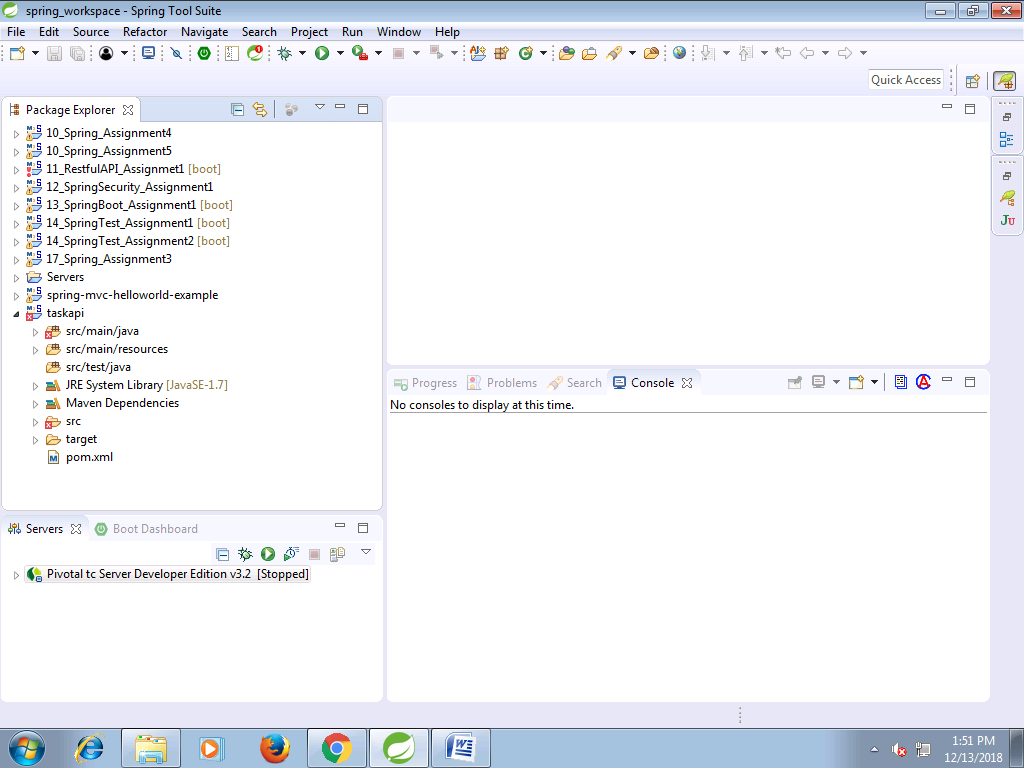
Project Structure - Task API



# CORSHeaderFilter.java

* package com.fsd.api.filter;
* import java.io.IOException;
* import javax.ws.rs.container.ContainerRequestContext;
* import javax.ws.rs.container.ContainerResponseContext;
* import javax.ws.rs.container.ContainerResponseFilter;
* import javax.ws.rs.ext.Provider;
* @Provider
* public class CORSHeaderFilter implements ContainerResponseFilter{
* @Override
* public void filter(ContainerRequestContext request, ContainerResponseContext response) throws IOException {
* System.out.println("Response Headers :"+response.getHeaders());
* System.out.println("Request Headers :"+request.getHeaders());
* response.getHeaders().add("Access-Control-Allow-Origin", "\*");
* //response.getHeaders().add("Access-Control-Allow-Credentials", "false");
* response.getHeaders().add("Access-Control-Allow-Methods", "GET, POST, DELETE, PUT, OPTIONS, HEAD");
* response.getHeaders().add("Access-Control-Allow-Headers", "Content-Type, Authorization, Accept, X-Requested-With");

