* Spring uses the [Jackson JSON](https://github.com/FasterXML/jackson) library to automatically marshal instances of type Any JaVa object into JSON.
* A key difference between a traditional MVC controller and the RESTful web service controller is the way that the HTTP response body is created. Rather than relying on a [view technology](https://spring.io/understanding/view-templates) to perform server-side rendering of the greeting data to HTML, this RESTful web service controller simply populates and returns a Java class object. The object data will be written directly to the HTTP response as JSON.
* Java class object must be converted to JSON using Jackson

**Consuming a RESTful Web Service:**

A more useful way to consume a REST web service programmatically . For this purpose Spring provides a convenient template class called [RestTemplate](https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/client/RestTemplate.html)

* @**ResponseBody** means the returned String is the response, not a view name
* @**RequestParam** means it is a parameter from the GET or POST request
* @GetMapping is a shortcut for @RequestMapping(method=GET). @RequestMapping maps

# Accessing data with MySQL:

# Step 1: Create a new database

mysql> create database db\_example; -- Create the new database

mysql> create user 'springuser'@'%' identified by 'ThePassword'; -- Creates the user

mysql> grant all on db\_example.\* to 'springuser'@'%'; -- Gives all the privileges to the new user on the newly created database

## Step 2: Create the application.properties file

The default in database is H2, so when you want to change this and use any other database you must define the connection attributes in the application.properties file.

spring.jpa.hibernate.ddl-auto=create

spring.datasource.url=jdbc:mysql://localhost:3306/db\_example

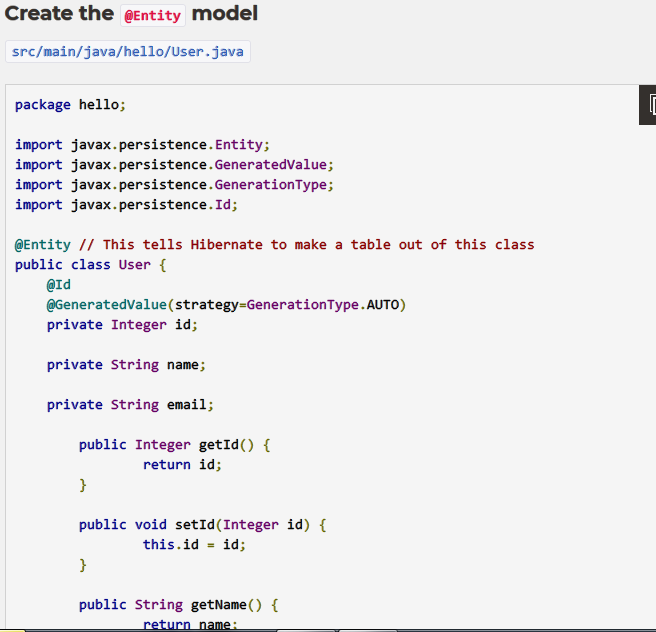
spring.datasource.username=springuser

spring.datasource.password=ThePassword

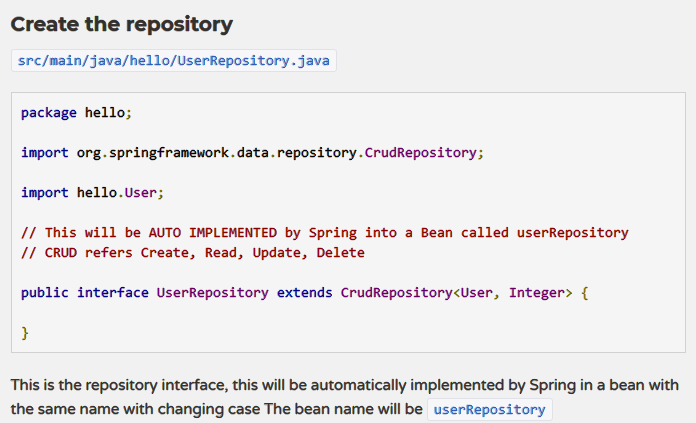
Here, spring.jpa.hibernate.ddl-auto can be none, update, create, create-drop.

