



Spring Boot JPA

Alessandro Midolo, PhD Dipartimento di Matematica e Informatica Università di Catania alessandro.midolo@unict.it <u>https://www.dmi.unict.it/amidolo/</u>

A.A. 2024/2025

Spring Data JPA

- Creare un'applicazione che utilizza
 Spring Data JPA per archiviare e recuperare dati in un database relazionale
- Creiamo un'applicazione che memorizza i POJO (Plain Old Java Objects) dei clienti in un database
- Il codice completo del progetto lo trovate già pronto sul repository github



Inizializziamo il Progetto

- Navigate all'url <u>https://start.spring.io/</u> per inizializzare il progetto
- Scegliere Maven e java come linguaggio
- Selezionare le dipendenze Spring Data JPA, H2 Database e Spring Web
- Generare il progetto
- Scaricare il file zip ed estrarlo all'interno del proprio IDE

\	File Edit Selection View Go Run	$\cdots \qquad \leftarrow \rightarrow$,		8 ~		- 0
رل ا	EXPLORER ····	J Customer.java	•			[>~ 🖽
	 ✓ JPADEMO > .mvn > src ✓ main ✓ java\com\labisd\JPADemo ✓ java\com\labisd\JPADemo ✓ java\com\labisd\JPADemo ✓ java\com\labisd\JPADemo ✓ java\com\labisd\JPADemo ✓ src ✓ java\com\labisd\JPADemo ✓ java\com\labisd\JPADemo	src > main > java > 1 package 2 3 import j 4 import j 5 import j 6 import j 7 8 @Entity 9 public c 10 11 @Id 12 @Gener 13 privat 14 privat	<pre>com > labisd > JPADemo > J Customer.java > com.labisd.JPADemo; jakarta.persistence.Entity; jakarta.persistence.GeneratedValue; jakarta.persistence.GenerationType; jakarta.persistence.Id; class Customer { class Customer { ratedValue(strategy=GenerationType.AUTO) te Long id; te String firstName;</pre>	 La c ind nor pre tab L'an con 	classe Customer è annotat icare che si tratta di un'enti esiste alcuna annotazione sume che questa entità sia ella denominata Customer nnotazione @Id fa in modo ne ID dell'oggetto	ta con @Entity, a ità JPA. (Poiché e @Table, si nappata a una) che JPA la ricor	
() ()	 mvnw mvnw.cmd pom.xml 	15 privat 16 17 17 protect 18 19 19 public 20 this 21 this 22 } 23 23	<pre>te String lastName; cted Customer() {} c Customer(String firstName, String lastName) { s.firstName = firstName; s.lastName = lastName;</pre>	 Lan dev Le a son siar stes 	ve essere generato automa altre due proprietà, firstNa no lasciate senza annotazio no mappate a colonne che ssi nomi delle proprietà ste	inde Indica che l nticamente i me e lastName ne. Si presume condividono gl esse	a, che i
83 83 83 84 84 84 84 84 84 84 84 84 84 84 84 84	 > OUTLINE > TIMELINE > JAVA PROJECTS > MAVEN 	24 @Overn 25 public 26 retu 27	<pre>ride c String toString() { urn String.format(format:"Customer[id=%d, firstName='%s', lastName='%s']", id, firstName, lastName); c Long getId() { urn id;</pre>	•	Il costruttore predefinit e Non lo usiamo direttame designato come protetto L'altro costruttore è que creare istanze di Custome database	o esiste solo pe nte, quindi è llo che useremo er da salvare ne	r JPA. o per

