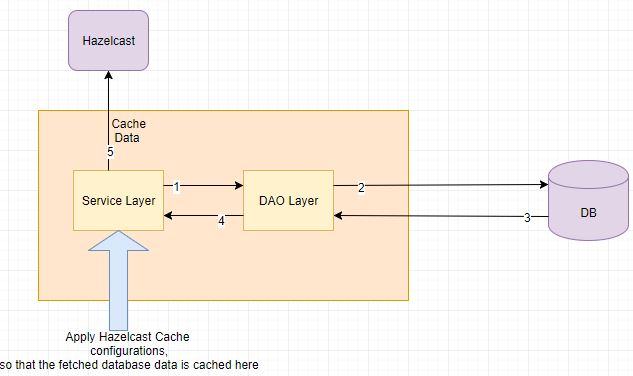
**Spring Boot + Hazelcast Simple Example**

In this post we develop a Spring Boot Application with Hazelcast. Previously we had developed a [Spring Boot Application for performing JDBC operations](https://www.javainuse.com/spring/bootjdbc). We will developing a similar module in this example and using Hazelcast to cache the data.

What is Hazelcast? Need for it?

* Hazelcast is a radical, new approach towards data that was designed from the ground up around distribution. It embraces a new, scalable way of thinking in that data should be shared for resilience and performance while allowing us to configure the trade-offs surrounding consistency, as the data requirements dictate. Hazelcast is a distributed, highly available and scalable . [You can find other details and tutorials of Hazelcast here.](https://www.javainuse.com/hazelcast)

Lets Begin-

The overview of application we will be developing is as follows-  
  
Maven Project will be as follows-

In the Maven we need the spring boot test dependency.Maven will be as follows-

<?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 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.javainuse</groupId>

<artifactId>boot-hazelcast</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>boot-jdbc</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.5.2.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-jdbc</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

**<dependency>**

**<groupId>org.springframework.boot</groupId>**

**<artifactId>spring-boot-starter-cache</artifactId>**

**</dependency>**

**<dependency>**

**<groupId>com.hazelcast</groupId>**

**<artifactId>hazelcast</artifactId>**

**</dependency>**

**<dependency>**

**<groupId>com.hazelcast</groupId>**

**<artifactId>hazelcast-spring</artifactId>**

**</dependency>**

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

In the application.properties file specify the datasource properties

spring.datasource.url=jdbc:h2:file:./DB

spring.datasource.platform=hsqldb

As we saw in the Spring Boot JDBC tutorial that any db initialization scripts stored in schema-.sql gets executed automatically.  
Since we using HSQL create the schema-hsqldb.sql file and specify the initialization scripts-

DROP TABLE IF EXISTS employee;

CREATE TABLE employee (

empId VARCHAR(10) NOT NULL,

empName VARCHAR(100) NOT NULL

);

Create the Employee Domain class

package com.javainuse.model;

public class Employee {

private String empId;

private String empName;

public String getEmpId() {

return empId;

}

public void setEmpId(String empId) {

this.empId = empId;

}

public String getEmpName() {

return empName;

}

public void setEmpName(String empName) {

this.empName = empName;

}

@Override

public String toString() {

return "Employee [empId=" + empId + ", empName=" + empName + "]";

}

}

Create the Hazelcast configuration class as follows-

package com.javainuse;

import com.hazelcast.config.Config;

import com.hazelcast.config.EvictionPolicy;

import com.hazelcast.config.MapConfig;

import com.hazelcast.config.MaxSizeConfig;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class HazelcastConfiguration {

**@Bean**

**public Config hazelCastConfig(){**

**return new Config()**

**.setInstanceName("hazelcast-instance")**

**.addMapConfig(**

**new MapConfig()**

**.setName("employees")**

**.setMaxSizeConfig(new MaxSizeConfig(200, MaxSizeConfig.MaxSizePolicy.FREE\_HEAP\_SIZE))**

**.setEvictionPolicy(EvictionPolicy.LRU)**

**.setTimeToLiveSeconds(20));**

**}**

}

Create Service interface to specify employee operations to be performed.

package com.javainuse.service;

import java.util.List;

import com.javainuse.model.Employee;

public interface EmployeeService {

void insertEmployee(Employee emp);

void insertEmployees(List<Employee> employees);

void getAllEmployees();

void getEmployeeById(String empid);

}

The Service class implementation with hazelcast configurations.

package com.javainuse.service.impl;

import java.util.List;

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

import org.springframework.cache.annotation.CacheConfig;

import org.springframework.cache.annotation.Cacheable;

import org.springframework.stereotype.Service;

import com.javainuse.dao.EmployeeDao;

import com.javainuse.model.Employee;

import com.javainuse.service.EmployeeService;

@Service

**@CacheConfig(cacheNames = "employees")**;

public class EmployeeServiceImpl implements EmployeeService {

@Autowired

EmployeeDao employeeDao;

