able to access one instance of a class at the same time (invoke a resource method on your resource in this case). You just need to make sure your resource is thread-safe.

To illustrate the difference between `getClasses()` and `getSingletons()` lets assume we have a `HelloWorldResource` like:

```
@Path("helloworld")
public class HelloWorldResource {

    private volatile int counter = 0;

    @GET
    @Produces("text/plain")
    public String getHello() {
        return "Hello World! " + counter++;
    }

}
```

Multiple invocations of `getHello` resource method would return you:

- in case you register your resource via `getClasses`

```
Hello World! 0
Hello World! 0
Hello World! 0
...
```

- in case you use `getSingletons` to register your resource

```
Hello World! 0
Hello World! 1
Hello World! 2
...
```

edited Jan 23 '14 at 9:32

Pavel Bucek
4,090 17 34

answered Aug 15 '13 at 16:13

Michal Gajdos
7,745 2 21 30

1 One more doubt I have about the signature of these methods. One is returning Set<Object> and other is Set<Class<?>>, why? When both methods are having the same purpose, then their return type should be same. – sHAILU Oct 2 '13 at 3:01

2 "The getSingletons() method returns actual instances that you create yourself within the implementation of your Application class. You use this method when you want to have control over instance creation of your resource classes and providers. For example, maybe you are using Spring to instantiate your JAX-RS objects, or you want to register an EJB that uses JAX-RS annotations." - source: java.dzone.com/articles/putting-java-rest – Fagner Brack Dec 27 '13 at 13:57