× ⊗ 0 ∆ 0 🖗 0 🕁 🗁 Java: Ready

🔍 Ln 2, Col 1 Spaces: 4 UTF-8 CRLF {} Java 🤀 🗘

∢	File Edit Selection View Go Run	… ←	- > [) 8 ~	
Ֆ	EXPLORER ····	J Custon	ner.java 🗨	J CustomerRepository.java X		$ ho$ ~ \square ··
	 ✓ JPADEMO > JPADEMO > .mvn ✓ src ✓ main ✓ java \ com \ labisd \ JPADemo J Customer.java J CustomerRepository.java J JpaDemoApplication.java > resources > test > target ▲ oitattributes 	src > main 1 r 2 4 5 2 6 7 r 8 9 10 11 12	n > java > package import j import o public i List <c Custom }</c 	<pre>com > labisd > JPADemo > J CustomerRepository.java > Language Support for Jav com.labisd.JPADemo; ava.util.List; rg.springframework.data.repository.CrudRepository; nterface CustomerRepository extends CrudRepository<customer, lo<br="">ustomer> findByLastName(String lastName); er findById(long id);</customer,></pre>	ra(TM) by Red Hat > ⊷ CustomerRe	pository
Д	.gitignore					
() ()	 HELP.md mvnw mvnw.cmd pom.xml 	• (• • (• (Custom I tipo di specifica Custome classe C entità Cu	erRepository estende l'interfaccia CrudRepository entità e ID con cui funziona, Customer e Long, sono ati nei parametri generici su CrudRepository erRepository eredita diversi metodi per lavorare con la ustomer, inclusi metodi per salvare, eliminare e trovare ustomer		
553 (1) (1) (1) (1) (1) (1) (1) (1) (1) (1)	> OUTLINE > TIMELINE > JAVA PROJECTS > MAVEN			 Spring Data JPA dichiarandone la Non è necessari del repository. S quando si esegu 	consente di definire altri r a firma del metodo o scrivere un'implementazio pring Data JPA crea un'imp le l'applicazione.	netodi di query one dell'interfaccia >lementazione
×	⊗ 0 ▲ 0 🖗 0 🎝 🗁 Java: Ready				🔍 Ln 12, Col 2 Spaces: 4	UTF-8 CRLF {} Java 🔠 🕻

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

SEARCH: Q Search

https://docs.spring.io/spring-data/commons/docs/current/api/org/springframework/data/repository/CrudRepository.html

×

Spring Data Core

Interface CrudRepository<T,ID>

All Superinterfaces:

Repository<T,ID>

All Known Subinterfaces:

ListCrudRepository<T,ID>

@NoRepositoryBean

public interface CrudRepository<T,ID> extends Repository<T,ID>

Interface for generic CRUD operations on a repository for a specific type.

Author:

Oliver Gierke, Eberhard Wolff, Jens Schauder

Method Summary

All Methods Insta	nce Methods Abstract Methods	
Modifier and Type	Method	Description
long	count()	Returns the number of entities available.
void	<pre>delete(T entity)</pre>	Deletes a given entity.
void	deleteAll()	Deletes all entities managed by the repository.
void	<pre>deleteAll(Iterable <? extends T> entities)</pre>	Deletes the given entities.
void	<pre>deleteAllById(Iterable <? extends ID> ids)</pre>	Deletes all instances of the type T with the given IDs.
void	deleteById(ID id)	Deletes the entity with the given id.
boolean	<pre>existsById(ID id)</pre>	Returns whether an entity with the given id exists.
Iterable <t></t>	findAll()	Returns all instances of the type.
Iterable <t></t>	<pre>findAllById(Iterable <id> ids)</id></pre>	Returns all instances of the type T with the given IDs.
Optional <t></t>	<pre>findById(ID id)</pre>	Retrieves an entity by its id.
<s extends="" t=""> S</s>	<pre>save(S entity)</pre>	Saves a given entity.
<s extends="" t=""> Iterable <s></s></s>	<pre>saveAll(Iterable <s> entities)</s></pre>	Saves all given entities.

