SPRINGBOOT APPLICATION ON DOCKER

- Younggyu Kim (younggyu.kim@oracle.com)
- Oracle Cloud Adoption Platform
- Principal Sales Consultant

SPRING BOOT APPLICATION ON DOCKER

- PT: https://gitpitch.com/credemol/docker-spring-tutorial?p=presentation#/
- Slack: https://cloudnativeapp.slack.com

PRE REQUISITES

- JDK 1.8: http://www.oracle.com/technetwork/java/javase/download
downloads-2133151.html
- Maven 3.5: https://maven.apache.org/download.cgi
- Spring Tool Suite: https://spring.io/tools/sts
- Postman: https://www.getpostman.com/apps

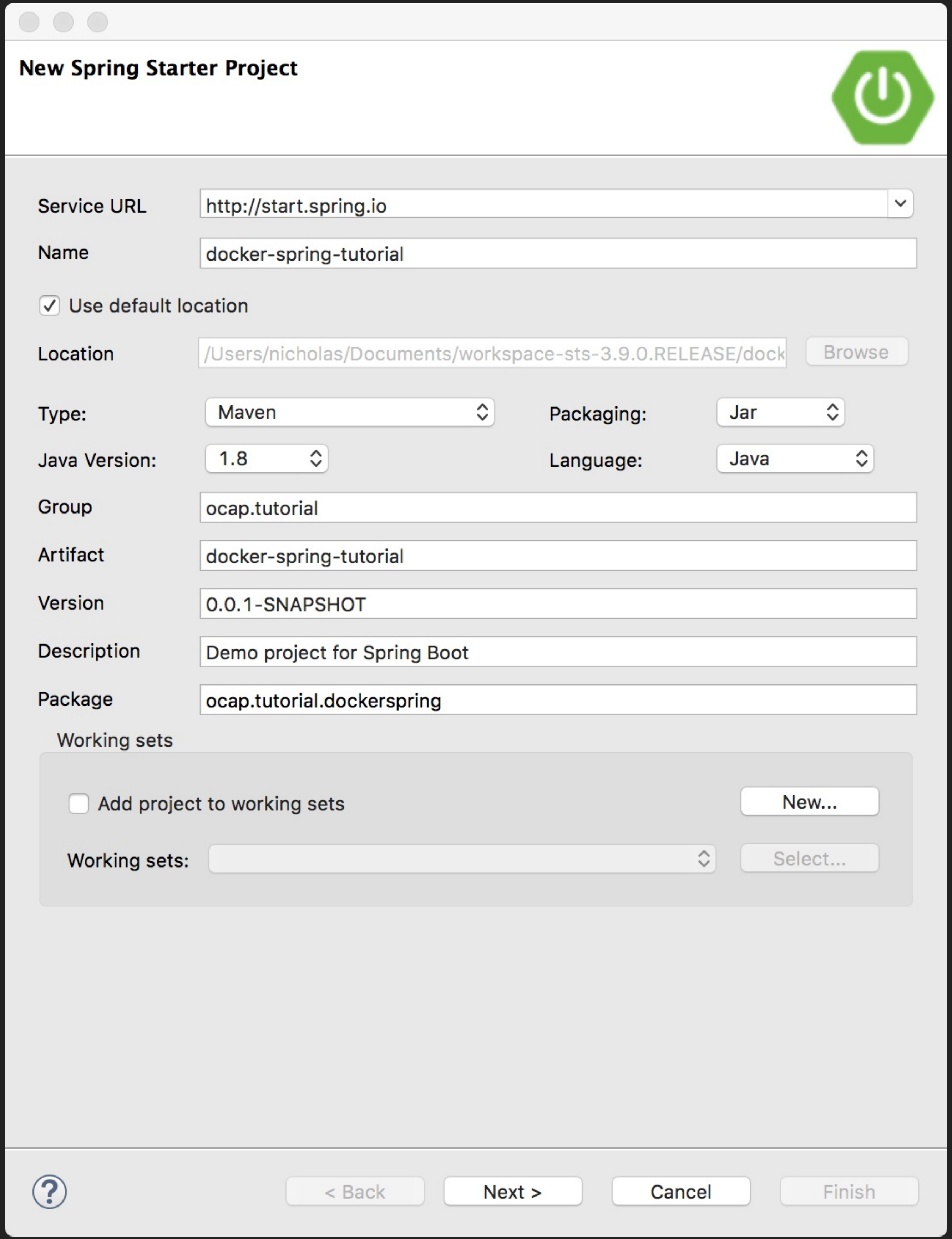
SET ENVIRONMENT VARIABLES:

- JAVA_HOME: specify the location where java has been installed
- MAVEN_HOME: specify the location where you unzipped the maven.zip file
- PATH: $JAVA_HOME/bin:$M2_HOME/bin:

```
$ java -version
$ mvn -v
$ echo $PATH
```

