Jeremy Cooley

4/27/2020

ITEC 4860 Spring 2020 – Dr. Im

Project 3 Code

**VehiclesProjectMavenApplication.java**

package jcooley1.demo;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.annotation.EnableAspectJAutoProxy;  
import org.springframework.scheduling.annotation.EnableScheduling;  
  
  
@SpringBootApplication  
@EnableScheduling  
@EnableAspectJAutoProxy  
public class VehiclesProjectMavenApplication {  
  
 @Log  
 @Timed  
 public static void main(String[] args) {  
 SpringApplication.*run*(VehiclesProjectMavenApplication.class, args);  
 }  
}

**Vehicle.java**

package jcooley1.demo;  
  
import javax.persistence.\*;  
import org.jetbrains.annotations.NotNull;  
@Entity  
@Table (name = "vehicles")  
public class Vehicle implements Comparable<Vehicle>  
{  
 @Id  
 @GeneratedValue (strategy = GenerationType.*AUTO*)  
 private long id;  
 private String makeModel;  
 private int year;  
 private double retailPrice;  
  
  
 public Vehicle() {  
 this.id = 0;  
 this.makeModel = "";  
 this.year = 0;  
 this.retailPrice = 0.0;  
 }  
  
 public Vehicle(long id, String makeModel, int year, double retailPrice) {  
 this.id = id;  
 this.makeModel = makeModel;  
 this.year = year;  
 this.retailPrice = retailPrice;  
 }  
  
  
 public long getId() { return id; }  
 public void setId(int id) { this.id = id; }  
  
  
 public String getMakeModel() { return makeModel; }  
 public void setMakeModel(String makeModel) {  
 this.makeModel = makeModel;  
 }  
  
 public int getYear() { return year; }  
 public void setYear(int year) { this.year = year; }  
  
  
 public double getRetailPrice() { return retailPrice; }  
 public void setRetailPrice(double retailPrice){  
 this.retailPrice = retailPrice;  
 }  
  
 @Override  
 public int compareTo(@NotNull Vehicle o) {  
 return (int) (this.id - o.getId());  
 }  
  
}

**VehicleRESTController.java**

package jcooley1.demo;  
  
import java.io.IOException;  
import java.util.ArrayList;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
  
@RestController  
public class VehicleRESTController  
{  
  
 private VehicleDAO vehicleDAO;  
  
  
 @Log  
 @Timed  
 @RequestMapping(value="/addVehicle", method=RequestMethod.*POST*)  
 public Vehicle addVehicle(@RequestBody Vehicle newVehicle) throws IOException  
 {  
 vehicleDAO.create(newVehicle);  
 return newVehicle;  
 }  
  
  
 @Log  
 @Timed  
 @RequestMapping(value="/getVehicle/{id}", method=RequestMethod.*GET*)  
 public Vehicle getVehicle(@PathVariable("id") int id) throws IOException  
 {  
 return vehicleDAO.getByID(id);  
 }  
  
  
 @Log  
 @Timed  
 @RequestMapping(value = "/getNumberOfVehicles", method = RequestMethod.*GET*)  
 public int getNumberOfVehicles() throws IOException  
 {  
 // gets the size of the inventory  
 return vehicleDAO.getCount();  
 }  
  
  
  
 @Log  
 @Timed  
 @RequestMapping(value = "/updateVehicle", method = RequestMethod.*PUT*)  
 public Vehicle updateVehicle(@RequestBody Vehicle newVehicle) throws IOException  
 {  
 vehicleDAO.update(newVehicle);  
 return newVehicle;  
 }  
  
  
 @Log  
 @Timed  
 @CheckBeforeDelete  
 @RequestMapping(value = "/deleteVehicle/{id}", method = RequestMethod.*DELETE*)  
 public ResponseEntity<String> deleteVehicle(@PathVariable("id") int id) throws IOException  
 {  
 // saves time by exiting early if id is invalid  
 if(id==0)  
 {  
 return new ResponseEntity<String>("Sorry vehicle ID does not exist", HttpStatus.*NOT\_FOUND*);  
 }  
  
  
 // tries to find the vehicle  
 Vehicle v = null;  
 v = vehicleDAO.getByID(id);  
  
  
  
 if (v != null)  
 {  
 vehicleDAO.delete(id);  
 return new ResponseEntity<String>("Vehicle deleted", HttpStatus.*FOUND*);  
 }  
 else  
 {  
 return new ResponseEntity<String>("Sorry vehicle ID not found", HttpStatus.*NOT\_FOUND*);  
 }  
 }  
  