••	•	$\leftarrow \rightarrow$		♀ JPADemo	8 ×	
Ch	EXPLORER	J JpaDemoApplication.java	× J Customer.java	J CustomerRepository.java		$ ho$ \checkmark \square \cdots
	 EXPLORER ···· ✓ JJ Explorer (☆ #E) > .mvn > .vscode ✓ src ✓ main ✓ java / com/labisd / JPADemo J Customer.java J CustomerRepository.java J JPAController.java J JPAController.java J JpaDemoApplication.java > resources > test > target .gitignore HELP.md mvnw mvnw mvnw.cmd pom.xml 	J JpaDemoApplication.java src > main > java > com > lak 11 public class JpaDe 18 19 @Bean 20 public Command 21 return (a 22 // sav 23 repos. 24 repos. 25 repos. 26 repos. 27 repos. 28 // fer 30 log.in 31 log.in 32 repos. 33 log.in 34 }); 35 log.in 36 // fer 37 // fer 38 Custon 39 log.in 40 log.in 41 log.in 42 log.in 44 // fer	<pre>X J Customer.java isd > JPADemo > J Jpa moApplication { LineRunner demo(Custor rgs) -> { re a few customers .tory.save(new Custome .tory.save(new Custome .tory.findAll().forEac .tory.findAll().forEac .tor(customer.toString() .to(""); .tor customer found wi .to(""); .tor customer.toString() .to(""); .tor customers by last .tor().tor customer customer .tor().tor customer customer .tor().tor customer.toString() .to(""); .tor customers by last .tor().tor customer .tor().tor customer .tor().tor().tor customer .tor()</pre>	<pre>J CustomerRepository.java aDemoApplication.java > 4 JpaDemoApplication > 0 omerRepository repository) [] er(firstName:"Dino", lastName:"Sauro")); er(firstName:"Rosa", lastName:"Spina")); er(firstName:"Dora", lastName:"Sella")); er(firstName:"Otto", lastName:"Volante")); er(firstName:"Guido", lastName:"Lamoto")); er(firstName:"Guido", lastName:"Lamoto")); with findAll():"); "); ch(customer -> { ()); tomer by ID tory.findById(id:1L); ith findById(1L):"); "); ());</pre>	 La classe JpaDemo/ include un metodo d sottopone Customer alcuni test Recupera Customer contesto dell'applica Quindi salva degli og usando il metodo sat findAll() recupera tu Customer dal databat findByld() recupera Customer tramite il s findByLastName() p clienti il cui cognome Il metodo demo() re CommandLineRunne automaticamente il o l'applicazione viene 	Application demo() che Repository ad Repository ad ggetti Customer ve() utti gli oggetti ase. un singolo suo ID ber trovare tutti i e è "Volante" stituisce un bean er che esegue codice quando avviata.
0	> OUTLINE	46 log.in 47 repos 48 log.in	fo(" tory.findByLastName() fo(bauer.toString());	"); lastName:"Volante").forEach(bauer -> { ;		
\circ	> TIMELINE	50 });	fo("");			
572	> JAVA PROJECTS	51 };				
h 1	> MAVEN	52 🔒				

🛛 🛇 0 🖄 0 🛛 🖗 0 🍰 🗁 Java: Ready

• •	•	$\leftarrow \rightarrow$		8 ~	
വ	EXPLORER	J JpaDemoApplication.java J	J CustomerRepository.java J JPAController.java	×	\triangleright ~ \square …
	\sim JPADEMO	src > main > java > com > labisd > J	JPADemo > J JPAController.java >		
	 > .mvn > .vscode > src > main > java / com / labisd / JPADemo J Customer.java J CustomerRepository.java J JPAController.java J JPAController.java > resources > test > target .gitattributes .gitignore HELP.md mvnw mvnw.cmd pom.xml 	6 import java.util.List; 7 import java.util.ArrayLi 8 9 @RestController 10 > public class JPAControll 11 12 CustomerRepository 13 14 > 14 > JPAController(Custom 15 this.repository 16 } 17 @GetMapping("/addCustomer custom 18 @GetMapping("/addCustomer custom 19 > public Customer custom 20 @RequestPara 21 [22 Customer newCustom 23 repository.save(24 System.out.print 25 return newCustom 26 } 27 @GetMapping("/getCustomer > com 28 @GetMapping("/getCustomer > com 30 List <customer> com 31 repository.findA 32 if(customers.isE 34 return "No 35 } 36 return "Customer </customer>	<pre>dist; iist; ler { repository; merRepository repository){ r = repository; stomer") tomer(@RequestParam(value = "firstName") String tomer(@RequestParam(value = "firstName") String tomer = new Customer(firstName, lastName) { tomer = new Customer(firstName, lastName); (newCustomer); tln("Customer added: " + newCustomer.toString() mer; stomers") stomers") stomers = new ArrayList (); All().forEach(customers::add); Empty()){ registered customers, please add some"; ers found: \n" + customers; </pre>	 firstName, Definiamo un @RestCompoter gestire le chiamate servizio Esponiamo i due @GetN gestire le richieste su "/addCustomer" e "/get 	ntroller per e al nostro Aapping per Customers
8	> TIMELINE	39 }			
ŝ	> JAVA PROJECTS	40			
2 1	> MAVEN				