CREATE A SPRING BOOT PROJECT

1. Run Spring Tool Suite and see where the workspace is
2. Run: File > New > Spring Starter Project

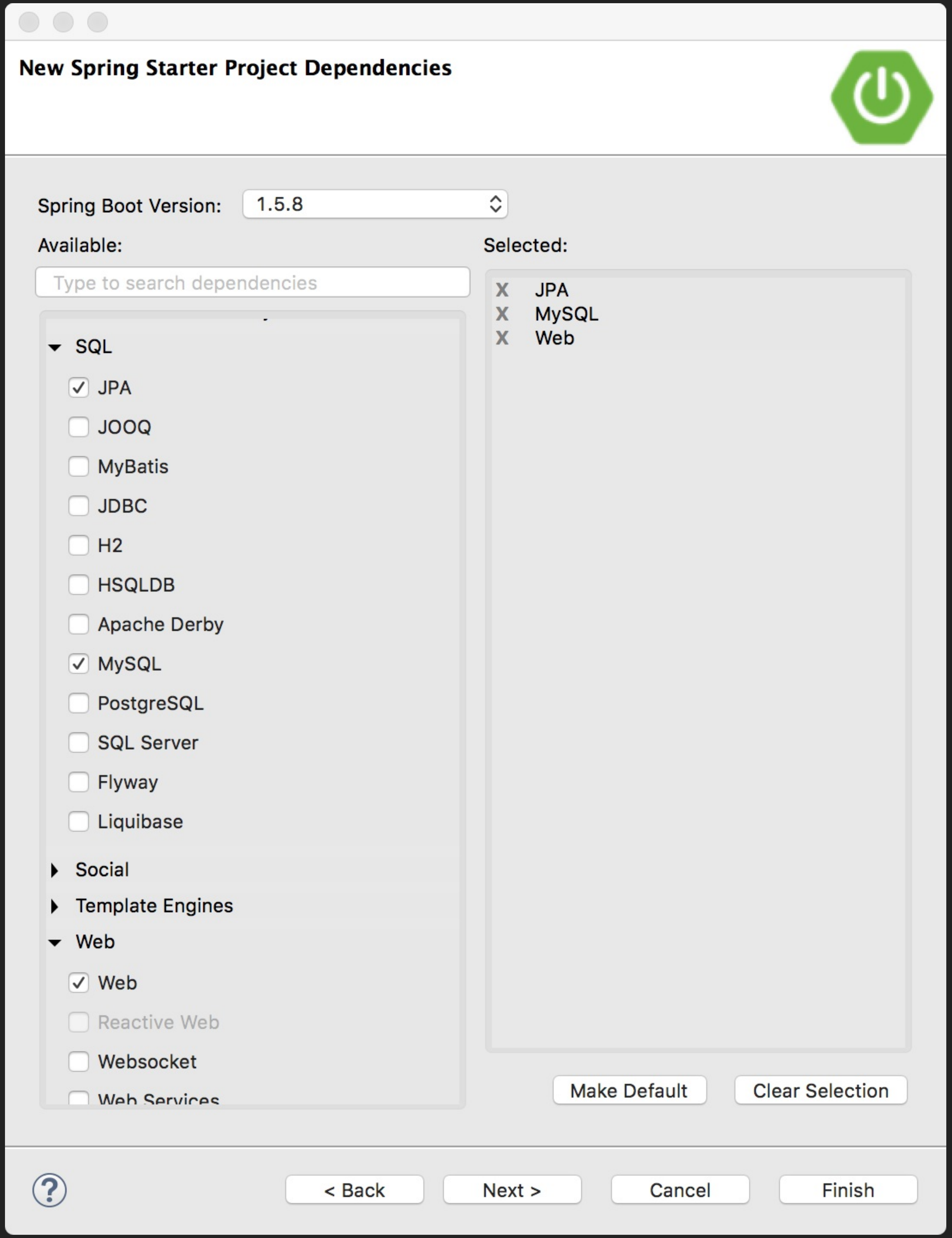

New Spring Starter Project

Service URL	http://start.spring.io
Name	docker-spring-tutorial
✓ Use default location	
Location	/Users/nicholas/Documents/workspace-sts-3.9.0.RELEASE/dock
Type:	Maven
Packaging:	Jar
Java Version:	1.8
Language:	Java
Group	ocap.tutorial
Artifact	docker-spring-tutorial
Version	0.0.1-SNAPSHOT
Description	Demo project for Spring Boot
Package	ocap.tutorial.dockerspring

Working sets

☐ Add project to working sets

Working sets:



New Spring Starter Project Dependencies

Spring Boot Version: 1.5.8

Available:

Type to search dependencies

- SQL
  - [✓] JPA
  - [ ] JOOQ
  - [ ] MyBatis
  - [ ] JDBC
  - [ ] H2
  - [ ] HSQLDB
  - [ ] Apache Derby
  - [✓] MySQL
  - [ ] PostgreSQL
  - [ ] SQL Server
  - [ ] Flyway
  - [ ] Liquibase
- Social
- Template Engines
- Web
  - [✓] Web
  - [ ] Reactive Web
  - [ ] Websocket
  - [ ] Web Services

Selected:

- X JPA
- X MySQL
- X Web

PROJECT PROPERTIES

Property Name	value
Name	docker-spring-tutorial
Type	Maven
Group	ocap.tutorial
Artifact	docker-spring-tutorial
Package	ocap.tutorial.dockerspring