  
 @Log  
 @Timed  
 @RequestMapping(value = "/getLatestVehicles", method = RequestMethod.*GET*)  
 public ArrayList<Vehicle> getLatestVehicles() throws IOException  
 {  
 // creates a new arraylist  
 ArrayList<Vehicle> list = new ArrayList<>();  
  
 // gets the size of the inventory  
 int count = vehicleDAO.getCount();  
  
 if (count >= 10)  
 {  
 // gets each of the last ten vehicles in the inventory  
 for(int i = count - 9; i <= count; i++)  
 {  
 // and adds them to our arraylist  
 list.add( vehicleDAO.getByID(i) );  
 }  
 }  
 else  
 {  
 System.*out*.println("Sorry, there aren't enough vehicles yet!");  
 }  
  
 return list;  
 }  
  
  
}

**VehicleDAO.java**

package jcooley1.demo;  
  
import java.math.BigInteger;  
import javax.persistence.EntityManager;  
import javax.persistence.PersistenceContext;  
import javax.persistence.Query;  
import org.springframework.stereotype.Repository;  
import org.springframework.transaction.annotation.Transactional;  
  
  
  
*/\*\*  
 \* Data Access Object - provide some specific data operations without exposing details of the database  
 \* Access data for the Vehicle entity.  
 \* Repository annotation allows Spring to find and configure the DAO.  
 \* Transactional annonation will cause Spring to call begin() and commit()  
 \* at the start/end of the method. If exception occurs it will also call rollback().  
 \*/*@Repository  
@Transactional  
public class VehicleDAO  
{  
 // PersistenceContext annotation used to specify there is a database source.  
 // EntityManager is used to create and remove persistent entity instances,  
 // to find entities by their primary key, and to query over entities.  
  
 @PersistenceContext  
 private EntityManager entityManager;  
  
  
  
 // CREATE aka PERSIST - insert vehicle into the db  
 public void create(Vehicle v) {  
 entityManager.persist(v);  
 }  
  
  
  
 // READ aka FIND - return the vehicle that matches the id  
 public Vehicle getByID(int id) {  
 return entityManager.find(Vehicle.class, id);  
 }  
  
  
  
 // UPDATE aka MERGE - update the vehicle in the db  
 public void update(Vehicle v) {  
 entityManager.merge(v);  
 }  
  
  
  
 // DELETE aka REMOVE - remove the vehicle from the db  
 public void delete(int id) {  
 Vehicle v = entityManager.find(Vehicle.class,id);  
 if (v != null) {  
 entityManager.remove(v);  
 }  
 }  
  
  
  
 public int getCount() {  
 // create the query ON the entity manager  
 Query q = entityManager.createNativeQuery("select *count*(\*) from vehicles");  
 // then save and return the resulting number  
 BigInteger count = (BigInteger) q.getSingleResult();  
 return count.intValue();  
 }  
   
 // This doesn't work, but hey, maybe I tried!  
 //  
 // public int getNumOfFourRunners() {  
 // // create the query ON the entity manager  
 // Query q =  
 // entityManager.createNativeQuery("select count(\*) from vehicles where makeModel = :makeModel").setParameter("Four-Runner",  
 // makeModel).getSingleResult();  
 // // then save and return the resulting number  
 // BigInteger count = (BigInteger) q.getSingleResult();  
 // return count.intValue();  
 // }  
   
}

**MyTasks.java**

package jcooley1.demo;  
  
import java.util.ArrayList;  
import org.apache.commons.lang3.RandomStringUtils;  
import org.apache.commons.lang3.RandomUtils;  
import org.springframework.scheduling.annotation.Scheduled;  
import org.springframework.stereotype.Component;  
import org.springframework.web.client.RestTemplate;  
  
  
  
@Component  
public class MyTasks  
{  
 private RestTemplate restTemplate = new RestTemplate();  
 private int id = 1;  
  
  
 @Log  
 @Timed  
 @Scheduled (fixedRate = 10000) // every ten seconds  
 public void addVehicle()  
 {  
 // URL that's mapped to the method we want  
 String postURL = "http://localhost:8080/addVehicle";  
  
 // Create the Vehicle Object  
 int year = RandomUtils.*nextInt*(1986,2017);  
 double price = (double) RandomUtils.*nextInt*(15000,45001);  
 String makeModel = RandomStringUtils.*randomAlphabetic*(5,10);  
 Vehicle v = new Vehicle(id++, makeModel, year, price);  
  
 // Execute on RestTemplate  
 restTemplate.postForObject(postURL,v,Vehicle.class);  
 }  
  