Interroghiamo il nostro servizio

- Possiamo quindi interrogare il nostro servizio attraverso il nostro browser
 - http://localhost:8080/getCustomers
 - http://localhost:8080/addCustomer?firstName=Ale&lastName=Midolo

REST API per accedere con JPA

REST API per accedere con JPA

- Creiamo un'applicazione Spring che consente di creare e recuperare oggetti **Person** archiviati in un database utilizzando Spring Data REST
- Spring Data REST prende le funzionalità di Spring HATEOAS e Spring Data JPA e le combina automaticamente
 - Con HATEOAS, un client interagisce con un'applicazione di rete i cui server applicativi forniscono informazioni in modo dinamico tramite ipermedia (e.g. URL)

Inizializziamo il Progetto

- Navigate all'url <u>https://start.spring.io/</u> per inizializzare il progetto
- Scegliere Maven e java come linguaggio
- Selezionare le dipendenze Rest repositories, Spring Data JPA e H2 Database
- Generare il progetto
- Scaricare il file zip ed estrarlo all'interno del proprio IDE

	•	\leftarrow \rightarrow	✓ JPARESTDemo	& ~	
	EXPLORER ····	J JparestDemoApplication.java	J Person.java ●		\triangleright ~ \square …
	\vee JPARESTDEMO	src > main > java > com > labisd > JF	PARESTDemo > 🤳 Person.java >		
Q	> .mvn	1 package com.labisd.JPARE	STDemo;		periory and Acids Patterning and a grant perior and any acids and acids and a grant perior and acids and a grant perior and acids acids and acids ac
	∽ src	2			genotic provide galance and genotic provide set of the
) o	\sim main	3 import jakarta.persisten	ce.Entity;		and an interpretation of the second s
5	\sim java/com/labisd/JPARESTDemo	4 import jakarta.persisten	ce.GenerationType:		
\sim	J JparestDemoApplication.java	6 import jakarta.persisten	ce.Id;		
¥>	J Person.java				
_	> resources	8 @Entity			
Ъ.	> test	9 public class Person {			
	> target	10 11 eTd		Definiamo la classe Person ch	ne
Л		12 @GeneratedValue(strate	qv = GenerationType.AUTO)	rappresenta la nostra entità a	l'interno
		13 private long id;		del Database	
		14		L'ID verrà generato automatio	amente
\Box		15 private String firstNa	me;	I campi firstName e lastName	sono
		16 private String LastNam	e;	correlati con i rispettivi getter	e setter
		18 public String getFirst	Name() {		
	Y pom.xm	19 return firstName;			
		20 }			
		21			
		22 public void setFirstNa	me(String firstName) {		
		25 this.firstwalle = fir	s thaile;		
		25			
		26 public String getLastN	ame() {		
		27 return lastName;			
		28 }			
		30 public void setLastNam	e(String lastName) {		
		31 this.lastName = last	Name;		
		32 }			
3		33 }			
5		34			
m	> JAVA PROJECTS				
2 1	> MAVEN				

• •	•	\leftarrow \rightarrow \bigcirc	>> JPARESTDemo	8 ~	
1 J1	EXPLORER	J JparestDemoApplication.java	J Person.java • J PersonRepository.java ×		\triangleright ~ \square …
	 JPARESTDEMO .mvn src main java / com / labisd / JPARESTDemo JparestDemoApplication.java J Person.java J PersonRepository.java resources test target .gitattributes .gitignore HELP.md mvnw mvnw.cmd pom.xml 	<pre>src > main > java > com > labisd > JPA package com.labisd.JPAREST import java.util.List; import org.springframework import org.springframework import org.springframework import org.springframework import org.springframework import org.springframework import org.springframework List<person> findByLastN v List<person> findByLastN v import import org.springframework import org.springframework interval interval interval interval interval interval interval interval interval interval interval i</person></person></pre>	ARESTDemo > J PersonRepository.java > +• PersonRepositor TDemo; (data.repository.PagingAndSortingRepository; (data.repository.Query.Param; (data.rest.core.annotation.RepositoryRestResource; polectionResourceRel = "people", path = "people") possitory extends PagingAndSortingRepository <person, Name(@Param("name") String name); esto repository è un'interfaccia che consente d erazioni che coinvolgono oggetti Person. Ottiel erazioni estendendo l'interfaccia PagingAndSo inita in Spring Data Commons ase di esecuzione, Spring Data REST crea autor implementazione di questa interfaccia e utilizza epositoryRestResource per indirizzare Spring M endpoint RESTful in /people</person, 	The seguire varie a l'annotazione AVC alla creazione	
(Q)					

