

Sviluppo di Web Service Java RESTful con JAX-RS e Jersey

Indice generale

PANORAMICA SUI WEB SERVICE RESTFUL.....3

PANORAMICA SU JAX-RS.....4

CREAZIONE DEL PRIMO WEB SERVICE REST.....5

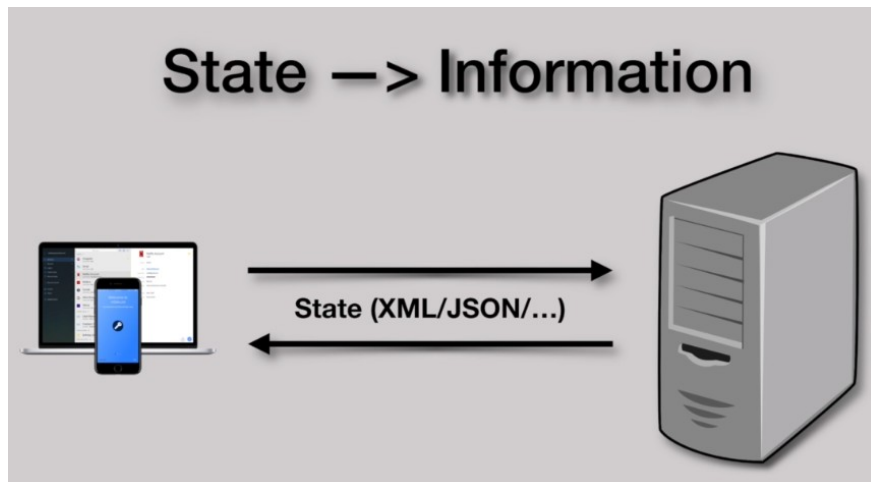
PANORAMICA SUI WEB SERVICE RESTFUL

Per prima cosa, cerchiamo di capire il significato del termine REST.

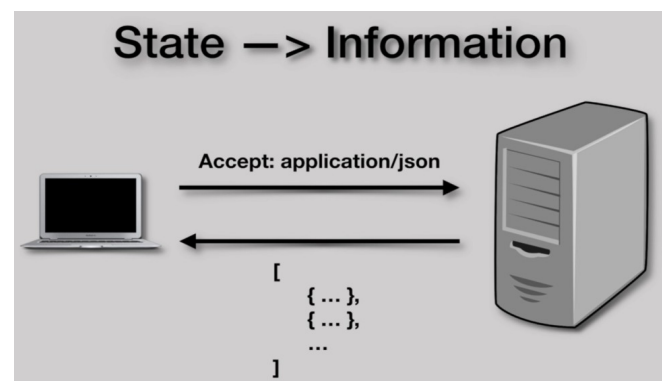
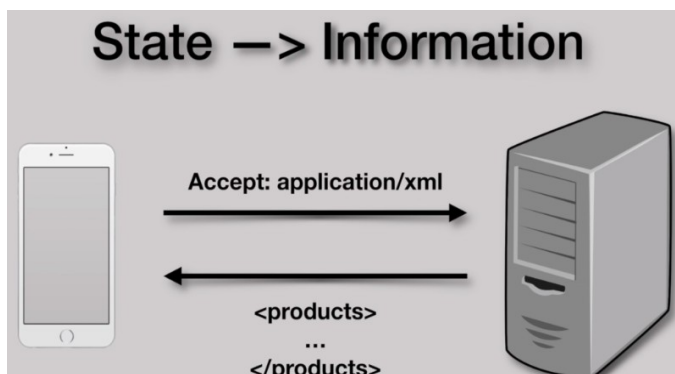
REST è l'acronimo di REpresentational State Transfer, che in italiano si traduce come "trasferimento di stato in rappresentazioni diverse".

Il termine "stato" qui si riferisce allo stato di un insieme di oggetti, denominato risorsa. Per esempio i prodotti, i clienti, gli ordini e gli utenti possono essere considerati tutti come risorse.

In generale ogni risorsa ha alcune informazioni (che definiscono il suo stato) che possono essere scambiate o trasferite in formati diversi, dette rappresentazioni.



Ad esempio, un'applicazione mobile può inviare richieste al server in formato XML, mentre un'altra web application può inviare richieste in formato JSON.



È possibile effettuare una negoziazione sul formato delle informazioni scambiate tra chi ha la risorsa che effettua una richiesta e chi ha la risorsa che eroga il servizio richiesto, in modo tale da avere due rappresentazioni uguali tra loro.

Per i web service REST è necessario:

- utilizzare un URI (Uniform Resource Identifier) per accedervi;
- utilizzare i metodi HTTP (GET, POST, PUT, DELETE...), che rappresentano le azioni da eseguire su una risorsa. Ad esempio, si può effettuare una richiesta GET al server per ottenere informazioni sullo stato di uno specifico ordine, oppure una richiesta POST nel caso in cui si voglia aggiungere un nuovo stato ad una risorsa esistente.