PROJECT DEPENDENCIES

- SQL > JPA
- SQL > MySQL
- Web > Web

CREATE DOCKERFILE-MYSQL

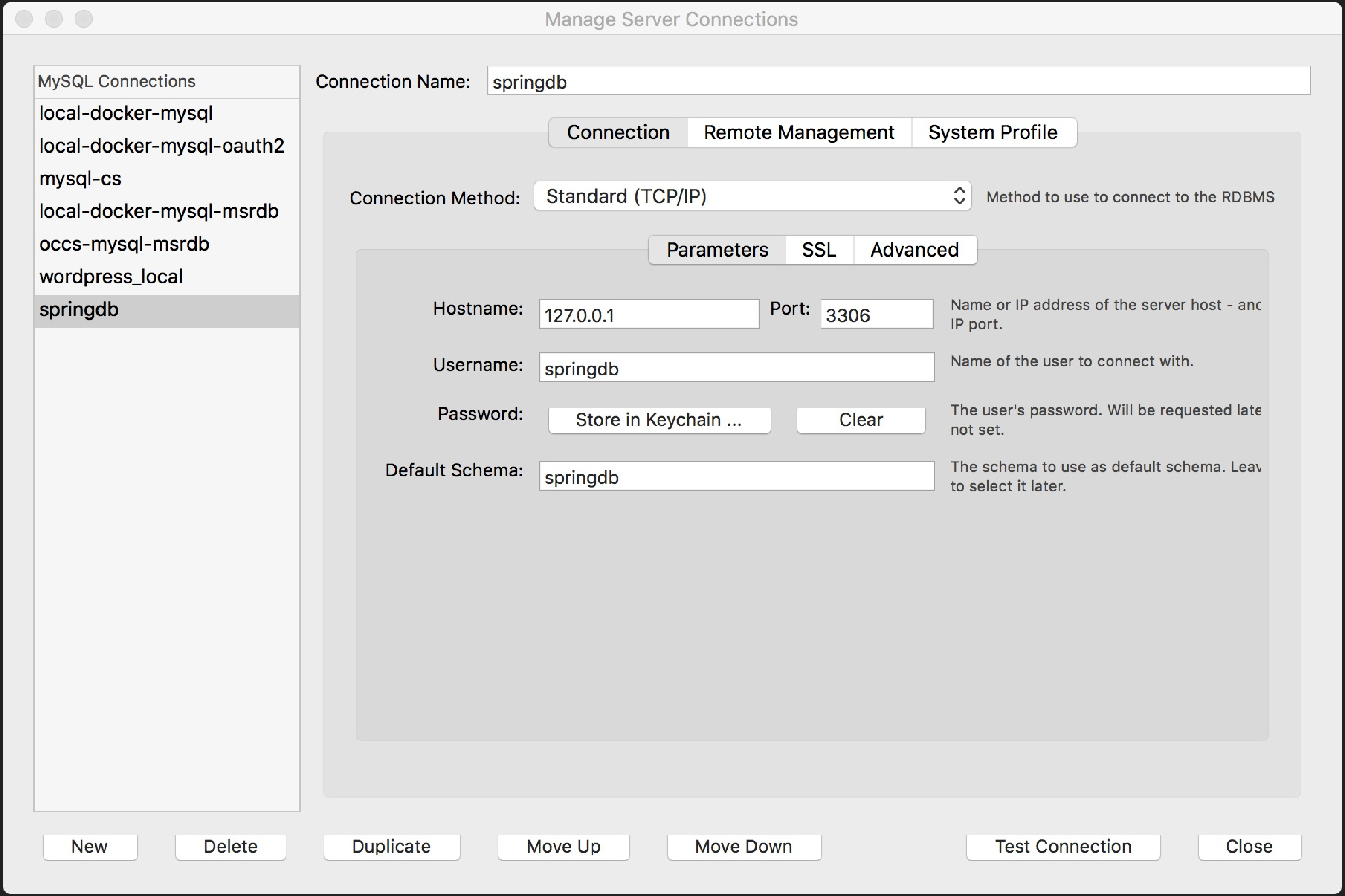
```
$ mkdir docker
$ cd docker
$ vi Dockerfile-mysql
```

DOCKERFILE-MYSQL

```
FROM mysql:5.7
ENV MYSQL_ROOT_PASSWORD="springdb"
ENV MYSQL_DATABASE="springdb"
ENV MYSQL_USER="springdb"
ENV MYSQL_PASSWORD="springdb"
EXPOSE 3306
```

BUILD DOCKER IMAGE AND RUN

```
$ docker build -t spring_db -f Dockerfile-mysql .
$ mkdir -p ~/tmp/spring_data

$ docker run -d --name spring_db -p 3306:3306 \
 -v ~/tmp/spring_data:/var/lib/mysql spring_db
```



MySQL Connections
- local-docker-mysql
- local-docker-mysql-oauth2
- mysql-cs
- local-docker-mysql-msrdb
- occs-mysql-msrdb
- wordpress_local
- springdb

Connection Name: springdb

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Hostname:	127.0.0.1 Port: 3306	Name or IP address of the server host - anc IP port.
Username:	springdb	Name of the user to connect with.
Password:	Store in Keychain ... Clear	The user's password. Will be requested late not set.
Default Schema:	springdb	The schema to use as default schema. Leav to select it later.

SRC/MAIN/RESOURCES/APPLICATION.P ROPERTIES

```
spring.datasource.url = jdbc:mysql://localhost:3306/springdb?useSSL=false

# Username and password
spring.datasource.username = springdb
spring.datasource.password = springdb
```

COMPILE YOUR PROJECT WITH MAVEN

1. Open a terminal window
2. Change directory to docker-spring-tutorial under Workspace Directory
3. run *mvn compile*. It takes a while to download required library from the Internet.

```
$ mvn clean
$ ls -l

$ mvn compile
$ ls -l target

$ mvn clean
$ ls -l
$ mvn package
$ ls -l target
```

