Day 4 Exercises

Exercise 1

Time: 15 mins.

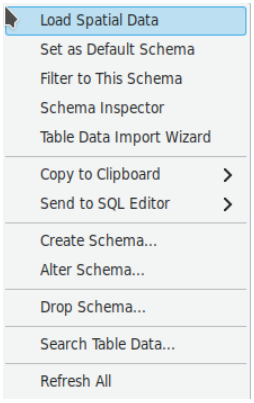
1. Download Eclipse Spring Tools
    I. Go to `http://spring.io/tools` and download the IDE
2. Verify your installation
    I. Run the eclipse ide
3. Create a SpringBoot Project
4. Review the generated code

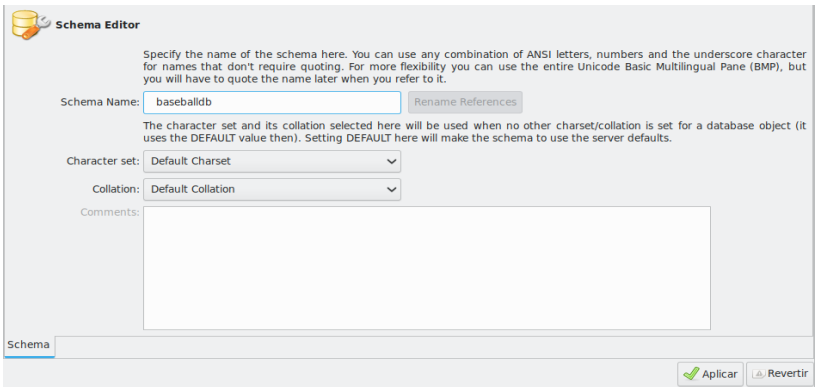
Exercise 2

Time: 15mins.

Create yor database

1. download MySQL community edition
    I. `Drop by https://dev.mysql.com/downloads/mysql/ and follw the instructions, you may need to register`
2. Run the install program
3. Download the MySQL Workbench (optional) from https://dev.mysql.com/downloads/workbench/
4. Try to connect from the workbench to the local server
5. Create a database called "**baseballdb**"
    I. If using the workbench you can use the navigator and right click, selecting "create schema"





If using the mySql CLI use the **CREATE DATABASE baseballdb** command

Exercise 3

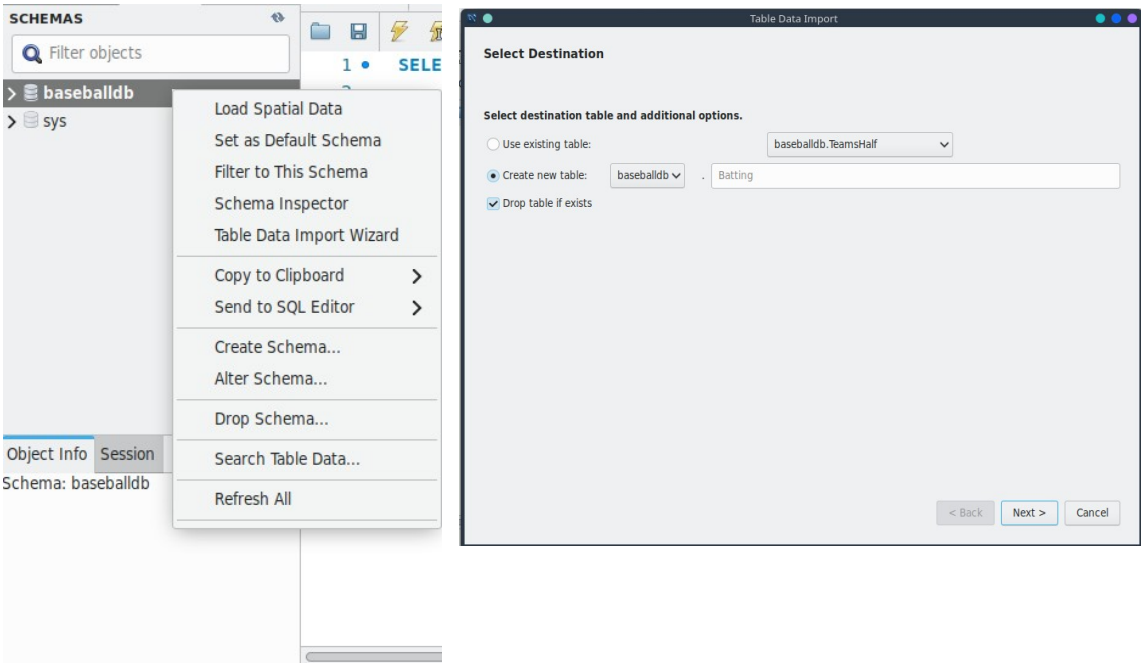
Time 15 minutes.