  
 @Log  
 @Timed  
 @Scheduled (cron = "0 \*/2 \* \* \* \*") // every other minute  
 public void deleteVehicle()  
 {  
 // Generate a random ID  
 int deleteID = RandomUtils.*nextInt*(0,5);  
  
 // Create the URL that takes in the ID as a parameter  
 String deleteURL = "http://localhost:8080/deleteVehicle{" + deleteID + "}";  
 String getURL = "http://localhost:8080/getVehicle{" + deleteID + "}";  
  
  
 Vehicle v = restTemplate.getForObject(getURL, Vehicle.class);  
 if (v != null)  
 {  
 // then execute the delete on RestTemplate  
 restTemplate.delete(deleteURL);  
 }  
  
 }  
  
  
 @Log  
 @Timed  
 @Scheduled (cron = "0 \* \* \* \* \*") // once every minute  
 public void updateVehicle()  
 {  
 // URL that's mapped to the method we want  
 String putURL = "http://localhost:8080/updateVehicle";  
  
 // Create the new Vehicle Object to replace the old one with  
 int newID = RandomUtils.*nextInt*(1,5);  
 int year = 4444;  
 double price = (double) 44444;  
 String makeModel = "Four-Runner";  
 Vehicle v = new Vehicle(newID, makeModel, year, price);  
  
  
 // Execute on RestTemplate  
 restTemplate.put(putURL, v);  
  
 }  
  
  
 @Log  
 @Timed  
 @Scheduled (cron = "0 0 \* \* \* \*") // at minute 0 of every hour  
 public void getLatestVehicles()  
 {  
 ArrayList<Vehicle> list = new ArrayList<>();  
  
 System.*out*.println("Fetching latest vehicles:");  
  
 // URL that's mapped to the method we want  
 String getURL = "http://localhost:8080/getLatestVehicles";  
  
 // execute on rest template and save to the arraylist  
 list = restTemplate.getForObject(getURL, ArrayList.class);  
  
  
 for (Vehicle v : list)  
 {  
 // print them to the console  
 System.*out*.println(v.toString());  
 }  
 }  
  
  
}

**CheckBeforeDeleteAspect.java**

package jcooley1.demo;  
  
import java.util.Scanner;  
import org.aspectj.lang.ProceedingJoinPoint;  
import org.aspectj.lang.annotation.\*;  
import org.springframework.stereotype.Component;  
  
  
@Component  
@Aspect  
public class CheckBeforeDeleteAspect  
{  
 // should execute on the deleteVehicle method in the RESTController  
 @Pointcut("execution(\* jcooley1.demo..\*.deleteVehicle(..))")  
 public void deleteMethods() {}  
  
 // I can't figure out how to work around the error it's giving me here:  
 // Error:(19, 0) ajc: applying to join point that doesn't return void:  
 // method-execution(org.springframework.http.ResponseEntity  
 // jcooley1.demo.VehicleRESTController.deleteVehicle(int))  
 @Around ("deleteMethods() && @annotation(CheckBeforeDelete)")  
 public void checkDelete( final ProceedingJoinPoint pjp ) throws Throwable {  
  
 System.*out*.println("Are you sure you want to delete this vehicle? (Y/N)");  
 Scanner scanner = new Scanner(System.*in*);  
 String answer = scanner.nextLine();  
  
 if (answer.equalsIgnoreCase("Y"))  
 {  
 // only proceed to delete when the user says it's ok  
 System.*out*.println("Okay, proceeding to delete:");  
 pjp.proceed();  
 } else {  
 System.*out*.println("Delete cancelled.");  
 }  
 }  
  
}

**CheckBeforeDelete.java** (the Annotation)

package jcooley1.demo;  
  
import java.lang.annotation.ElementType;  
import java.lang.annotation.Retention;  
import java.lang.annotation.RetentionPolicy;  
import java.lang.annotation.Target;  
  
@Retention(RetentionPolicy.*RUNTIME*)  
@Target(ElementType.*METHOD*)  
public @interface CheckBeforeDelete  
{  
 public String name() default "";  
}

**CheckVehiclePriceAspect.java**

package jcooley1.demo;  
  
import org.aspectj.lang.ProceedingJoinPoint;  
import org.aspectj.lang.annotation.\*;  
import org.springframework.stereotype.Component;  
  
  
@Component  
@Aspect  
public class CheckVehiclePriceAspect  
{  
  
 // this pointcut courtesy of Dr. Im  
 @Pointcut("execution(\* jcooley1.demo..\*(Vehicle))")  
 public void publicMethodAcceptingVehicle() {}  
  