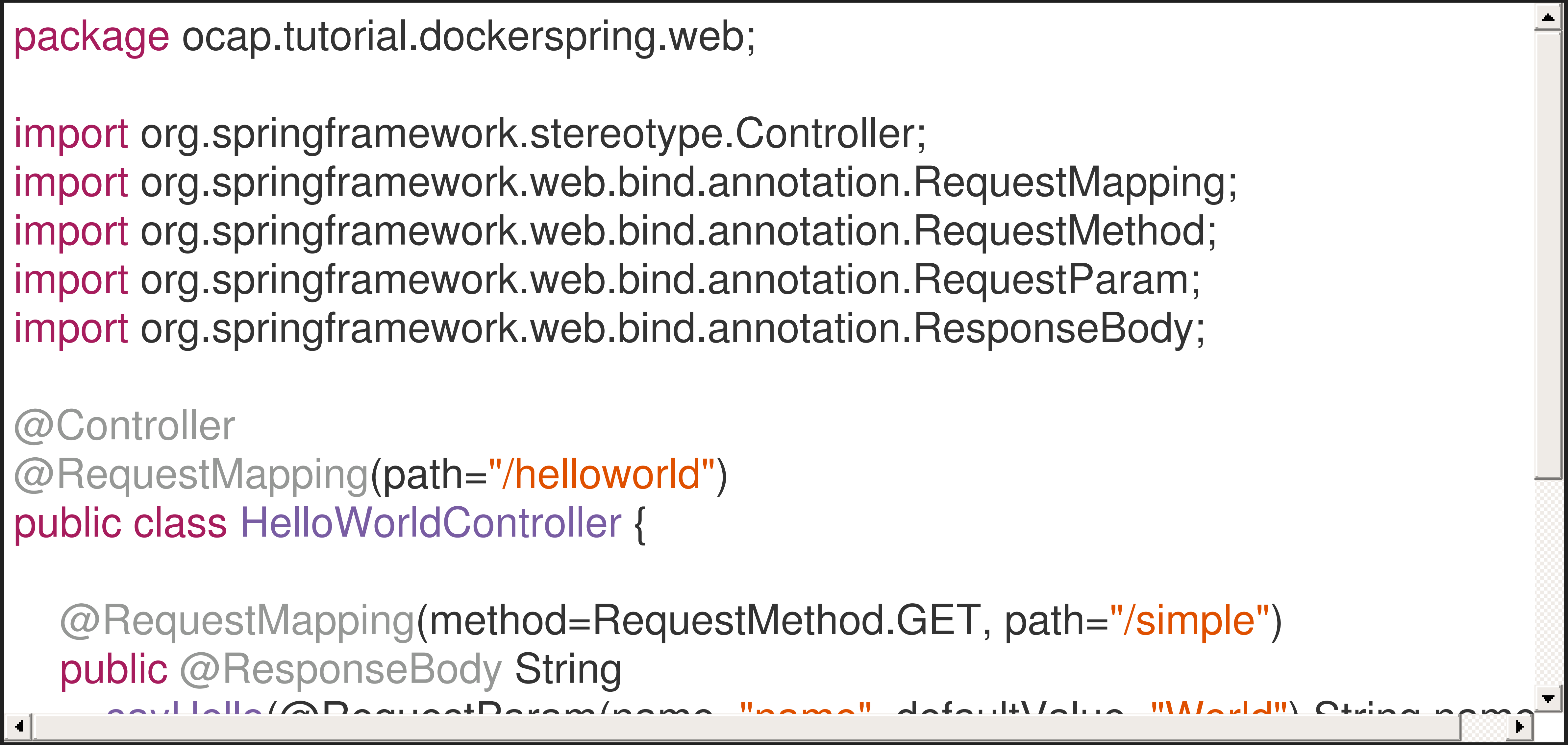
CREATE A CONTROLLER

1. select ocap.tutorial.dockerspring under src/main/java directory
2. Run New > File > Class

Property Name	Property Value
Package	ocap.tutorial.dockerspring.web
Name	HelloWorldController

CONTROLLER

File:
ocap/tutorial/dockerspring/web/HelloWorldController.java



```
package ocap.tutorial.dockerspring.web;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
@RequestMapping(path="/helloworld")
public class HelloWorldController {

    @RequestMapping(method=RequestMethod.GET, path="/simple")
    public @ResponseBody String
        sayHello(@RequestParam(name="name", defaultValue="World") String name
```

TEST HELLOWORLDCONTROLLER

```
$ mvn clean package
$ java -jar target/docker-spring-tutorial-0.0.1-SNAPSHOT.jar
```

- http://localhost:8080/helloworld/simple
- http://localhost:8080/helloworld/simple?name=Kim

ENTITY

JPA: Java Persistence API

1. select ocap.tutorial.dockerspring under src/main/java directory
2. Run New > File > Class

Property Name	Property Value
Package	ocap.tutorial.dockerspring.entity
Name	User

ENTITY

File ocap/tutorial/dockerspring/entity/User.java

```
package ocap.tutorial.dockerspring.entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

@Entity
public class User {
    @Id @GeneratedValue
    private long id;

    @Column
    private String username;
```

USE ECLIPSE SHORTCUT KEYS

1. Run Source > Generate Constructors from superclass
2. Run Source > Generate Constructor using Fields
   - select id, username, email, password
   - select username, email, password
3. Run Source > Generate Getters and Setters
   - click Select All button
4. Run Source > Generate toString
5. Run Source > Generate hashCode and equals
   - select id

REPOSITORY

Repository:

1. select ocap.tutorial.dockerspring under src/main/java directory
2. Run New > File > Interface

Property Name	Property Value
Package	ocap.tutorial.dockerspring.repo
Name	UserRepository

REPOSITORY

File:
ocap/tutorial/dockerspring/repo/UserRepository.java

```
package ocap.tutorial.dockerspring.repo;

import org.springframework.data.repository.CrudRepository;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;

import ocap.tutorial.dockerspring.entity.User;

@RepositoryRestResource
public interface UserRepository extends CrudRepository<User, Long>{
}
```