Upload the data to the database

Answer

Download the Baseball Data Bank

From the MySQL workbench select the table import wizard



Review the datatypes and finish the task, do the same for both Batting and Player tables. Please review that the table and field names are in lowercase.

Exercise 4

Time 120 mins.

Create a complete API application for showing the data of batting and players.

Remember to create your swaggers, and set a route to:

- show all players
- show all playera of a team in a specific year

Answer:

Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
      <modelVersion>4.0.0</modelVersion>
      <parent>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-parent</artifactId>
            <version>2.7.1</version>
            <relativePath /> <!-- lookup parent from repository -->
      </parent>
      <groupId>com.beisbolicos</groupId>
      <artifactId>PlayersStats-boot</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <name>PlayersStats-boot</name>
      <description>SpringBoot for playerStats</description>
      <properties>
            <java.version>11</java.version>
            <java.version>11</java.version>
            <springfox.swagger.version>2.9.2</springfox.swagger.version>
      </properties>
      <dependencies>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-data-jpa</artifactId>
            </dependency>
            <dependency>
                  <groupId>org.springframework</groupId>
                  <artifactId>spring-web</artifactId>

            </dependency>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-devtools</artifactId>
                  <scope>runtime</scope>
                  <optional>true</optional>
            </dependency>
            <dependency>
                  <groupId>mysql</groupId>
                  <artifactId>mysql-connector-java</artifactId>
                  <scope>runtime</scope>
            </dependency>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-configuration-processor</artifactId>
                  <optional>true</optional>
            </dependency>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-test</artifactId>
                  <scope>test</scope>
            </dependency>
```

```
		<dependency>
			<groupId>org.springframework.restdocs</groupId>
			<artifactId>spring-restdocs-mockmvc</artifactId>
			<scope>test</scope>
		</dependency>
		<dependency>
			<groupId>io.springfox</groupId>
			<artifactId>springfox-swagger2</artifactId>
			<version>${springfox.swagger.version}</version>
		</dependency>

		<dependency>
			<groupId>io.springfox</groupId>
			<artifactId>springfox-swagger-ui</artifactId>
			<version>${springfox.swagger.version}</version>
		</dependency>
	</dependencies>

	<build>
		<plugins>
			<plugin>
				<groupId>org.asciidoctor</groupId>
				<artifactId>asciidoctor-maven-plugin</artifactId>
				<version>1.5.8</version>
				<executions>
					<execution>
						<id>generate-docs</id>
						<phase>prepare-package</phase>
						<goals>
							<goal>process-asciidoc</goal>
						</goals>
						<configuration>
							<backend>html</backend>
							<doctype>book</doctype>
						</configuration>
					</execution>
				</executions>
				<dependencies>
					<dependency>
						<groupId>org.springframework.restdocs</groupId>
						<artifactId>spring-restdocs-
asciidoctor</artifactId>
						<version>${spring-restdocs.version}</version>
					</dependency>
				</dependencies>
			</plugin>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
			</plugin>
		</plugins>
	</build>

