Laboratorio di Programmazione di Rete – B
esempio REST server

Federica Paganelli

AA 2021-22

Università di Pisa

JAX-RS

- JAX-RS
  - API Java progettata per semplificare lo sviluppo di applicazioni che utilizzano l'architettura REST.
  - Utilizza annotazioni Java per semplificare lo sviluppo dei servizi Web RESTful.
- Gli sviluppatori decorano i file di classe del linguaggio di programmazione Java con annotazioni JAX-RS per definire le risorse e le azioni che possono essere eseguite su tali risorse.
- Le annotazioni JAX-RS sono annotazioni di runtime, attraverso meccanismi di reflection verrano generate classi di supporto per la risorsa

Vedi: Chapter 13 Building RESTful Web Services with JAX-RS https://docs.oracle.com/cd/E19798-01/821-1841/6nmq2cp1v/index.html

Jersey

- Jersey è l'implementazione di riferimento della specifica di API JAX-RS per la realizzazione di Web Service RESTful su piattaforma Java.
- Una risorsa JAX-RS è un POJO (plain old java object) annotato che fornisce metodi per gestire le richieste HTTP alle URI a cui la risorsa è associata
- Jersey permette di creare risorse semplicemente utilizzando delle specifiche annotazioni per i metodi e le classi.
- Il framework si occupa di gestire le richieste HTTP e la negoziazione della rappresentazione, mentre il programmatore si può occupare della soluzione del problema.

HelloWorldResource

```
import javax.ws.rs.GET;
import javax.ws.rs.Produces;
import javax.ws.rs.Path;
   // class will be at addressable at the URI "/helloworld"
   @Path("/helloworld")
   public class HelloWorldResource {
   // The java method will process HTTP GET requests
   @GET
   /* The Java method will produce content identified by the
   * MIME Media type "text/plain"
   */
   @Produces("text/plain")
   public String getMessage() {
      return "Hello World";} }
```

JAX-RS Annotations

- `HelloWorldResource` e un semplice esempio di risorsa realizzabile tramite Jersey.
- Si tratta di una semplice classe `JAVA` + un insieme di annotazioni

`@Path` indica la URI a cui la risorsa risulta raggiungibile, in questo caso

`http://www.example.com/helloworld`.

È anche possibile incorporare variabili negli URI (URI Template). Ad esempio, potresti chiedere il nome di un utente e passarlo all'applicazione come variabile nell'URI: `/helloworld/{username}.`

`@GET` evidenzia il metodo HTTP da gestire

`@Produces` specifica uno o più `MIME-Type` con cui la rappresentazione della risorsa può essere trasferito al client che ne fa richiesta.

`@Consumes` specifica uno o più `MIME-Type` che possono essere accettati in ingresso

Jersey ResourceConfig

- Implementazione di una Application in Jersey
- Utile per configurare le risorse usate da una applicazione

```
//packages() method lists the package(s) you want Jersey to scan for @Path
and @Provider classes and register them for you
ResourceConfig config = new
ResourceConfig().packages("it.cnit.rest.server.controller");


//Set up the logging feature to print out requests/responses
// The register() method lets you manually register classes and instances of
resources manually
config.register(new LoggingFeature(Logger.getLogger("Server"), Level.INFO,
null, null));


config.register(new JacksonFeature()); // a way to tell Jersey "please use
the Jackson libraries for JSON parsing and serialization",
```