ADD JPA PROPERTIES

File: src/main/resources/application.properties

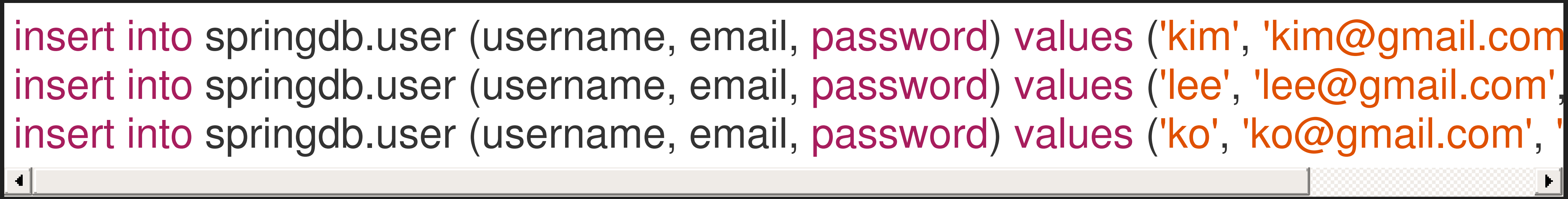
```
spring.datasource.url = jdbc:mysql://localhost:3306/springdb?useSSL=false

# Username and password
spring.datasource.username = springdb
spring.datasource.password = springdb

#JPA properties
spring.jpa.show-sql = true
spring.jpa.hibernate.ddl-auto = update
```

REBUILD

```
$ mvc clean package
$ mvc spring-boot:run
```

INSERT SAMPLE USERS

Run these sql statements through Mysql Workbench



```
insert into springdb.user (username, email, password) values ('kim', 'kim@gmail.com
insert into springdb.user (username, email, password) values ('lee', 'lee@gmail.com',
insert into springdb.user (username, email, password) values ('ko', 'ko@gmail.com', '
```

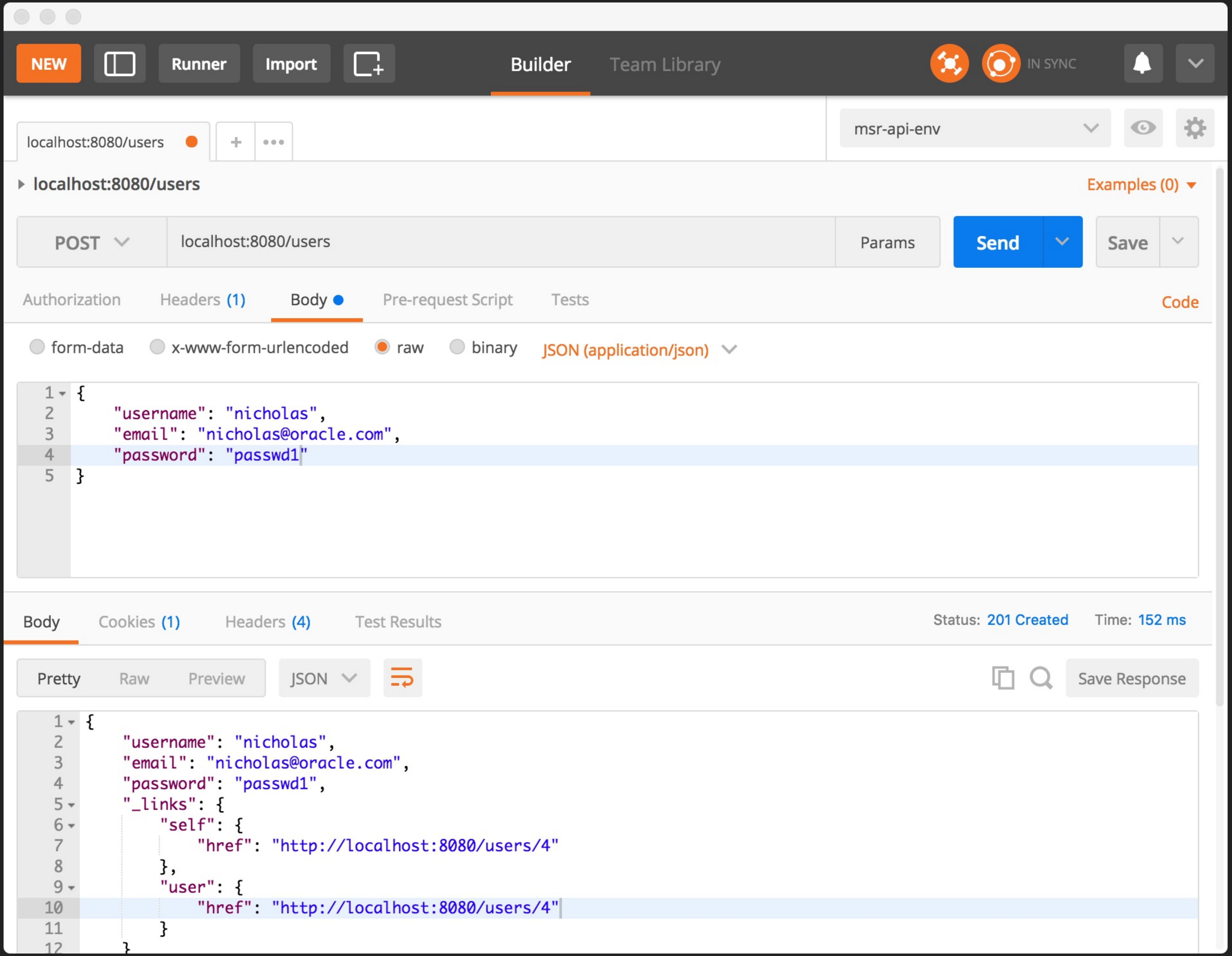
GET USER INFORMATION

- http://localhost:8080
- http://localhost:8080/users
- http://localhost:8080/users/1