* }
* }

# SpringMongoConfig.java

* package com.fsd.api.taskapi.config;
* import org.springframework.context.annotation.Bean;
* import org.springframework.data.mongodb.MongoDbFactory;
* import org.springframework.data.mongodb.core.MongoTemplate;
* import org.springframework.data.mongodb.core.SimpleMongoDbFactory;
* import com.fsd.api.taskapi.repository.ITaskRepository;
* import com.fsd.api.taskapi.repository.TaskRepositoryImpl;
* import com.mongodb.MongoClient;
* public class SpringMongoConfig {
* public @Bean
* MongoDbFactory mongoDbFactory() throws Exception {
* return new SimpleMongoDbFactory(new MongoClient(), "task\_db");
* }
* public @Bean
* MongoTemplate mongoTemplate() throws Exception {
* MongoTemplate mongoTemplate = new MongoTemplate(mongoDbFactory());
* return mongoTemplate;
* }
* public @Bean
* ITaskRepository TaskRepository() throws Exception{
* ITaskRepository taskRepo = new TaskRepositoryImpl();
* return taskRepo;
* }
* }

# StaticApplicationContext.java

* package com.fsd.api.taskapi.controller;
* import org.springframework.beans.BeansException;
* import org.springframework.context.ApplicationContext;
* import org.springframework.context.ApplicationContextAware;
* import org.springframework.context.annotation.Configuration;
* @Configuration
* public class StaticApplicationContext implements ApplicationContextAware{
* private static ApplicationContext applicationContext = null;
* public void setApplicationContext(ApplicationContext context) throws BeansException {
* applicationContext = context;
* }
* /\*\*
* \* Note that this is a static method which expose ApplicationContext
* \*\*/
* public static ApplicationContext getContext(){
* return applicationContext;
* }
* }

# TaskResouces.java

* package com.fsd.api.taskapi.controller;
* import java.util.Collection;
* import java.util.Date;
* import javax.ws.rs.Consumes;
* import javax.ws.rs.DELETE;
* import javax.ws.rs.GET;
* import javax.ws.rs.POST;
* import javax.ws.rs.PUT;
* import javax.ws.rs.Path;
* import javax.ws.rs.PathParam;
* import javax.ws.rs.Produces;
* import javax.ws.rs.core.MediaType;
* import org.slf4j.Logger;
* import org.slf4j.LoggerFactory;
* import com.fsd.api.taskapi.model.Task;
* import com.fsd.api.taskapi.service.ITaskService;
* import com.fsd.api.taskapi.service.TaskServiceImpl;
* @Path("/tasks")
* public class TaskResouces {
* private static final ITaskService taskService = (TaskServiceImpl) StaticApplicationContext.getContext().getBean("taskService");
* private static final Logger logger = LoggerFactory.getLogger(TaskResouces.class);
* @GET
* @Path("/{taskId}")
* @Produces(MediaType.APPLICATION\_JSON)
* public Task getTask(@PathParam("taskId") Long id){
* return taskService.getTask(id);
* }
* @GET
* @Produces(MediaType.APPLICATION\_JSON)
* public Collection<Task> getTasks(){
* logger.debug("Fetching all task list--->");
* if(taskService.getTasks().size() == 0)
* return null;
* else
* return taskService.getTasks();
* }

* @POST
* @Consumes(MediaType.APPLICATION\_JSON)
* @Produces(MediaType.APPLICATION\_JSON)
* public Task createTask(Task task) {
* logger.debug("Task Creation Called : "+task.getTaskName());
* task.setCreatedDate(new Date());
* Task rtnTask = taskService.createTask(task);
* return rtnTask;
* }
* @PUT
* @Path("/{taskId}")
* @Consumes(MediaType.APPLICATION\_JSON)
* @Produces(MediaType.APPLICATION\_JSON)
* public Task updateTask(@PathParam("taskId") Long taskId, Task task) {
* task.setModifiedDate(new Date());
* task.setTaskId(taskId);
* Task rtnTask = taskService.updateTask(task);
* return rtnTask;
* }
* @DELETE
* @Path("/{taskId}")
* @Consumes(MediaType.APPLICATION\_JSON)
* @Produces(MediaType.APPLICATION\_JSON)
* public Task removeTask(@PathParam("taskId") Long taskId) {
* logger.debug("Calling delete Task end task....");
* Task retTask = taskService.removeTask(taskId);
* return retTask;
* }
* }