PANORAMICA SU JAX-RS

JAX-RS è una specifica standardizzata da JCP (Java Community Process). JCP è l'istituzione che si occupa di regolare lo sviluppo della tecnologia JAVA.

JAX-RS supporta la creazione di web service REST, che è semplificata grazie all'utilizzo delle annotation presenti all'interno di questa specifica. Una nota importante è che per utilizzare JAX-RS non è richiesta alcuna configurazione.

JAX-RS ha alcune annotation che ci permettono di mappare una classe JAVA come risorsa web. Diamo un'occhiata alle annotation più comunemente utilizzate:

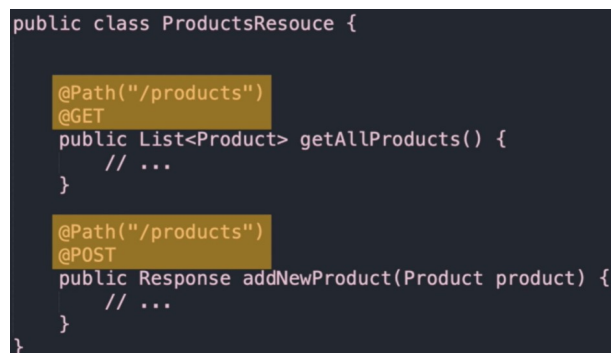
- `@Path`: corrisponde al path a cui risponde un metodo del servizio;



The diagram illustrates the mapping between a URL and a Java class. At the top, the URL `http://example.com/api/products` is shown, with `products` highlighted in yellow. Below it, a Java class `ProductsResource` is shown with a method `getAll()`. The `@Path("/products")` annotation is highlighted in yellow, indicating its role in mapping the URL path to the class method.

```
public class ProductsResource {  
    @Path("/products")  
    public List<Product> getAll() {  
        // ...  
    }  
}
```

- `@GET`, `@POST`, `@PUT`, `@DELETE`: specificano il tipo di richiesta HTTP della risorsa;



The diagram shows a Java class `ProductsResource` with two methods. The first method `getAllProducts()` is annotated with `@Path("/products")` and `@GET`, both highlighted in yellow. The second method `addNewProduct()` is annotated with `@Path("/products")` and `@POST`, also highlighted in yellow, showing how different HTTP methods are mapped to different methods in the same class.

```
public class ProductsResource {  
    @Path("/products")  
    @GET  
    public List<Product> getAllProducts() {  
        // ...  
    }  
    @Path("/products")  
    @POST  
    public Response addNewProduct(Product product) {  
        // ...  
    }  
}
```

- `@Produces`: specifica il tipo di risposta restituita. Il tipo dell'informazione è noto come MIME type.



The diagram shows a Java class `ProductsResource` with a method `getOne()`. The method is annotated with `@Path("/products/{id}")`, `@GET`, and `@Produces({"application/xml", "application/json"})`. The `@Produces` annotation and its value are highlighted in yellow, demonstrating how to specify the MIME type of the response.

```
public class ProductsResource {  
    @Path("/products/{id}")  
    @GET  
    @Produces({"application/xml", "application/json"})  
    public Product getOne(@PathParam("id") int id) {  
        // ...  
    }  
}
```

- `@Consumes`: specifica il tipo di richiesta accettata;

CREAZIONE DEL PRIMO WEB SERVICE REST

Per poter usare le API di JAX-RS, si devono scaricare e inserire all'interno del progetto in cui andremo a sviluppare i servizi REST. Se si sta utilizzando una build automation (ad esempio Maven o Gradle) si devono inserire le dipendenze, altrimenti va fatto tutto a mano.

Nel nostro caso utilizziamo Maven, quindi possiamo importare la seguente dipendenza all'interno del file pom.xml:

```
<dependencies>
  <dependency>
    <groupId>javax.ws.rs</groupId>
    <artifactId>javax.ws.rs-api</artifactId>
    <version>2.0</version>
  </dependency>

  <dependency>
    <groupId>com.sun.jersey</groupId>
    <artifactId>jersey-server</artifactId>
    <version>1.17</version>
  </dependency>

  <dependency>
    <groupId>com.sun.jersey</groupId>
    <artifactId>jersey-servlet</artifactId>
    <version>1.17</version>
  </dependency>

  <dependency>
    <groupId>com.fasterxml.jackson.jaxrs</groupId>
    <artifactId>jackson-jaxrs-json-provider</artifactId>
    <version>2.9.5</version>
  </dependency>
</dependencies>
```

Ora possiamo iniziare a creare il nostro primo servizio REST:

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;

@Path("/hello")
public class RisorsaHello {

    @GET
    public String saluto(){ //http://localhost:8080/Rubrica_REST/hello
        return "Ciao, piacere di conoscerti!";
    }
}
```