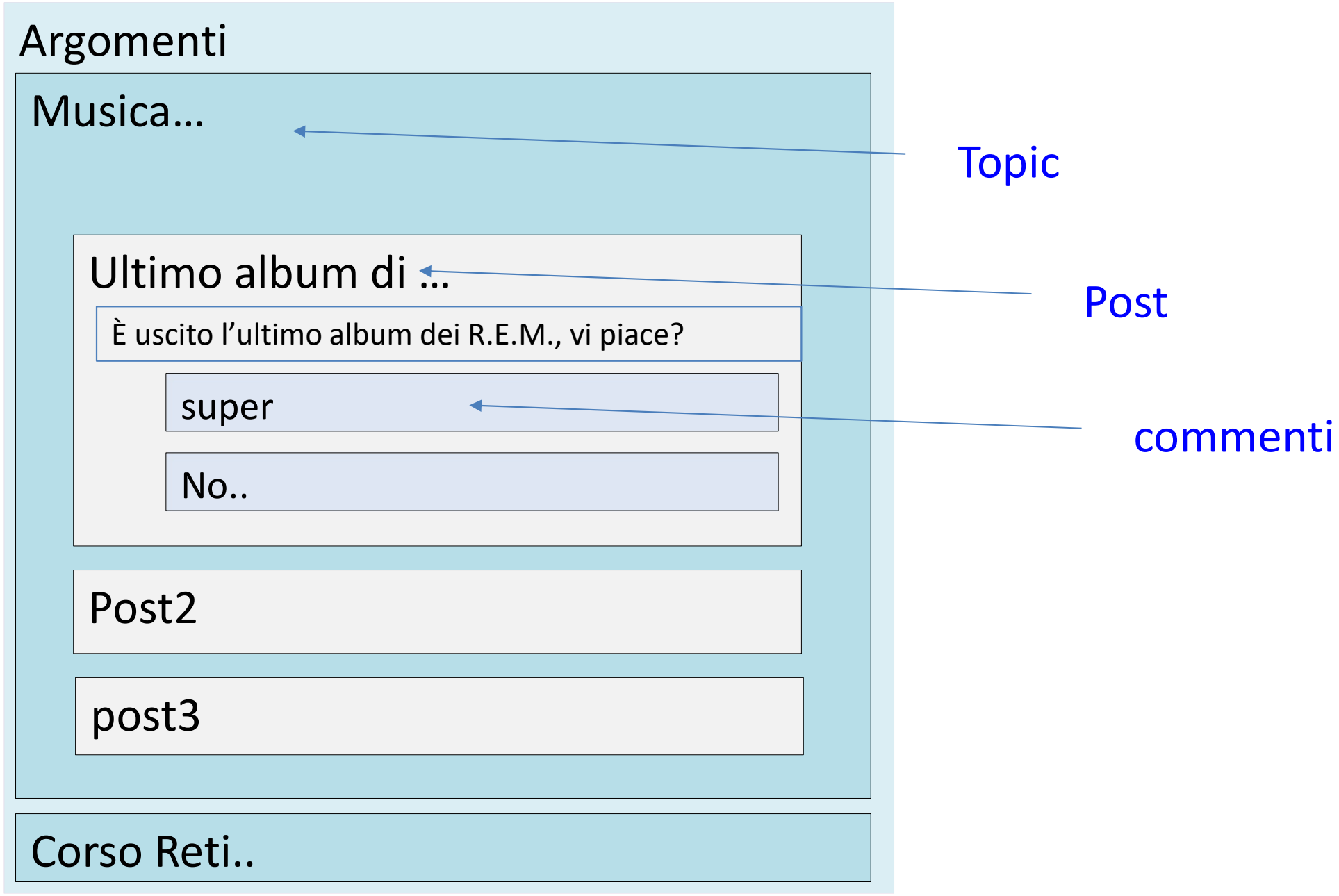
Deployment

- Il framework Jersey comprende anche vari connettori per diverse tipologie di web server e application server.
- E’ possibile esporre i metodi utilizzarlo in un'istanza embedded del server HTTP Grizzly oppure all'interno di una web application servita da un application server (e.g. Tomcat oppure GlassFish)