</project>
```

appliction.properties

```
server.port=2023

spring.datasource.url= jdbc:mysql://localhost:3306/baseballdb
spring.datasource.username=root
spring.datasource.password=XXXXXXX

#spring.jpa.hibernate.ddl-auto=create-drop

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
#spring.datasource.hikari.connection-timeout=60000
spring.jpa.database=mysql

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
spring.jackson.serialization.FAIL_ON_EMPTY_BEANS=false
```

DataStoreSetup.java

```java
package com.beisbolicos.playerStats;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.autoconfigure.domain.EntityScan;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;
import org.springframework.jdbc.datasource.DriverManagerDataSource;

@Configuration
@EnableJpaRepositories(basePackages = { "com.beisbolicos.playerStats.repo" })
@ComponentScan(value = "com.beisbolicos.playerStats.*")
@EntityScan(basePackages = { "com.beisbolicos.playerStats.entity" })
public class DataStoreSetup {

	@Value("${spring.datasource.url}")
	String databaseUrl;

	@Value("${spring.datasource.username}")
	String databaseUser;

	@Value("${spring.datasource.password}")
	String databasePassword;

	@Bean
	public DataSource dataSource() {

		DriverManagerDataSource dataSource = new DriverManagerDataSource();
		dataSource.setUrl(databaseUrl);
		dataSource.setUsername(databaseUser);
		dataSource.setPassword(databasePassword);
		return dataSource;
	}

}
```

PlayersStatBootApplication.java

```java
package com.beisbolicos.playerStats;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.RequestMapping;

@SpringBootApplication
public class PlayersStatsBootApplication {

	public static void main(String[] args) {
		SpringApplication.run(PlayersStatsBootApplication.class, args);
	}

	@RequestMapping(path = "/")
	public String helloWorld() {
		return "Hello World";
	}
}
```

PeopleController.java

```java
package com.beisbolicos.playerStats.controller;
//import java.util.ArrayList;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.beisbolicos.playerStats.entity.People;
import com.beisbolicos.playerStats.serviceImpl.PeopleService;

import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;
import io.swagger.annotations.Example;
import io.swagger.annotations.ExampleProperty;
import io.swagger.annotations.SwaggerDefinition;
import io.swagger.annotations.Tag;

@RestController
@SwaggerDefinition(tags = {@Tag (name="Consultas", description="Diferentes
consultas para la BDB")})

public class PeopleController {

	@Autowired
	PeopleService peopleService;
```

```java
	@PostMapping(value = "/people")


	public ResponseEntity<Object> createEmployee(@RequestBody People people) {

		peopleService.createPeople(people);
		return new ResponseEntity<Object>("Successfully Saved", HttpStatus.OK);
	}

	@GetMapping(value = "/people/{id}")
	/**
	 * Obtiene los datos generales del jugador de acuerdo a su id
	 * @param id Generalmente las 5 primeras letras del apellido más dos del nombre y dos dígitos de secuencia por colisión
	 * @return Datos generales del jugador o manager
	 */
	@ApiOperation(value = "Queries a people on the Baseball Data Bank",
	              notes = "Queries a people on the Baseball Data Bank")
	@ApiResponses(value = {
	            @ApiResponse(code = 200, message = "Successfully got the people",
	                         examples = @Example(value = @ExampleProperty(mediaType="application/json",
	                                                value="{\"playerId\":\"allenjo02\","
	                                                        + "\"nameFirst\":\"Johnny\","
	                                                        + "\"nameLast\":\"Allen\","
	                                                        + "\"birthCity\":\"Lenoir\","
	                                                        + "\"birthContry\":\"USA\","
	                                                        + "\"birthDay\":\"30\","
	                                                        + "\"birthMonth\":\"9\","
	                                                        + "\"birthYear\":\"1904\"}"))),
	                @ApiResponse(code = 404, message = "Player or manager not found"),
	                @ApiResponse(code = 400, message = "Missing or invalid request body"),
	                @ApiResponse(code = 500, message = "Internal error")
	                }
	)
	public ResponseEntity<Object> getPeople(@PathVariable String id) {