  
 // this advice courtesy of Dr. Im  
 @Around("publicMethodAcceptingVehicle() && @annotation(CheckVehiclePrice)")  
 public void checkVehiclePrice( final ProceedingJoinPoint pjp ) throws Throwable {  
  
 Vehicle v = (Vehicle) pjp.getArgs()[0];  
 System.*out*.println("### Checking Vehicle: " + v.toString());  
  
 if (v.getRetailPrice() > 10000)  
 {  
 // only accept (ADD/UPDATE) when vehicle price is acceptable  
 pjp.proceed();  
 } else {  
 // do nothing for unacceptable vehicles  
 System.*out*.println("Rejecting vehicle... Do nothing");  
 }  
 }  
  
}

**CheckVehiclePrice.java**  (the Annotation)

package jcooley1.demo;  
  
import java.lang.annotation.ElementType;  
import java.lang.annotation.Retention;  
import java.lang.annotation.RetentionPolicy;  
import java.lang.annotation.Target;  
  
@Retention(RetentionPolicy.*RUNTIME*)  
@Target(ElementType.*METHOD*)  
public @interface CheckVehiclePrice  
{  
 public String name() default "";  
}

**LoggingAspect.java**

package jcooley1.demo;  
  
import org.aspectj.lang.JoinPoint;  
import org.aspectj.lang.annotation.Aspect;  
import org.aspectj.lang.annotation.Before;  
import org.aspectj.lang.annotation.Pointcut;  
import org.springframework.stereotype.Component;  
  
@Component  
@Aspect  
public class LoggingAspect {  
  
 @Pointcut("execution(public \* \*(..))")  
 public void publicMethod() {}  
  
 @Pointcut("execution(\* jcooley1.demo..\*(..))")  
 public void methodsInCooleysPackage() {}  
  
 // should execute on the deleteVehicle method in the RESTController  
 // and the delete method in the DAO  
 @Pointcut("execution(\* jcooley1.demo..\*.delete\*(..))")  
 public void deleteMethods() {}  
  
  
  
 @Before("methodsInCooleysPackage() && @annotation(Log)")  
 public void addLog( final JoinPoint joinPoint ) {  
 System.*out*.println("Executing: "+joinPoint.getSignature());  
  
 Object[] arguments = joinPoint.getArgs();  
 for (Object argument : arguments) {  
 if (argument != null) {  
 System.*out*.println(argument.getClass().getSimpleName() + " = " + argument);  
 }  
 }  
 }  
  
}

**Log.java** (the Annotation)

package jcooley1.demo;  
  
import java.lang.annotation.ElementType;  
import java.lang.annotation.Retention;  
import java.lang.annotation.RetentionPolicy;  
import java.lang.annotation.Target;  
  
@Retention(RetentionPolicy.*RUNTIME*)  
@Target(ElementType.*METHOD*)  
public @interface Log {  
 public String name() default "";  
}

**TimingAspect.java**

package jcooley1.demo;  
  
import org.aspectj.lang.ProceedingJoinPoint;  
import org.aspectj.lang.annotation.Around;  
import org.aspectj.lang.annotation.Aspect;  
import org.aspectj.lang.annotation.Pointcut;  
import org.springframework.stereotype.Component;  
  
@Component  
@Aspect  
public class TimingAspect {  
  
 @Pointcut("execution(\* \*(..))")  
 public void allMethods() {}  
  
 @Pointcut("execution(\* jcooley1.demo..\*(..))")  
 public void methodsInCooleysPackage() {}  
  
  
 @Around( "methodsInCooleysPackage() && @annotation(Timed)" )  
 public Object profile( final ProceedingJoinPoint joinPoint) throws Throwable {  
 final long start = System.*currentTimeMillis*();  
 try {  
 final Object value = joinPoint.proceed();  
 return value;  
 } catch (Throwable t) {  
 throw t;  
 } finally {  
 final long stop = System.*currentTimeMillis*();  
 System.*out*.println("Execution time of "+ joinPoint.getSignature().getName() + " : "+ (stop-start));  
 }  
 }  
}

**Timed.java** (the Annotation)

package jcooley1.demo;  
  
import java.lang.annotation.ElementType;  
import java.lang.annotation.Retention;  
import java.lang.annotation.RetentionPolicy;  
import java.lang.annotation.Target;  
  
@Retention(RetentionPolicy.*RUNTIME*)  
@Target(ElementType.*METHOD*)  
public @interface Timed {  
 public String name() default "";  
}