# Database.java

* package com.fsd.api.taskapi.dao;
* import java.util.Date;
* import org.springframework.context.ApplicationContext;
* import org.springframework.context.annotation.AnnotationConfigApplicationContext;
* import org.springframework.data.mongodb.core.MongoOperations;
* import org.springframework.data.mongodb.core.query.Criteria;
* import org.springframework.data.mongodb.core.query.Query;
* import com.fsd.api.taskapi.config.SpringMongoConfig;
* import com.fsd.api.taskapi.model.ParentTask;
* import com.fsd.api.taskapi.model.SequenceGenerator;
* import com.fsd.api.taskapi.model.Task;
* public class Database {
* public static void main(String[] args) {
* ApplicationContext ctx = new AnnotationConfigApplicationContext(SpringMongoConfig.class);
* MongoOperations mongoOperation = (MongoOperations)ctx.getBean("mongoTemplate");
* SequenceGenerator obj = new SequenceGenerator();
* obj.setCounter(1l);
* obj.setId("tbl\_task.taskid");
* // mongoOperation.save(obj);
* obj.setCounter(101l);
* obj.setId("tbl\_parenttask.parentid");
* mongoOperation.save(obj);
* // Query searchUserQuery = new Query(Criteria.where("parentId").is(101l));
* /\* ParentTask obj2 = new ParentTask();
* obj2.setParentId(101l);
* obj2.setParentTaskName("Parenet Task");
* Task task =new Task();
* task.setTaskId(1l);
* task.setTaskName("First Task");
* task.setParentTask(obj2);
* task.setStartDate(new Date());
* task.setEndDate(null);
* task.setCreatedDate(new Date());
* mongoOperation.save(obj2);
* mongoOperation.save(task);
* Query searchUserQuery = new Query(Criteria.where("parentTaskMap.parentId").is(102l));
* // find the saved user again.
* Task savedUser = mongoOperation.findOne(searchUserQuery, Task.class);
* System.out.println("1. find - ParentTaskMap : " + mongoOperation.findAll(SequenceGenerator.class).size());
* System.out.println("1. find - ParentTaskMap : " + savedUser.getTaskName());\*/
* }
* }

# MongoDBSingleton.java

* package com.fsd.api.taskapi.dao;
* import com.mongodb.DB;
* import com.mongodb.MongoClient;
* public class MongoDBSingleton {
* private static MongoDBSingleton mDbSingleton;
* private static MongoClient mongoClient;
* private static DB db ;

* private static final String dbHost = "localhost";
* private static final int dbPort = 27017;
* private static final String dbName = "taskapidb";
* private MongoDBSingleton(){};
* public static MongoDBSingleton getInstance(){
* if(mDbSingleton == null){
* mDbSingleton = new MongoDBSingleton();
* }
* return mDbSingleton;
* }
* public DB getTestdb(){
* if(mongoClient == null){
* try {
* mongoClient = new MongoClient(dbHost , dbPort);
* } catch (Exception e) {
* return null;
* }
* }
* if(db == null) {
* db = mongoClient.getDB(dbName);
* }
* return db;
* }
* }

# SequenceException.java

* **package** com.fsd.api.taskapi.exception;
* **public** **class** SequenceException **extends** RuntimeException {
* **private** **static** **final** **long** ***serialVersionUID*** = 1L;
* **private** String errCode;
* **private** String errMsg;

* /\*\*
* \* **@return** the errCode
* \*/
* **public** String getErrCode() {
* **return** errCode;
* }
* /\*\*
* \* **@param** errCode the errCode to set
* \*/
* **public** **void** setErrCode(String errCode) {
* **this**.errCode = errCode;
* }
* /\*\*
* \* **@return** the errMsg
* \*/
* **public** String getErrMsg() {
* **return** errMsg;
* }
* /\*\*
* \* **@param** errMsg the errMsg to set
* \*/
* **public** **void** setErrMsg(String errMsg) {
* **this**.errMsg = errMsg;
* }
* **public** SequenceException(String errMsg) {
* **this**.errMsg = errMsg;
* }
* }