		People people = peopleService.getPeopleById(id);
		return new ResponseEntity<Object>(people, HttpStatus.OK);
	}

	@PutMapping(value = "/people")
	public ResponseEntity<Object> updateEmployee(@RequestBody People people) {

		peopleService.updatePeople(people);
		return new ResponseEntity<Object>("Successfully Updated", HttpStatus.OK);
	}

	@DeleteMapping(value = "/people/{id}")
	public ResponseEntity<Object> deleteEmployee(@PathVariable String id) {

		peopleService.deletePeople(id);
		return new ResponseEntity<Object>("Successfully Deleted", HttpStatus.OK);
```

```
	}

	@GetMapping(value="/people")
	public ResponseEntity <Object> getPeople(){
		List<People> people = peopleService.getPeople();
		ResponseEntity<Object> listPeople = new ResponseEntity<>(people,
HttpStatus.OK);
		//for(People item : people) {
		//    listPeople.add(new ResponseEntity<Object>(status))
		//}
		return listPeople;
	}

	@GetMapping(value="/people/{teamId}/{yearId}")
	public ResponseEntity <Object> getPeople(@PathVariable String teamId,
@PathVariable int yearId){
		List<People> people = peopleService.getPeople(teamId, yearId);
		ResponseEntity<Object> listPeople = new ResponseEntity<Object>(people,
HttpStatus.OK);
		return listPeople;
	}

}
```

Batting.java

```
package com.beisbolicos.playerStats.entity;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.IdClass;
import javax.persistence.Table;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@Entity
@IdClass(BattingKey.class)
@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
@Table(name="batting")
public class Batting implements Serializable{
	/**
	 *
	 */
	private static final long serialVersionUID = 1L;
	@Id
	@Column(name = "playerid", nullable = false)
	private String playerId;
	@Id
	@Column(name = "yearid", nullable = false)
	private int yearId;
	@Id
	@Column(name = "stint")
	private String stint;
	@Id
	@Column(name = "teamid", nullable = false)
```

```
private String teamId;
@Column(name="lgid")
private String lgId;
@Column(name="G")
private int games;
@Column(name="ab")
private int atBat;
@Column(name="r")
private int runs;
@Column(name="h")
private int hits;
@Column(name="2b")
private int doubleHits;
@Column(name="3b")
private int tripeHits;
@Column(name = "hr")
private int homeRuns;
@Column(name = "rbi")
private int runsBattedIn;
@Column(name = "sb")
private int stolenBases;
@Column(name = "cs")
private int caughtStealing;
@Column(name = "bb")
private int baseOnBalls;
@Column(name = "so")
private int strikedut;
@Column(name = "ibb")
private int intentionalBaseOnBalls;
@Column(name="hbp")
private int hitByPitch;
@Column(name="sh")
private int sacrificeHit;
@Column(name="sf")
private int sacrificeFly;
@Column(name="gidp")
private int groundInDoublePlay;