CREATE USER THROUGH POSTMAN

- **HTTP Method:** POST
- **URL:** localhost:8080/users
- **Headers**
  - Content-Type: application/json
- **Body:** raw

```
{
    "username": "nicholas",
    "email": "nicholas@oracle.com",
    "password": "passwd1"
}
```



localhost:8080/users

POST localhost:8080/users

```
{
    "username": "nicholas",
    "email": "nicholas@oracle.com",
    "password": "passwd1"
}
```

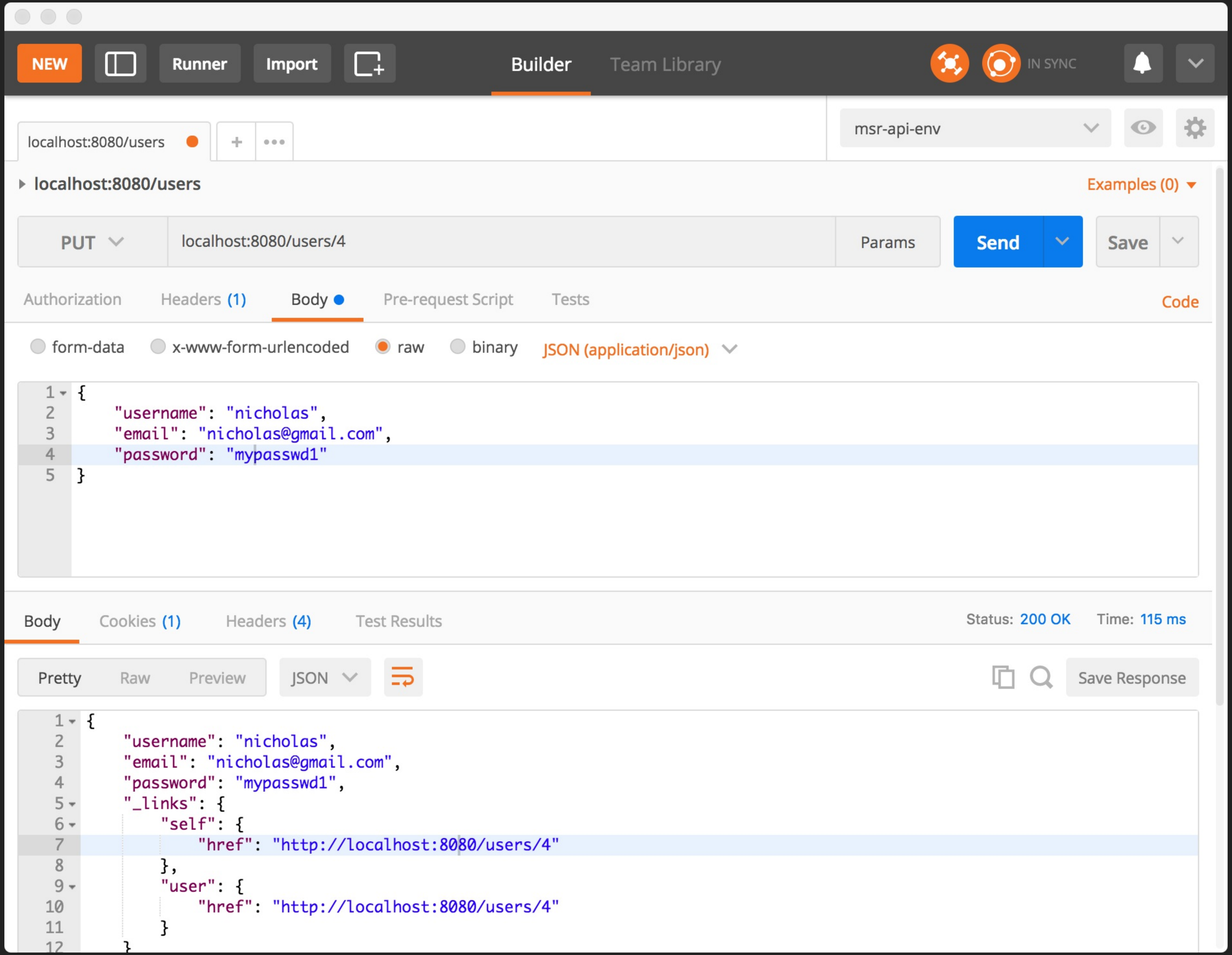
Status: 201 Created Time: 152 ms

```
{
    "username": "nicholas",
    "email": "nicholas@oracle.com",
    "password": "passwd1",
    "_links": {
        "self": {
            "href": "http://localhost:8080/users/4"
        },
        "user": {
            "href": "http://localhost:8080/users/4"
        }
    }
```

CREATE USER THROUGH POSTMAN

- **HTTP Method**: PUT
- **URL**: localhost:8080/users/4
- **Headers**
    - Content-Type: application/json
- **Body**: raw

```
{
    "username": "nicholas",
    "email": "nicholas@gmail.com",
    "password": "mypasswd1"
}
```



localhost:8080/users

PUT localhost:8080/users/4

Body: [✓ raw] JSON (application/json)