# ParentTask.java

* package com.fsd.api.taskapi.model;
* import javax.xml.bind.annotation.XmlRootElement;
* import org.springframework.data.annotation.Id;
* import org.springframework.data.mongodb.core.mapping.Document;
* import org.springframework.data.mongodb.core.mapping.Field;
* /\*\*
* \* @author 427629
* \*
* \*/
* @XmlRootElement
* @Document(collection="tbl\_parenttask")
* public class ParentTask {
* @Id
* private Long parentId;
* private Long taskId;
* private String parentTaskName;
* @Field
* private Boolean isActive = true;
* /\*\*
* \* @return the parentId
* \*/
* public Long getParentId() {
* return parentId;
* }
* /\*\*
* \* @param parentId the parentId to set
* \*/
* public void setParentId(Long parentId) {
* this.parentId = parentId;
* }
* /\*\*
* \* @return the taskId
* \*/
* public Long getTaskId() {
* return taskId;
* }
* /\*\*
* \* @param taskId the taskId to set
* \*/
* public void setTaskId(Long taskId) {
* this.taskId = taskId;
* }
* /\*\*
* \* @return the parentTaskName
* \*/
* public String getParentTaskName() {
* return parentTaskName;
* }
* /\*\*
* \* @param parentTaskName the parentTaskName to set
* \*/
* public void setParentTaskName(String parentTaskName) {
* this.parentTaskName = parentTaskName;
* }
* /\*\*
* \* @return the isActive
* \*/
* public Boolean getIsActive() {
* return isActive;
* }
* /\*\*
* \* @param isActive the isActive to set
* \*/
* public void setIsActive(Boolean isActive) {
* this.isActive = isActive;
* }

* }

# SequenceGenerator.java

* package com.fsd.api.taskapi.model;
* import org.springframework.data.annotation.Id;
* import org.springframework.data.mongodb.core.mapping.Document;
* @Document(collection="tbl\_sequence")
* public class SequenceGenerator {
* @Id
* private String id;
* private Long counter;
* /\*\*
* \* @return the id
* \*/
* public String getId() {
* return id;
* }
* /\*\*
* \* @param id the id to set
* \*/
* public void setId(String id) {
* this.id = id;
* }
* /\*\*
* \* @return the counter
* \*/
* public Long getCounter() {
* return counter;
* }
* /\*\*
* \* @param counter the counter to set
* \*/
* public void setCounter(Long counter) {
* this.counter = counter;
* }