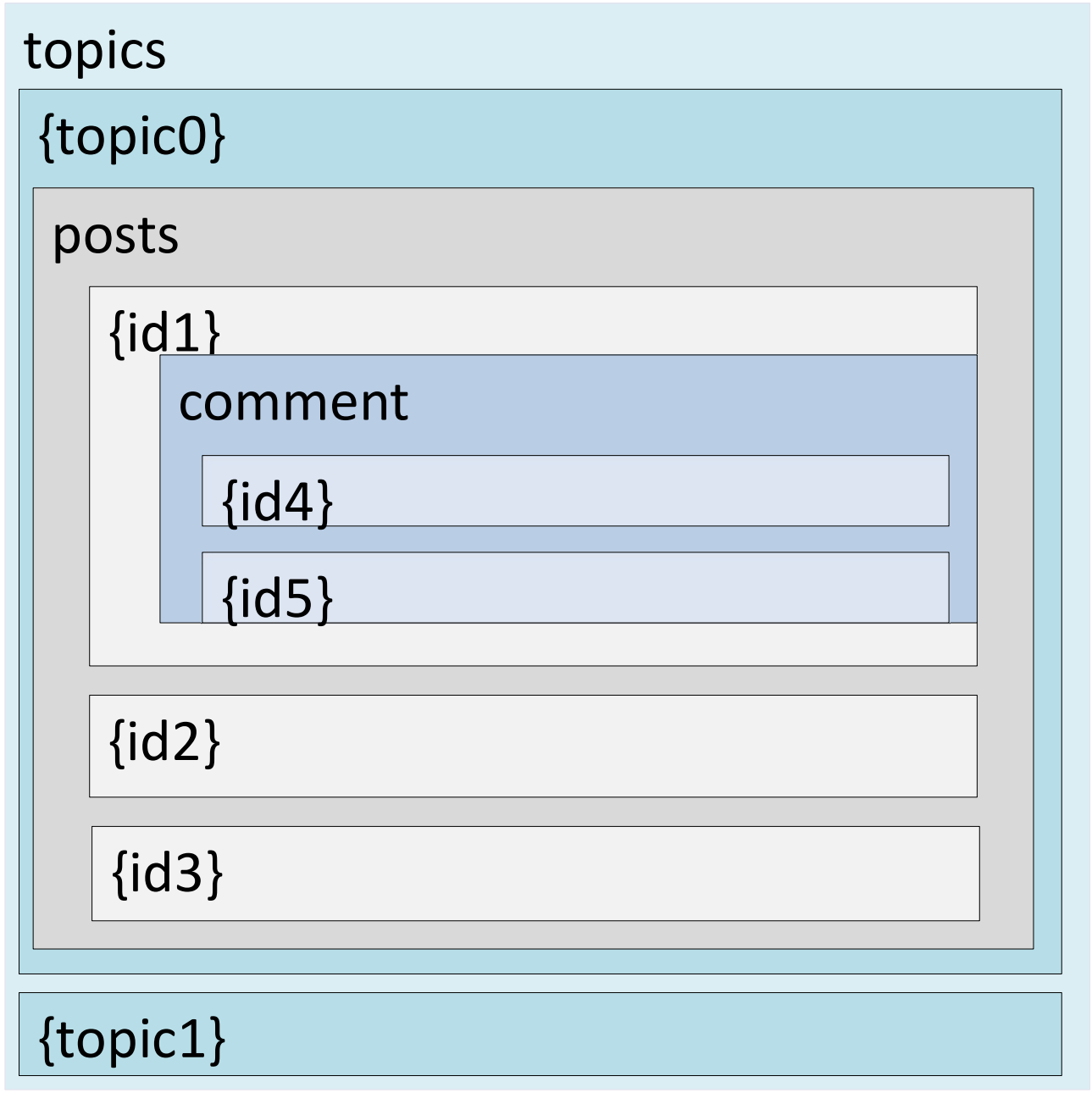
```
HttpServer httpServer =
GrizzlyHttpServerFactory.createHttpServer(URI.create("http://localhost:9999"),
config);
```

```
Create new HttpServer instance.
Parameters:
uri URI on which the Jersey web application will be deployed. Only first path
segment will be used as context path, the rest will be ignored.
Configuration: web application configuration.
Returns:newly created HttpServer.
```

Esempio: Blog/forum



Esempio: Blog/forum



Definiamo le risorse e le relazioni di contenimento

GET /posts
Description: [questo è l'elenco di…
ciao@gmail.com]
Post id1
Post 1d2

Esempio: API per Blog

3. Gli URI contengono gli ID delle risorse figlio
   - /topic
   - /topic/{id}
   - /topic/{id}/post
   - /topic/{id}/post/{id}
   - /topic/{id}/post/{id}/comment
   - /topic/{id}/post/{id}/comment/{id}

Esempio: API per Blog

	GET	PUT	POST	DELETE
/topic	✓		✓	
/topic/{id}	✓	✓		✓
/topic/{id}/post	✓		✓	✓
/topic/{id}/post/{id}	✓	✓		✓
… e così via….				