> TIMELINE

> JAVA PROJECTS > MAVEN

Interroghiamo il servizio

- Invocando solo http://localhost:8080, il servizio mostra ciò che espone
 - C è un link /people che mette a disposizione tre diverse opzioni: ?page, ?size e ?sort
 - Queste opzioni sono ereditate dall'interfaccia PagingAndSortingRepository

```
Alessandros-MacBook-Air:~ midolo$ curl http://localhost:8080
{
    "_links" : {
        "people" : {
            "href" : "http://localhost:8080/people{?page,size,sort*}",
            "templated" : true
        },
        "profile" : {
            "href" : "http://localhost:8080/profile"
        }
    }
Alessandros-MacBook-Air:~ midolo$
```

Attualmente non ci sono "**Person**" all'interno del nostro database, quindi la chiamata a /people restituisce la lista vuota

```
[Alessandros-MacBook-Air:~ midolo$ curl http://localhost:8080/people
  "_embedded" : {
    "people" : [ ]
  },
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/people?page=0&size=20"
    },
    "profile" : {
      "href" : "http://localhost:8080/profile/people"
    },
    "search" : {
      "href" : "http://localhost:8080/people/search"
  },
  "page" : {
    "size" : 20,
    "totalElements" : 0,
    "totalPages" : 0,
    "number" : 0
```

Creiamo un nuovo record di tipo **Person** usando il comando curl (ci permette di effettuare chiamate POST):

curl -i -H "Content-Type:application/json" -d '{"firstName": "Frodo", "lastName": "Baggins"}' http://localhost:8080/people

- -i : Assicura che si possa vedere il messaggio di risposta, incluse le intestazioni. Viene mostrato l'URI della Persona appena creata
- -H "Content-Type:application/json": Imposta il tipo di contenuto in modo che l'applicazione sappia che il payload contiene un oggetto JSON
- -d: indica i dati che si vogliono inviare
- Se usate windows, bisogna usare le doppie virgolette e non le singole
 - -d "{\"firstName\": \"Frodo\", \"lastName\": \"Baggins\"}"

```
Alessandros-MacBook-Air:~ midolo$ curl -i -H "Content-Type:application/json" -d '{"firstName": "Frodo", "lastName": "Baggins"}' http://localhost:8080/people
HTTP/1.1 201
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Location: http://localhost:8080/people/1
Content-Type: application/hal+json
Transfer-Encoding: chunked
Date: Wed, 06 Nov 2024 14:37:51 GMT
  "firstName" : "Frodo",
  "lastName" : "Baggins",
  "_links" <u>: {</u>
    "self" : {
      "href" : "http://localhost:8080/people/1"
    },
    "person" : {
      "href" : "http://localhost:8080/people/1"
```

- L'oggetto people contiene una lista che include la persona Frodo appena creata
- Al suo interno la persona creata ha un campo "self" che indica proprio l'url per accedere direttamente alla risorsa

}Alessandros-MacBook-A {	ir:~ midolo\$	curl http://loo	calhost:8080/peop	le
"firstName" : "Frodo	",			
"lastName" : "Baggir	s",			
"_links" : {				
"href" : "http:/	/localhost:80	80/people/1"		
},				
"person" : {				
"href" : "http:/	/localhost:80	80/people/1"		
}				

Alessandros-MacBook-Air:~ midolo\$ curl http://localhost:8080/people

```
"_embedded" : {
  "people" : [ {
    "firstName" : "Frodo",
    "lastName" : "Baggins",
    " links" : {
      "self" : {
        "href" : "http://localhost:8080/people/1"
      },
      "person" : {
        "href" : "http://localhost:8080/people/1"
},
"_links" : {
  "self" : {
    "href" : "http://localhost:8080/people?page=0&size=20"
  },
  "profile" : {
    "href" : "http://localhost:8080/profile/people"
  },
  "search" : {
    "href" : "http://localhost:8080/people/search"
},
"page" : {
  "size" : 20,
  "totalElements" : 1,
  "totalPages" : 1,
  "number" : 0
```