* }

# Task.java

* package com.fsd.api.taskapi.model;
* import java.util.Date;
* import javax.json.bind.annotation.JsonbDateFormat;
* import javax.xml.bind.annotation.XmlRootElement;
* import org.springframework.data.annotation.Id;
* import org.springframework.data.mongodb.core.mapping.DBRef;
* import org.springframework.data.mongodb.core.mapping.Document;
* import org.springframework.data.mongodb.core.mapping.Field;
* /\*\*
* \* @author 427629
* \*
* \*/
* @XmlRootElement
* @Document(collection="tbl\_task")
* public class Task {
* @Id
* private Long taskId;
* @DBRef
* private ParentTask parentTask;
* private String taskName;
* @JsonbDateFormat(value="yyyy-MM-dd HH:mm:ss")
* private Date startDate;
* @JsonbDateFormat(value="yyyy-MM-dd HH:mm:ss")
* private Date endDate;
* private int priority;
* @JsonbDateFormat(value="yyyy-MM-dd HH:mm:ss")
* private Date createdDate;
* @JsonbDateFormat(value="yyyy-MM-dd HH:mm:ss")
* private Date modifiedDate;
* @Field
* private Boolean isActive = true;
* /\*\*
* \* @return the taskId
* \*/
* public Long getTaskId() {
* return taskId;
* }
* /\*\*
* \* @param taskId the taskId to set
* \*/
* public void setTaskId(Long taskId) {
* this.taskId = taskId;
* }
* /\*\*
* \* @return the parentTask
* \*/
* public ParentTask getParentTask() {
* return parentTask;
* }
* /\*\*
* \* @param parentTask the parentTask to set
* \*/
* public void setParentTask(ParentTask parentTask) {
* this.parentTask = parentTask;
* }
* /\*\*
* \* @return the taskName
* \*/
* public String getTaskName() {
* return taskName;
* }
* /\*\*
* \* @param taskName the taskName to set
* \*/
* public void setTaskName(String taskName) {
* this.taskName = taskName;
* }
* /\*\*
* \* @return the startDate
* \*/
* public Date getStartDate() {
* return startDate;
* }
* /\*\*
* \* @param startDate the startDate to set
* \*/
* public void setStartDate(Date startDate) {
* this.startDate = startDate;
* }
* /\*\*
* \* @return the endDate
* \*/
* public Date getEndDate() {
* return endDate;
* }
* /\*\*
* \* @param endDate the endDate to set
* \*/
* public void setEndDate(Date endDate) {
* this.endDate = endDate;
* }
* /\*\*
* \* @return the priority
* \*/
* public int getPriority() {
* return priority;
* }
* /\*\*
* \* @param priority the priority to set
* \*/
* public void setPriority(int priority) {
* this.priority = priority;
* }
* /\*\*
* \* @return the createdDate
* \*/
* public Date getCreatedDate() {
* return createdDate;
* }
* /\*\*
* \* @param createdDate the createdDate to set
* \*/
* public void setCreatedDate(Date createdDate) {
* this.createdDate = createdDate;
* }
* /\*\*
* \* @return the modifiedDate
* \*/
* public Date getModifiedDate() {
* return modifiedDate;
* }
* /\*\*
* \* @param modifiedDate the modifiedDate to set
* \*/
* public void setModifiedDate(Date modifiedDate) {
* this.modifiedDate = modifiedDate;
* }
* /\*\*
* \* @return the isActive
* \*/
* public Boolean getIsActive() {
* return isActive;
* }
* /\*\*
* \* @param isActive the isActive to set
* \*/
* public void setIsActive(Boolean isActive) {
* this.isActive = isActive;
* }
* /\* (non-Javadoc)
* \* @see java.lang.Object#toString()
* \*/
* @Override
* public String toString() {
* return "Task [taskId=" + taskId + ", parentTaskMap=" + parentTask + ", taskName=" + taskName + ", startDate="
* + startDate + ", endDate=" + endDate + ", priority=" + priority + ", createdDate=" + createdDate
* + ", modifiedDate=" + modifiedDate + ", isActive=" + isActive + "]";
* }
* }

# IParentTaskRepository.java

* package com.fsd.api.taskapi.repository;
* import org.springframework.data.mongodb.repository.MongoRepository;
* import com.fsd.api.taskapi.model.ParentTask;
* public interface IParentTaskRepository extends MongoRepository<ParentTask, String>{
* }

# ISequenceDao.java

* **package** com.fsd.api.taskapi.repository;
* **import** com.fsd.api.taskapi.exception.SequenceException;
* **public** **interface** ISequenceDao {
* **long** getNextSequenceId(String key) **throws** SequenceException;
* }

# ITaskRepository.java

* package com.fsd.api.taskapi.repository;
* import org.springframework.data.mongodb.repository.MongoRepository;
* import com.fsd.api.taskapi.model.Task;
* public interface ITaskRepository extends MongoRepository<Task, String>{
* Task findById(Long id);
* }