* none This is the default for MySQL, no change to the database structure.
* update Hibernate changes the database according to the given Entity structures.
* create Creates the database every time, but don’t drop it when close.
* create-drop Creates the database then drops it when the SessionFactory closes.
* We here begin with create because we don’t have the database structure yet. After the first run, we could switch it to update or none according to program requirements. Use update when you want to make some change to the database structure.
* The default for H2 and other embedded databases is create-drop, but for others like MySQL is none

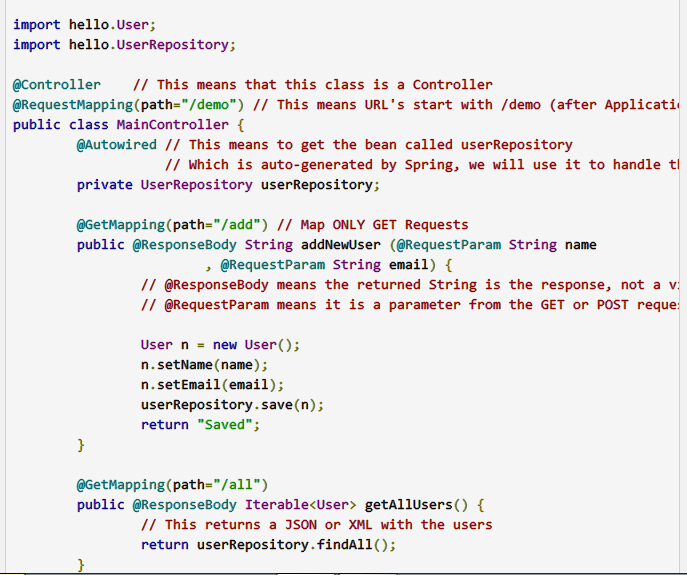
## Step 3: Create the @Entity model



This is the entity class which Hibernate will automatically translate into a table.

Step 4: 

## Step 5 : Create a new controller for your Spring application



## Step 6 : Make the application executable



* @ResponseBody annotation is used to map the response object in the response body. Once the response object is returned by the handler method, MappingJackson2HttpMessageConverter kicks in and convert it to JSON response.

### Compare SOAP and REST web services?

|  |  |
| --- | --- |
| **SOAP** | **REST** |
| SOAP is a standard protocol for creating web services. | REST is an architectural style to create web services. |
| SOAP is acronym for Simple Object Access Protocol. | REST is acronym for REpresentational State Transfer. |
| SOAP uses WSDL to expose supported methods and technical details. | REST exposes methods through URIs, there are no technical details. |
| SOAP web services and client programs are bind with WSDL contract | REST doesn’t have any contract defined between server and client |
| SOAP web services and client are tightly coupled with contract. | REST web services are loosely coupled. |
| SOAP learning curve is hard, requires us to learn about WSDL generation, client stubs creation etc. | REST learning curve is simple, POJO classes can be generated easily and works on simple HTTP methods. |
| SOAP supports XML data format only | REST supports any data type such as XML, JSON, image etc. |
| SOAP web services are hard to maintain, any change in WSDL contract requires us to create client stubs again and then make changes to client code. | REST web services are easy to maintain when compared to SOAP, a new method can be added without any change at client side for existing resources. |
| SOAP web services can be tested through programs or software such as Soap UI. | REST can be easily tested through CURL command, Browsers and extensions such as Chrome Postman. |

**1) Explain what is REST and RESTFUL?**

REST represents REpresentational State Transfer; it is a relatively new aspect of writing web API.

RESTFUL is referred for web services written by applying REST architectural concept are called RESTful services, it focuses on system resources and how state of resource should be transported over HTTP protocol to different clients written in different language. In RESTFUL web service HTTP methods like GET, POST, PUT and DELETE can be used to perform CRUD operations.