Con la "search" è possibile farsi restituire tutti gli url per effettuare delle "query", inclusi anche i parrametri HTTP da usare





Si possono usare i comandi REST PUT, PATCH e DELETE per sostituire, aggiornare ed eliminare record esistenti

```
Alessandros-MacBook-Air:~ midolo$ curl -X PUT -H "Content-Type:application/json" -d '{"firstName": "Bilbo", "lastName": "Baggins"}' http://localhost:8080/people/1
{
    "firstName" : "Bilgo",
    "lastName" : "Baggins",
    "_links" : {
        "self" : {
            "href" : "http://localhost:8080/people/1"
        },
        "person" : {
            "href" : "http://localhost:8080/people/1"
        }
    }
}
```

}Alessandros-MacBook-Air:~ midolo\$ curl -X PATCH -H "Content-Type:application/json" -d '{"firstName": "Bilbo Jr."}' http://localhost:8080/people/1

```
"firstName" : "Bilbo Jr.",
"lastName" : "Baggins",
"_links" : {
    "self" : {
        "href" : "http://localhost:8080/people/1"
    },
    "person" : {
        "href" : "http://localhost:8080/people/1"
    }
}
```

}Alessandros-MacBook-Air:~ midolo\$ curl -X DELETE http://localhost:8080/people/1

```
"firstName" : "Bilbo Jr.",
"lastName" : "Baggins",
"_links" : {
    "self" : {
        "href" : "http://localhost:8080/people/1"
    },
    "person" : {
        "href" : "http://localhost:8080/people/1"
    }
```

Consume a RESTful Web Service

Consume a RESTful Web Service

- Creiamo un'applicazione Spring che consente di "consumare" un servizio web RESTful
- L'applicazione userà il RestTemplate di Spring per ottenere una citazione random su SpringBoot, attraverso l'indirizzo <u>http://localhost:8080/api/random</u>

Inizializziamo il Progetto

- Navigate all'url <u>https://start.spring.io/</u> per inizializzare il progetto
- Scegliere Maven e java come linguaggio
- Selezionare la dipendenza Spring Web
- Generare il progetto
- Scaricare il file zip ed estrarlo all'interno del proprio IDE

Spring Boot Quoters

- Abbiamo bisogno di un servizio REST che esponga effettivamente delle risorse
- Usiamo il servizio messo a disposizione da Spring Boot, chiamato quoters (<u>https://github.com/spring-guides/quoters</u>)
- Facciamo quindi il clone del repository e lo eseguiamo nella nostra macchina
- Possiamo accedere ai risultati attraverso l'url <u>http://localhost:8080/api/random</u> (Questo indirizzo recupera casualmente una citazione su Spring Boot e la restituisce come documento JSON)
- A questo punto torniamo all'applicazione definita prima e la modifichiamo tale che possa usufruire in modo automatico e programmato delle risorse messe a disposizione da questo servizio

•		$\leftarrow \ \rightarrow$, [⊘] Con:	sume_Rest_Service		8 ~) []] 08
2	EXPLORER ···	• J Con	sumeRestApplication.java U	J Quote.java U X	J Value.java U		D ~ tì	, []]
J	\sim consume_rest_service	src > r	nain > java > com > labisd > con	sume > consumeRest >	J Quote.java > ኆ	\$ Quote		
) 0	 > .mvn ✓ src ✓ main 	1 2 3	package com.labisd.consume import com.fasterxml.jacks	.consumeRest; on.annotation.JsonIg	noreProperties;			
11	∽ java/com/labisd/consum ●	5	@JsonIgnoreProperties(igno	reUnknown = true)				
	J ConsumeRestApplication	J 6	public record Quote(String	type, Value value)	{ }			
>	J Quote.java I	J						
-	J Value.java l	J						
	> resources		c. Craisma		"			
Ţ	> test > target		Crelamo La classe tipo Strir	e Quote è un record e nga, e value di tipo Val	ha due parametr ha (un tipo che d	i, type di efiniremo		
_	🚸 .gitattributes ເ	J	SUCCESSI	vamente)				
	🚸 .gitignore u	J						
	🕈 mvnw ເ	J	Per asso	ciare direttamente i da	ati ai tini nersonal	izzati		
	📲 mvnw.cmd u	J	bisogna	specificare che il nom	e della variabile s	sia		
	 pom.xml README.md 	J	esattame restituito chiave n usare l'a esatta de • Nel nost	ente lo stesso della chi o dall'API. Nel caso in c el documento JSON n nnotazione @JsonProp el documento JSON ro caso i nomi delle va SON, quindi possione	iave nel documer cui il nome della v non corrispondan perty per specifica ariabili sono ugua	ito JSON ^r ariabile e la o, si può are la chiave li a quelli		
2)	> OUTLINE		@Jsonla	inoreProperties				
y	> TIMELINE		<u> </u>					
3	> JAVA PROJECTS							
1	> MAVEN							
	ి main* 🕂 🛞 0 🛆 0 🕼 0 🗁 Java:	Ready			Ð.	n 6, Col 50 Spaces:	4 UTF-8 LF {} Java	83 (\$