# ParentTaskRepositoryImpl.java

* package com.fsd.api.taskapi.repository;
* import java.util.Date;
* import java.util.List;
* import java.util.Optional;
* import org.springframework.beans.factory.annotation.Autowired;
* import org.springframework.data.domain.Example;
* import org.springframework.data.domain.Page;
* import org.springframework.data.domain.Pageable;
* import org.springframework.data.domain.Sort;
* import org.springframework.data.mongodb.core.MongoOperations;
* import org.springframework.data.mongodb.core.query.Criteria;
* import org.springframework.data.mongodb.core.query.Query;
* import org.springframework.data.mongodb.core.query.Update;
* import org.springframework.stereotype.Repository;
* import com.fsd.api.taskapi.model.ParentTask;
* import com.fsd.api.taskapi.model.Task;
* @Repository
* public class ParentTaskRepositoryImpl implements IParentTaskRepository{
* @Autowired
* private MongoOperations mongoOperation;
* @Override
* public <S extends ParentTask> List<S> saveAll(Iterable<S> entites) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public List<ParentTask> findAll() {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public List<ParentTask> findAll(Sort sort) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public <S extends ParentTask> S insert(S entity) {
* mongoOperation.save(entity);
* return entity;
* }
* @Override
* public <S extends ParentTask> List<S> insert(Iterable<S> entities) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public <S extends ParentTask> List<S> findAll(Example<S> example) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public <S extends ParentTask> List<S> findAll(Example<S> example, Sort sort) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public Page<ParentTask> findAll(Pageable arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public long count() {
* // TODO Auto-generated method stub
* return 0;
* }
* @Override
* public void delete(ParentTask parentTask) {
* mongoOperation.save(parentTask);
* }
* @Override
* public void deleteAll() {
* // TODO Auto-generated method stub
* }
* @Override
* public void deleteAll(Iterable<? extends ParentTask> arg0) {
* // TODO Auto-generated method stub
* }
* @Override
* public void deleteById(String arg0) {
* // TODO Auto-generated method stub
* }
* @Override
* public boolean existsById(String arg0) {
* // TODO Auto-generated method stub
* return false;
* }
* @Override
* public Iterable<ParentTask> findAllById(Iterable<String> arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public Optional<ParentTask> findById(String arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public <S extends ParentTask> S save(S parentTask) {
* System.out.println("Task details :-"+parentTask.toString());
* Query query = new Query();
* query.addCriteria(Criteria.where("parentId").is(parentTask.getParentId()));
* Update update = new Update();
* update.set("parentTaskName", parentTask.getParentTaskName());
* ParentTask retTask = mongoOperation.findAndModify(query, update, ParentTask.class);
* return (S) retTask;
* }
* @Override
* public <S extends ParentTask> long count(Example<S> arg0) {
* // TODO Auto-generated method stub
* return 0;
* }
* @Override
* public <S extends ParentTask> boolean exists(Example<S> arg0) {
* // TODO Auto-generated method stub
* return false;
* }
* @Override
* public <S extends ParentTask> Page<S> findAll(Example<S> arg0, Pageable arg1) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public <S extends ParentTask> Optional<S> findOne(Example<S> arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* }

# SequenceDaoImpl.java

* package com.fsd.api.taskapi.repository;
* import org.springframework.beans.factory.annotation.Autowired;
* import org.springframework.data.mongodb.core.FindAndModifyOptions;
* import org.springframework.data.mongodb.core.MongoOperations;
* import org.springframework.data.mongodb.core.query.Criteria;
* import org.springframework.data.mongodb.core.query.Query;
* import org.springframework.data.mongodb.core.query.Update;
* import org.springframework.stereotype.Repository;
* import com.fsd.api.taskapi.exception.SequenceException;
* import com.fsd.api.taskapi.model.SequenceGenerator;
* @Repository
* public class SequenceDaoImpl implements ISequenceDao {
* @Autowired
* private MongoOperations mongoOperation;
* @Override
* public long getNextSequenceId(String key) throws SequenceException {
* //get sequence id
* Query query = new Query(Criteria.where("\_id").is(key));
* //increase sequence id by 1
* Update update = new Update();
* update.inc("counter", 1);
* //return new increased id
* FindAndModifyOptions options = new FindAndModifyOptions();
* options.returnNew(true);
* options.upsert(true);// To auto create sequence
* //this is the magic happened.
* SequenceGenerator seqId =
* mongoOperation.findAndModify(query, update, options, SequenceGenerator.class);
* //if no id, throws SequenceException
* //optional, just a way to tell user when the sequence id is failed to generate.
* if (seqId == null) {
* throw new SequenceException("Unable to get sequence id for key : " + key);
* }
* return seqId.getCounter();
* }
* }