public String getPlayerId() {
      return playerId;
}



public void setPlayerId(String playerId) {
      this.playerId = playerId;
}



public int getYearId() {
      return yearId;
}
```

```java
public void setYearId(int yearId) {
	this.yearId = yearId;
}



public String getStint() {
	return stint;
}



public void setStint(String stint) {
	this.stint = stint;
}



public String getTeamId() {
	return teamId;
}



public void setTeamId(String teamId) {
	this.teamId = teamId;
}



public String getLgId() {
	return lgId;
}



public void setLgId(String lgId) {
	this.lgId = lgId;
}



public int getGames() {
	return games;
}



public void setGames(int games) {
	this.games = games;
}



public int getAtBat() {
	return atBat;
}
```

```
public void setAtBat(int atBat) {
	this.atBat = atBat;
}



public int getRuns() {
	return runs;
}



public void setRuns(int runs) {
	this.runs = runs;
}



public int getHits() {
	return hits;
}



public void setHits(int hits) {
	this.hits = hits;
}



public int getDoubleHits() {
	return doubleHits;
}



public void setDoubleHits(int doubleHits) {
	this.doubleHits = doubleHits;
}



public int getTripeHits() {
	return tripeHits;
}



public void setTripeHits(int tripeHits) {
	this.tripeHits = tripeHits;
}



public int getHomeRuns() {
	return homeRuns;
```

```
}


public void setHomeRuns(int homeRuns) {
	this.homeRuns = homeRuns;
}


public int getRunsBattedIn() {
	return runsBattedIn;
}


public void setRunsBattedIn(int runsBattedIn) {
	this.runsBattedIn = runsBattedIn;
}


public int getStolenBases() {
	return stolenBases;
}


public void setStolenBases(int stolenBases) {
	this.stolenBases = stolenBases;
}


public int getCaughtStealing() {
	return caughtStealing;
}


public void setCaughtStealing(int caughtStealing) {
	this.caughtStealing = caughtStealing;
}


public int getBaseOnBalls() {
	return baseOnBalls;
}


public void setBaseOnBalls(int baseOnBalls) {
	this.baseOnBalls = baseOnBalls;
}
```

```java
public int getStrikedut() {
	return strikedut;
}



public void setStrikedut(int strikedut) {
	this.strikedut = strikedut;
}



public int getIntentionalBaseOnBalls() {
	return intentionalBaseOnBalls;
}



public void setIntentionalBaseOnBalls(int intentionalBaseOnBalls) {
	this.intentionalBaseOnBalls = intentionalBaseOnBalls;
}



public int getHitByPitch() {
	return hitByPitch;
}



public void setHitByPitch(int hitByPitch) {
	this.hitByPitch = hitByPitch;
}



public int getSacrificeHit() {
	return sacrificeHit;
}



public void setSacrificeHit(int sacrificeHit) {
	this.sacrificeHit = sacrificeHit;
}



public int getSacrificeFly() {
	return sacrificeFly;
}



public void setSacrificeFly(int sacrificeFly) {
	this.sacrificeFly = sacrificeFly;
}
```

```
	public int getGroundInDoublePlay() {
		return groundInDoublePlay;
	}



	public void setGroundInDoublePlay(int groundInDoublePlay) {
		this.groundInDoublePlay = groundInDoublePlay;
	}



	@Override
	public String toString() {
		return "Batting [id=" + playerId + ", team=" + teamId + ", Year=" +
yearId + ", at Bat="
				+ atBat + ", Hits=" + hits +"]";
	}
}
```

BattingKey.java

```
package com.beisbolicos.playerStats.entity;

import java.io.Serializable;

public class BattingKey implements Serializable{

		private String playerId;
		private int yearId;
		private String stint;
		private String teamId;

		public String getPlayerId() {
			return playerId;
		}

		public void setPlayerId(String playerId) {
			this.playerId = playerId;
		}

		public int getYearId() {
			return yearId;
		}

		public void setYearId(int yearId) {
			this.yearId = yearId;
		}

		public String getStint() {
			return stint;
		}

		public void setStint(String stint) {
			this.stint = stint;
		}
```

```
	public String getTeamId() {
		return teamId;
	}

	public void setTeamId(String teamId) {
		this.teamId = teamId;
	}

}
```

People.java

```
package com.beisbolicos.playerStats.entity;


import java.io.Serializable;


import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;


import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@Entity

@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})

@Table(name="people")

public class People implements Serializable{

		/**

		 *

		 */

		private static final long serialVersionUID = 1L;


		@Id

		@Column(name = "playerid", nullable = false, unique = true)

		private String playerId;
```

```
@Column(name="namefirst" )
private String nameFirst;

@Column(name="namelast")
private String nameLast;

@Column(name="birthcity")
private String birthCity;

@Column(name="birthcountry")
private String birthContry;

@Column(name="birthday")
private String birthDay;

@Column(name="birthmonth")
private String birthMonth;