••	•	$\leftarrow \rightarrow$	$ ho$ Consume_Rest_Service	8 ~	
Дı	EXPLORER ····	J Consume	RestApplication.java U J Quote.java U	J Value.java U 🗙	\triangleright ~ \Im \square …
H L L L L L L L L L L L L L L L L L L L	 CONSUME_REST_SERVICE .mvn src main java/com/labisd/consum J ConsumeRestApplication U J Quote.java U Value.java 	<pre>src > main > 1</pre>	<pre>> java > com > labisd > consume > consumeRest > kage com.labisd.consume.consumeRest; ort com.fasterxml.jackson.annotation.JsonIg onIgnoreProperties(ignoreUnknown = true) lic record Value(Long id, String quote) { }</pre>	J Value.java > 😤 Value	
L L L L L L L L L L L L L L L L L L L	 > test > target .gitattributes .gitignore W HELP.md mvnw mvnw.cmd pom.xml README.md 		 La classe Value è un record e h Long, e quote di tipo String Questi sono i campi che troviar restituito dal servizio REST 	na due parametri, id di tipo mo all'interno del file JSON	
8					
52 T	> JAVA PROJECTS > MAVEN				

• •	•	$\leftarrow \rightarrow$		8 ~	
ſ	EXPLORER ····	J ConsumeRestApplication.java U 🔸	J Quote.java U J Value.java U		⊳∨ เว ⊡ …
	✓ CONSUME_REST_SERVICE	src > main > java > com > labisd > cor	sume > consumeRest > J ConsumeRestApplication.ja	ava > 😤 ConsumeRestApplication	
\cap	> .mvn	1 package com labisd consume	e.consumeRest;		
\sim	∽ src ●	2			
٩o	∽ main ●	<pre>3 import org.slf4j.Logger; 4 import org.slf4j.LoggerFac</pre>	rtory.		
ð <u>11</u>	√ java/com/labisd/consum ●	5 import org.springframework	k.boot.CommandLineRunner;		
\sim	J ConsumeRestApplication U	6 import org.springframework	k.boot.SpringApplication;		• Il Logger invia
₽,>	J Quote.java U	7 import org.springframework	<pre>k.boot.autoconfigure.SpringBootApplication;</pre>		l'output al log (nel
	J Value.java U	8 import org.springframework	<pre>k.boot.web.client.RestTemplateBuilder;</pre>		nostro caso la
₽ <u></u>		9 import org.springframewor	k.context.annotation.Bean;		console)
ц,	> test	10 import org.springframework	k.context.annotation.Profile;		
Л	/ lest	12 1mport org.springframework	k.web.client.Restremplate;	l.	
A	> largel	13 @SpringBootApplication			
_	• .gitattributes U	14 public class ConsumeRestA	oplication ┨		Il RestTemplate usa
ᄂ	 ♦ .gitignore U 	15	• _		la libreria Jackson
	➡ HELP.md	16 private static final I	Logger log = LoggerFactory.getLogger(clazz:Cons	<pre>sumeRestApplication.class);</pre>	per gestire i file
	🕈 mvnw 🛛 U	17			JSON per
	📲 mvnw.cmd 🛛 U	Run Debug	in (Cturing [] and a)		processare i dati
	🕴 pom.xml 🛛 U	18 public static void ma	<pre>in(String[] args) { run(ConsumePostApplication class args); </pre>		ricevuti
	 README.md 	20 }	indif(consumeries (Apprication Class, args),		
		21			
		22 @Bean			
		23 public RestTemplate re	<pre>estTemplate(RestTemplateBuilder builder) {</pre>		
		24 return builder.bu	ild();	•	
		25 }			eseque il RestTemplate e
		26			di conseguenza recupera
		27 @Bean			le quotes richieste
		28 (Profile("itest")	per run(PectTemplate rectTemplate) throws Even	ation J	
		30 return args -> {			
		31 Ouote guote =	restTemplate.getForObject(
		32 "http	://localhost:8080/api/random", Quote.class);		
(8)		33 log.info(quote	e.toString());		
		34 };			
502	> JAVA PROJECTS	35 😯 }			
	> MAVEN	36			
× ß	° main* 😳 🏵 0 🛆 0 🖾 0 🗁 Java: Re	ady		🔍 🛛 Ln 36, Col 2	Tab Size: 4 UTF-8 LF {} Java 🤀 🗘