# TaskRepositoryImpl.java

* package com.fsd.api.taskapi.repository;
* import java.util.Date;
* import java.util.List;
* import java.util.Optional;
* import org.springframework.beans.factory.annotation.Autowired;
* import org.springframework.data.domain.Example;
* import org.springframework.data.domain.Page;
* import org.springframework.data.domain.Pageable;
* import org.springframework.data.domain.Sort;
* import org.springframework.data.mongodb.core.MongoOperations;
* import org.springframework.data.mongodb.core.query.Criteria;
* import org.springframework.data.mongodb.core.query.CriteriaDefinition;
* import org.springframework.data.mongodb.core.query.Query;
* import org.springframework.data.mongodb.core.query.Update;
* import org.springframework.stereotype.Repository;
* import com.fsd.api.taskapi.model.Task;
* @Repository
* public class TaskRepositoryImpl implements ITaskRepository{
* @Autowired
* private MongoOperations mongoOperation;
* @Override
* public List<Task> findAll() {
* return mongoOperation.findAll(Task.class);
* }
* @Override
* public List<Task> findAll(Sort arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public <S extends Task> List<S> findAll(Example<S> arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public <S extends Task> List<S> findAll(Example<S> arg0, Sort arg1) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public <S extends Task> S insert(S task) {
* S savedTask = null;
* try {
* mongoOperation.save(task);
* Query searchUserQuery = new Query(Criteria.where("taskId").is(task.getTaskId()));
* CriteriaDefinition criteriaDefinition = null;
* savedTask = (S)mongoOperation.findOne(searchUserQuery, Task.class);
* }catch(Exception e) {
* System.out.println(e.getClass() +" ---------"+e.getMessage());
* }
* return savedTask;
* }
* @Override
* public <S extends Task> List<S> insert(Iterable<S> arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public <S extends Task> List<S> saveAll(Iterable<S> arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public Page<Task> findAll(Pageable arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public long count() {
* // TODO Auto-generated method stub
* return 0;
* }
* @Override
* public void delete(Task task) {
* System.out.println("Delete Task :"+task.toString());
* mongoOperation.save(task);
* }
* @Override
* public void deleteAll() {
* // TODO Auto-generated method stub
* }
* @Override
* public void deleteAll(Iterable<? extends Task> arg0) {
* // TODO Auto-generated method stub
* }
* @Override
* public void deleteById(String arg0) {
* // TODO Auto-generated method stub
* }
* @Override
* public boolean existsById(String arg0) {
* // TODO Auto-generated method stub
* return false;
* }
* @Override
* public Iterable<Task> findAllById(Iterable<String> arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public Optional<Task> findById(String arg0) {
* // TODO Auto-generated method stub
* return null;
* }

* /\* Custom Repository method
* \* \*/
* @Override
* public Task findById(Long id) {
* Task task = null;
* try {
* Query searchUserQuery = new Query(Criteria.where("taskId").is(id));
* task = mongoOperation.findOne(searchUserQuery, Task.class);
* }catch(Exception e) {
* System.out.println("Exception in Find by Id");
* }
* return task;
* }
* @Override
* public <S extends Task> S save(S task) {
* System.out.println("Task details :-"+task.toString());
* Query query = new Query();
* query.addCriteria(Criteria.where("taskId").is(task.getTaskId()));
* Update update = new Update();
* update.set("taskName", task.getTaskName());
* update.set("priority", task.getPriority());
* update.set("modifiedDate", new Date());
* update.set("startDate", task.getStartDate());
* update.set("endDate", task.getEndDate());
* if(task.getParentTask() == null)
* update.set("parentTask", task.getParentTask());
* Task retTask = mongoOperation.findAndModify(query, update, Task.class);
* return (S) retTask;
* }
* @Override
* public <S extends Task> long count(Example<S> arg0) {
* // TODO Auto-generated method stub
* return 0;
* }
* @Override
* public <S extends Task> boolean exists(Example<S> arg0) {
* // TODO Auto-generated method stub
* return false;
* }
* @Override
* public <S extends Task> Page<S> findAll(Example<S> arg0, Pageable arg1) {
* // TODO Auto-generated method stub
* return null;
* }
* @Override
* public <S extends Task> Optional<S> findOne(Example<S> arg0) {
* // TODO Auto-generated method stub
* return null;
* }
* }