**2) Mention what are the HTTP methods supported by REST?**

HTTP methods supported by REST are:

* **GET:**It requests a resource at the request URL. It should not contain a request body as it will be discarded. Maybe it can be cached locally or on the server.
* **POST:**It submits information to the service for processing; it should typically return the modified or new resource
* **PUT:**At the request URL it update the resource
* **DELETE:**At the request URL it removes the resource
* **OPTIONS:**It indicates which techniques are supported
* **HEAD:**About the request URL it returns meta information

**3) POST vs PUT**

|  |  |
| --- | --- |
| POST | PUT |
| POST sends data to a particular URI and expects the resource at that URI to deal with the request. The web server at this point can decide what to do with the data in the context of specified resource | "PUT" puts a file or resource at a particular URI and exactly at that URI. If there is already a file or resource at that URI, PUT changes that file or resource. If there is no resource or file there, PUT makes one |
| POST is not idempotent, meaning if you invoke POST multiple times it keeps creating more resources | PUT is idempotent meaning, invoking it any number of times will not have an impact on resources. |

**4)  what is the difference between SOAP and REST?**

|  |  |
| --- | --- |
| **SOAP** | **REST** |
| SOAP is a protocol through which two computer communicates by sharing XML document | Rest is a service architecture and design for network-based software architectures |
| SOAP permits only XML | REST supports many different data formats |
| SOAP based reads cannot be cached | REST reads can be cached |
| SOAP is like custom desktop application, closely connected to the server | A REST client is more like a browser; it knows how to standardized methods and an application has to fit inside it |
| SOAP is slower than REST | REST is faster than SOAP |
| It runs on HTTP but envelopes the message | It uses the HTTP headers to hold meta information |

**5) List out the tools or API for developing or testing web api?**

Testing tools for web services for REST APIs includes

Spring REST web service using MVC

Jersey API

CFX

Axis

Restlet,

**6) REST HTTP Response code:**

HTTP defines forty standard status codes that can be used to convey the results of a client’s request. The status codes are divided into the five categories presented below.

|  |  |
| --- | --- |
| **CATEGORY** | **DESCRIPTION** |
| 1xx: Informational | Communicates transfer protocol-level information. |
| 2xx: Success | Indicates that the client’s request was accepted successfully. |
| 3xx: Redirection | Indicates that the client must take some additional action in order to complete their request. |
| 4xx: Client Error | This category of error status codes points the finger at clients. |
| 5xx: Server Error | The server takes responsibility for these error status codes. |

|  |  |  |
| --- | --- | --- |
| **Status Code** | **Description** | |
| 1xx: Informational | | |
| 100 | Continue | |
| 101 | Switching Protocols | |
| 102 | Processing (WebDAV) | |
| 2xx: Success | | |
| 200 OK | Response to a successful REST API action. The HTTP method can be GET, POST, PUT, PATCH or DELETE. | |
| 201 Created |  | |
| 204 No Content |  | |
| 3xx: Redirection | | |
| 304 | Not Modified | |
| 4xx: Client Error | | |
| 400 Bad Request | | The request is malformed, such as message body format error. |
| 401 Unauthorized | | Wrong or no authentication ID/password provided. |
| 403 Forbidden | | It's used when the authentication succeeded but authenticated user doesn't have permission to the request resource. |
| 404 Not Found | | When a non-existent resource is requested. |
| 405 Method Not Allowed | | The error checking for unexpected HTTP method. For example, the RestAPI is expecting HTTP GET, but HTTP PUT is used. |
| 429 Too Many Requests | | The error is used when there may be DOS attack detected or the request is rejected due to rate limiting |
| 5xx: Server Error | | |
| 500 | Internal Server Error | |

**7) How to secure your REST API?**

* Keep it Simple. Secure an API/System – just how secure it needs to be. ...
* Always Use HTTPS. ...
* Use Password Hash. ...
* Never expose information on URLs. ...
* Consider OAuth. ...
* Consider Adding Timestamp in Request. ...
* Input Parameter Validation.

Spring boot micro services dynamic scaling:

https://dzone.com/articles/spring-boot-autoscaler

https://dzone.com/articles/dynamic-configuration-management-in-microservice-a