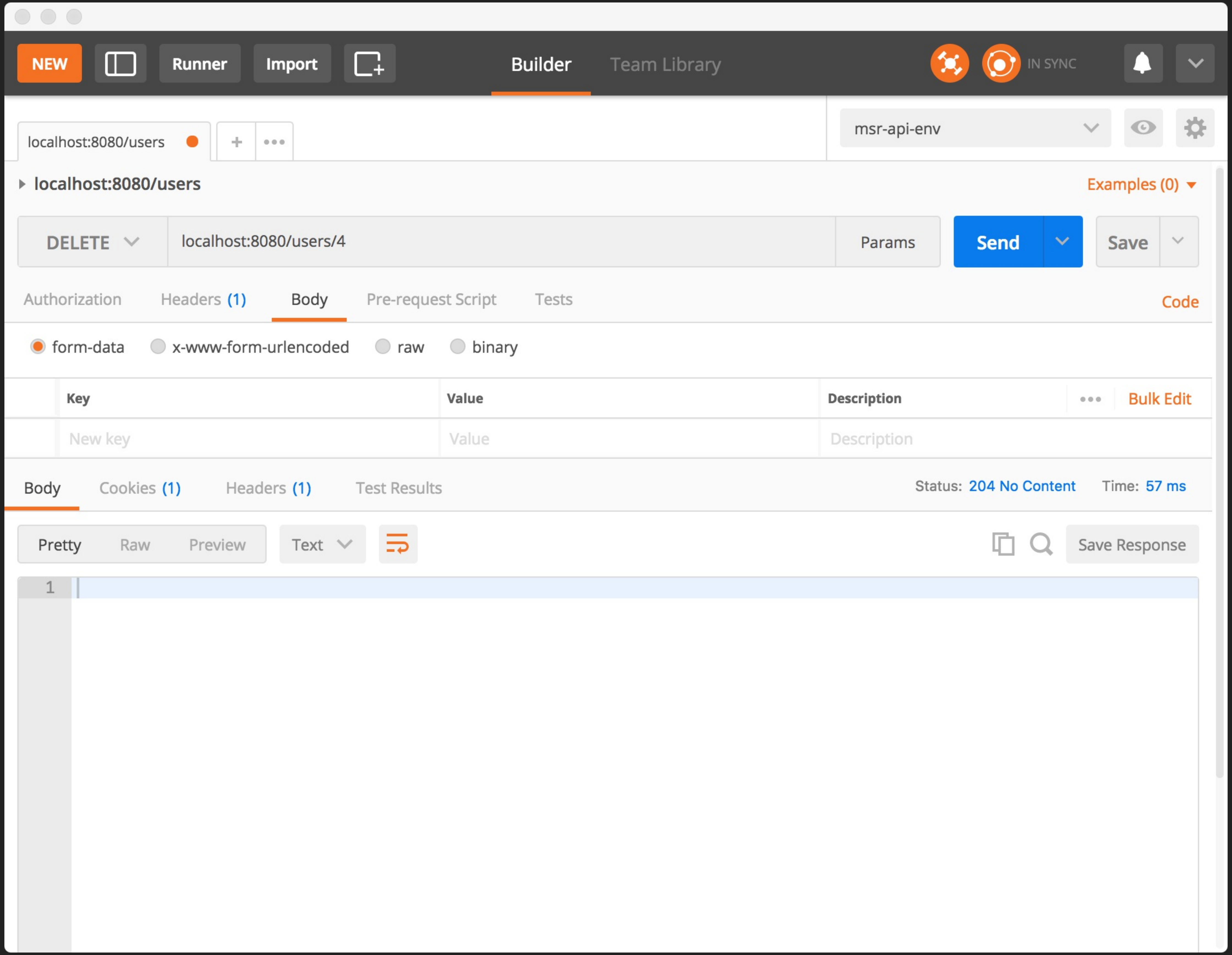
```
{
    "username": "nicholas",
    "email": "nicholas@gmail.com",
    "password": "mypasswd1"
}
```

Status: 200 OK Time: 115 ms

```
{
    "username": "nicholas",
    "email": "nicholas@gmail.com",
    "password": "mypasswd1",
    "_links": {
        "self": {
            "href": "http://localhost:8080/users/4"
        },
        "user": {
            "href": "http://localhost:8080/users/4"
        }
    }
```