@Column(name="birthyear")
private String birthYear;



public String getPlayerId() {
        return playerId;
}
```

```
public void setPlayerId(String playerId) {
	this.playerId = playerId;
}



public String getNameFirst() {
	return nameFirst;
}



public void setNameFirst(String nameFirst) {
	this.nameFirst = nameFirst;
}



public String getNameLast() {
	return nameLast;
}



public void setNameLast(String nameLast) {
	this.nameLast = nameLast;
}
```

```
public String getBirthCity() {
	return birthCity;
}



public void setBirthCity(String birthCity) {
	this.birthCity = birthCity;
}



public String getBirthContry() {
	return birthContry;
}



public void setBirthContry(String birthContry) {
	this.birthContry = birthContry;
}



public String getBirthDay() {
	return birthDay;
}
```

```
public void setBirthDay(String birthDay) {
        this.birthDay = birthDay;
}



public String getBirthMonth() {
        return birthMonth;
}



public void setBirthMonth(String birthMonth) {
        this.birthMonth = birthMonth;
}



public String getBirthYear() {
        return birthYear;
}



public void setBirthYear(String birthYear) {
        this.birthYear = birthYear;
}
```

```
	public static long getSerialversionuid() {
		return serialVersionUID;
	}



	@Override
	public String toString() {
		return "player [id=" + playerId + ", firstName=" + nameFirst + ", lastName=" + nameLast + ", birthCity="
				+ birthCity + ", birthCountry=" + birthContry + ", birthDate=" + birthDay + "/"+birthMonth+"/"+birthYear+"]";
	}
}
```

BattingRepository.java

```
package com.beisbolicos.playerStats.repo;

import org.springframework.data.jpa.repository.JpaRepository;

import com.beisbolicos.playerStats.entity.Batting;

public interface BattingRepository extends JpaRepository<Batting, String>
{

}
```

PeopleRepository

```
package com.beisbolicos.playerStats.repo;

import org.springframework.stereotype.Repository;

import com.beisbolicos.playerStats.entity.People;

import java.util.List;
```

```java
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

@Repository
public interface PeopleRepository extends JpaRepository<People, String>{

	@Query("select p  from "
			+ "People p, Batting b where p.playerId=b.playerId"
			+ " and b.teamId = ?1 and b.yearId = ?2")
	public List<People> findByTeamYear(String teamId, int yearId);

}
```

IpeopleService.java

```
package com.beisbolicos.playerStats.service;


import java.util.List;


import com.beisbolicos.playerStats.entity.People;


public interface IPeopleService {

	public void createPeople(People employee);


	public People getPeopleById(String id);


	public void updatePeople(People employee);


	public void deletePeople(String id);


	public List<People> getPeople();


}
```

PeopleService.java

```
package com.beisbolicos.playerStats.serviceImpl;
```

```
import java.util.List;

import javax.persistence.EntityNotFoundException;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.beisbolicos.playerStats.entity.People;
import com.beisbolicos.playerStats.repo.PeopleRepository;
import com.beisbolicos.playerStats.service.IPeopleService;

@Service
public class PeopleService implements IPeopleService {
	@Autowired
	PeopleRepository peopleRepository;

	@Override
	public void createPeople(People employee) {
		peopleRepository.save(employee);

	}

	@Override
	public People getPeopleById(String id) {
		People people;
		try {
			people = peopleRepository.getById(id);
		}catch (EntityNotFoundException e){
			people = null;
		}
```

```
        return people;
    }

    @Override
    public void updatePeople(People employee) {
        peopleRepository.save(employee);

    }

    @Override
    public void deletePeople(String id) {
        peopleRepository.deleteById(id);

    }

    @Override
    public List<People> getPeople() {
        List<People> people = peopleRepository.findAll();
        return people;
    }

    public List<People> getPeople(String teamId, int yearId) {
        List<People>people = peopleRepository.findByTeamYear(teamId, yearId);
        return people;
    }

}
```