@Override

public void insertEmployee(Employee employee) {

employeeDao.insertEmployee(employee);

}

@Override

public void insertEmployees(List<Employee> employees) {

employeeDao.insertEmployees(employees);

}

@Override

**@Cacheable()**

public List<Employee> getAllEmployees() {

System.out.println("Inside the service layer");

return employeeDao.getAllEmployees();

}

@Override

public void getEmployeeById(String empId) {

Employee employee = employeeDao.getEmployeeById(empId);

System.out.println(employee);

}

}

Create the DAO interface.

package com.javainuse.dao;

import java.util.List;

import com.javainuse.model.Employee;

public interface EmployeeDao {

void insertEmployee(Employee cus);

void insertEmployees(List<Employee> employees);

List<Employee> getAllEmployees();

Employee getEmployeeById(String empId);

}

The DAO implementation class.

package com.javainuse.dao.impl;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

import javax.annotation.PostConstruct;

import javax.sql.DataSource;

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

import org.springframework.jdbc.core.BatchPreparedStatementSetter;

import org.springframework.jdbc.core.RowMapper;

import org.springframework.jdbc.core.support.JdbcDaoSupport;

import org.springframework.stereotype.Repository;

import com.javainuse.dao.EmployeeDao;

import com.javainuse.model.Employee;

@Repository

public class EmployeeDaoImpl extends JdbcDaoSupport implements EmployeeDao{

@Autowired

DataSource dataSource;

@PostConstruct

private void initialize(){

setDataSource(dataSource);

}

@Override

public void insertEmployee(Employee emp) {

String sql = "INSERT INTO employee " +

"(empId, empName) VALUES (?, ?)" ;

getJdbcTemplate().update(sql, new Object[]{

emp.getEmpId(), emp.getEmpName()

});

}

@Override

public void insertEmployees(List<Employee> employees) {

String sql = "INSERT INTO employee " + "(empId, empName) VALUES (?, ?)";

getJdbcTemplate().batchUpdate(sql, new BatchPreparedStatementSetter() {

public void setValues(PreparedStatement ps, int i) throws SQLException {

Employee employee = employees.get(i);

ps.setString(1, employee.getEmpId());

ps.setString(2, employee.getEmpName());

}

public int getBatchSize() {

return employees.size();

}

});

}

@Override

public List<Employee> getAllEmployees(){

String sql = "SELECT \* FROM employee";

List<Map<String, Object>> rows = getJdbcTemplate().queryForList(sql);

List<Employee> result = new ArrayList<Employee>();

for(Map<String, Object> row:rows){

Employee emp = new Employee();

emp.setEmpId((String)row.get("empId"));

emp.setEmpName((String)row.get("empName"));

result.add(emp);

}

return result;

}

@Override

public Employee getEmployeeById(String empId) {

String sql = "SELECT \* FROM employee WHERE empId = ?";

return (Employee)getJdbcTemplate().queryForObject(sql, new Object[]{empId}, new RowMapper<Employee>(){

@Override

public Employee mapRow(ResultSet rs, int rwNumber) throws SQLException {

Employee emp = new Employee();

emp.setEmpId(rs.getString("empId"));

emp.setEmpName(rs.getString("empName"));

return emp;

}

});

}

}

Finally create the class with @SpringBootApplication and EnableCaching annotation.

package com.javainuse;

import java.util.ArrayList;

import java.util.List;

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

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cache.annotation.EnableCaching;

import org.springframework.context.ApplicationContext;

import com.javainuse.model.Employee;

import com.javainuse.service.EmployeeService;

@SpringBootApplication

**@EnableCaching**

public class SpringBootJdbcApplication {

@Autowired

EmployeeService employeeService;

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(SpringBootJdbcApplication.class, args);

EmployeeService employeeService = context.getBean(EmployeeService.class);

Employee emp= new Employee();

emp.setEmpId("emp");

emp.setEmpName("emp");

Employee emp1= new Employee();

emp1.setEmpId("emp1");

emp1.setEmpName("emp1");

Employee emp2= new Employee();

emp2.setEmpId("emp2");

emp2.setEmpName("emp2");

employeeService.insertEmployee(emp);

List<Employee> employees = new ArrayList<>();

employees.add(emp1);

employees.add(emp2);

employeeService.insertEmployees(employees);

System.out.println("Inside the main class making call to service first time");

List<Employee> employeeList1 = employeeService.getAllEmployees();

for (Employee employee : employeeList1) {

System.out.println(employee.toString());

}

System.out.println("Inside the main class making call to service second time where it will use hazelcast");

List<Employee> employeeList2 = employeeService.getAllEmployees();

for (Employee employee : employeeList2) {

System.out.println(employee.toString());

}

}

}

Start the application-  
  