	•	(application.properties		
ſ,	EXPLORER	··· J Consume	R ≣ application.properties src/main/resources	file results	⊳∨ เъ ш …
	 CONSUME_REST_SERVICE .mvn src main java/com/labisd/consum ConsumeRestApplication Quote.java Value.java resources test target .gitignore HELP.md mvnw mvnw.cmd pom.xml README.md 	 src > main > 1 pack 2 3 impo 4 impo 5 impo 0 6 impo 0 7 impo 0 9 impo 10 impo 11 impo 12 0 13 @Spi 14 pub¹ 15 16 2 16 2 17 U 18 1 19 20 21 22 23 24 25 26 27 28 29 30 31 	<pre>java_com > labisd > consume > consumeRest > _J = ConsumeRestApplica cage com.labisd.consume.consumeRest; prt org.slf4j.Logger; prt org.springframework.boot.CommandLineRunner; prt org.springframework.boot.SpringApplication; prt org.springframework.boot.autoconfigure.SpringBootApplication prt org.springframework.boot.web.client.RestTemplateBuilder; prt org.springframework.context.annotation.Bean; prt org.springframework.context.annotation.Profile; prt org.springframework.web.client.RestTemplate; ringBootApplication tic class ConsumeRestApplication { private static final Logger log = LoggerFactory.getLogger(clazz Run Debug public static void main(String[] args) { SpringApplication.run(ConsumeRestApplication.class, args); } @Bean public RestTemplate restTemplate(RestTemplateBuilder builder) { return builder.build(); } @Bean @Profile("!test") public CommandLineRunner run(RestTemplate restTemplate) throws return args -> { Quote quote = restTemplate.getForObject(</pre>	<pre>ation.java > {; ConsumeRestApplication ></pre>	<text></text>
8		32 33	<pre>"http://localhost:8080/api/random", Quote.class log.info(quote.toString());</pre>	s);	
		34	};		
502	> JAVA PROJECTS	35	}		
11	> MAVEN	36 }			

🎽 🎖 main* 😌 🛞 0 🛆 0 🖉 0 🗁 Java: Ready

	•	$\leftarrow \rightarrow \qquad $	
	EXPLORER	· J ConsumeRestApplication.java U ● Ξ application.properties U × J Quote.java U J Value.java U	ţ, 🛙 …
	 EXPLORER CONSUME_REST_SERVICE .mvn src main java/com/labisd/consum J ConsumeRestApplication J Quote.java J Value.java J Value.java Value.java Static static static implates application.properties test target .gitattributes .gitignore HELP.md mvnw.cmd mvnw.cmd README.md 	 ConsumeRestApplication.java U • E application.properties U × J Quote java U J Value java U src > main > resources > E application.properties spring.application.name=consumeRest server.port=8081 • All'interno del file application.properties andiamo ad inserire l'opzione "serve.port=8081" per modificare la porta su cui il servizio starà in ascolto 	
ହ	> OUTLINE		

> TIMELINE

 > JAVA PROJECTS

 > MAVEN

> MAVEN







Alessandro Midolo, PhD

Dipartimento di Matematica e Informatica

Università di Catania

alessandro.midolo@unict.it

https://www.dmi.unict.it/amidolo/

A.A. 2024/2025