INSTITUTO TECNOLÓGICO DE NUEVO LOEN

Carrera:

Lenguajes autómatas 1

Trabajo:

Investigación de jax en java

Profesor:

Juan Pablo Rosas Valdazo

Alumno:

Eduardo Martínez Martínez

15690231

En pocas palabras, la arquitectura JAX-WS es una arquitectura más fácil de entender para el desarrollo de servicios web. En las ofertas de servicios web anteriores, las funcionalidades de las API de JAX-RPC y JAXB anteriores se superponían bastante. Cuando surgió JAXB 1.x después de JAX-RPC, y como la funcionalidad de enlace de datos se hizo más completa con estándares mejorados como XML

Puede utilizar JAX-WS para crear aplicaciones web y servicios web, incorporando la funcionalidad más nueva de servicios web basados en XML.

JAX-WS 2.0 es extremadamente fácil de usar. Este artículo le mostrará cómo crear un servicio web simple utilizando JAX-WS 2.0 con tecnología Java SE 6. Lo primero que necesita es una clase con uno o más métodos que desee exportar como un servicio web:

paquete hola

clase pública funciones de círculo {

  pública doble getArea (doble radio) {

    devuelve java.lang.Math.PI * (r * r);

  }

  pública doble getCircumference (doble radio) {

    devuelve 2 * java.lang.Math.PI * r;

  }

}

Java EE 7 con JAX-RS 2.0 aporta varias características útiles, que simplifican aún más el desarrollo y conducen a la creación de aplicaciones con arquitectura RESTful para Java SE/EE aún más sofisticadas pero livianas a la vez.

oast House es un ejemplo sencillo de JAX-RS 2.0 apto para Java que administra y genera *coffee beans* (componentes de Java). Roast House está representado como un recurso CoffeeBeansResource. El URI "coffeebeans" identifica en forma unívoca el recurso CoffeeBeansResource (ver Listado 1).

```
//...
import javax.annotation.PostConstruct;
import javax.enterprise.context.ApplicationScoped;
import javax.ws.rs.DELETE;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
```

```
import javax.ws.rs.PathParam;
import javax.ws.rs.container.ResourceContext;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.Response;
@ApplicationScoped
@Path("coffeebeans")
public class CoffeeBeansResource {

    @Context
    ResourceContext rc;

    Map<String, Bean> bc;

    @PostConstruct
    public void init() {
        this.bc = new ConcurrentHashMap<>();
    }

    @GET
    public Collection<Bean> allBeans() {
        return bc.values();
    }

    @GET
    @Path("{id}")
    public Bean bean(@PathParam("id") String id) {
        return bc.get(id);
    }

    @POST
    public Response add(Bean bean) {
        if (bean != null) {
            bc.put(bean.getName(), bean);
        }
        final URI id = URI.create(bean.getName());
        return Response.created(id).build();
    }

    @DELETE
    @Path("{id}")
    public void remove(@PathParam("id") String id) {
        bc.remove(id);
    }

    @Path("/roaster/{id}")
    public RoasterResource roaster(){
        return this.rc.initResource(new RoasterResource());
    }
}
```