DELETE USER THROUGH POSTMAN

- **HTTP Method**: DELETE
- **URL**: localhost:8080/users/4



localhost:8080/users

DELETE localhost:8080/users/4

Body: [✓ form-data] x-www-form-urlencoded raw binary

Key	Value	Description

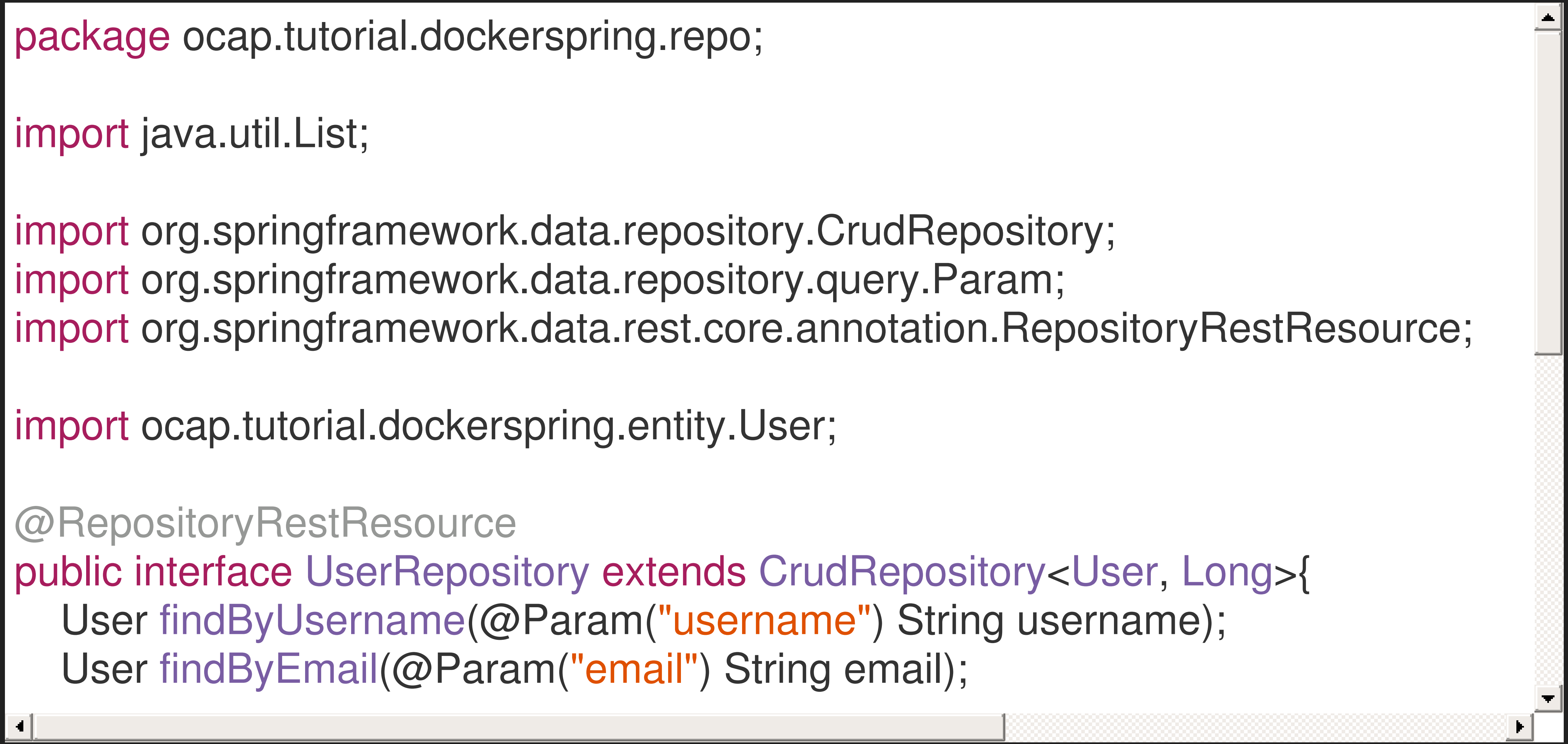
Status: 204 No Content Time: 57 ms

HTTP METHOD VS SQL

HTTP Method	SQL Statement	Note
GET	SELECT	List, One Item
POST	CREATE	
PUT	UPDATE	
DELETE	DELETE	

SOPHISTICATED QUERY USING FINDBY

File: ocap/tutorial/dockerspring/repo/UserRepository.java



```java
package ocap.tutorial.dockerspring.repo;

import java.util.List;

import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;

import ocap.tutorial.dockerspring.entity.User;

@RepositoryRestResource
public interface UserRepository extends CrudRepository<User, Long>{
    User findByUsername(@Param("username") String username);
    User findByEmail(@Param("email") String email);
```

TEST QUERY

- http://localhost:8080/users/search/findByUsername?username=kim
- http://localhost:8080/users/search/findByEmail?email=kim@gmail.com
- http://localhost:8080/users/search/findByEmailStartingWith email=k
- http://localhost:8080/users/search/findByEmailEndingWith?email=com

TEST QUERY

- http://localhost:8080/users/search/findByEmailContaining?email=gmail
- http://localhost:8080/users/search/findByEmailLike?email=%25gmail%25
- http://localhost:8080/users/search/findByUsernameAndEma
username=kim&email=kim@gmail.com
- http://localhost:8080/users/search/findByUsernameOrEmail
username=kim&email=kim@gmail.com

CONTAINERIZING

File: src/main/resources/application.properties

```
spring.datasource.url = jdbc:mysql://${SPRING_DB:localhost}:3306/springdb?useSS

# Username and password
spring.datasource.username = springdb
spring.datasource.password = springdb

#JPA properties
spring.jpa.show-sql = true
spring.jpa.hibernate.ddl-auto = update
```

DOCKERFILE

File: docker/Dockerfile-spring

```
$ mvn clean package
$ cd docker
$ rm docker-spring-tutorial-0.0.1-SNAPSHOT.jar
$ cp -f ../target/docker-spring-tutorial-0.0.1-SNAPSHOT.jar  ./
$ vi Dockerfile-spring
```

Dockerfile-spring

```
FROM openjdk:8-jdk-alpine
RUN mkdir -p /usr/src/app
COPY docker-spring-tutorial-0.0.1-SNAPSHOT.jar /usr/src/app/
CMD java -jar /usr/src/app/docker-spring-tutorial-0.0.1-SNAPSHOT.jar
EXPOSE 8080
```

BUILD & RUN SPRING APPLICATION

```
$ docker build -t spring_app -f Dockerfile-spring .
$
$ docker run -d --name spring_app -p 8080:8080 \
  --link spring_db:spring_db \
  -e SPRING_DB=spring_db spring_app
```

Q & A