# ITaskService.java

* package com.fsd.api.taskapi.service;
* import java.util.Collection;
* import com.fsd.api.taskapi.model.Task;
* public interface ITaskService {
* Task createTask(Task task);
* Task getTask(Long id);
* Task updateTask(Task task);
* Task removeTask(Long taskId);
* Collection<Task> getTasks();
* }

# TaskServiceImpl.java

* package com.fsd.api.taskapi.service;
* import java.util.Collection;
* import java.util.Date;
* import org.springframework.beans.factory.annotation.Autowired;
* import org.springframework.stereotype.Service;
* import com.fsd.api.taskapi.model.ParentTask;
* import com.fsd.api.taskapi.model.Task;
* import com.fsd.api.taskapi.repository.IParentTaskRepository;
* import com.fsd.api.taskapi.repository.ISequenceDao;
* import com.fsd.api.taskapi.repository.ITaskRepository;
* @Service
* public class TaskServiceImpl implements ITaskService{
* private static final String TASK\_SEQ\_KEY = "tbl\_task.taskid";
* private static final String PARENTTASK\_SEQ\_KEY = "tbl\_parenttask.parentid";
* @Autowired
* private ISequenceDao sequenceDao;
* @Autowired
* private ITaskRepository taskDao;
* @Autowired
* private IParentTaskRepository parentTaskRepo;

* @Override
* public Task createTask(Task task) {
* Task retTask = null;
* task.setTaskId(sequenceDao.getNextSequenceId(TASK\_SEQ\_KEY));
* if(task.getParentTask() != null && task.getParentTask().getParentId() == null
* && task.getParentTask().getParentTaskName() != null
* && !task.getParentTask().getParentTaskName().isEmpty()) {
* ParentTask parentTask = new ParentTask();
* parentTask.setParentId(sequenceDao.getNextSequenceId(PARENTTASK\_SEQ\_KEY));
* parentTask.setParentTaskName(task.getParentTask().getParentTaskName());
* parentTaskRepo.insert(parentTask);
* task.setParentTask(parentTask);
* }
* System.out.println("Task :"+task.toString());
* retTask = taskDao.insert(task);
* return retTask;
* }
* @Override
* public Task getTask(Long taskId) {
* Task task = taskDao.findById(taskId);
* return task;
* }
* @Override
* public Task updateTask(Task task) {
* if(task.getParentTask() != null && task.getParentTask().getParentId() != null && task.getParentTask().getParentId().longValue() != 0) {
* parentTaskRepo.save(task.getParentTask());
* }else if(task.getParentTask() != null && task.getParentTask().getParentId() != null && task.getParentTask().getParentId().longValue() == 0) {
* ParentTask parentTask = new ParentTask();
* parentTask.setParentId(sequenceDao.getNextSequenceId(PARENTTASK\_SEQ\_KEY));
* parentTask.setParentTaskName(task.getParentTask().getParentTaskName());
* parentTaskRepo.insert(parentTask);
* task.setParentTask(parentTask);
* }
* Task retTask = taskDao.save(task);
* return retTask;
* }
* @Override
* public Task removeTask(Long taskId) {
* Task task = getTask(taskId);
* if(task != null && task.getParentTask() != null) {
* ParentTask pTask = task.getParentTask();
* pTask.setIsActive(false);
* parentTaskRepo.delete(pTask);
* }
* task.setEndDate(new Date());
* task.setIsActive(false);
* taskDao.delete(task);
* return task;
* }
* @Override
* public Collection<Task> getTasks() {
* return taskDao.findAll();
* }


* }

# mongoDbConfig.properties

* #Local MongoDB config
* #spring.data.mongodb.authentication-database=admin
* #spring.data.mongodb.username=root
* #spring.data.mongodb.password=root
* spring.data.mongodb.database=task\_db
* spring.data.mongodb.port=27017
* spring.data.mongodb.host=localhost

# application-context.xml

* <beans xmlns=*"http://www.springframework.org/schema/beans"*
* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*
* xmlns:context=*"http://www.springframework.org/schema/context"*
* xmlns:mongo=*"http://www.springframework.org/schema/data/mongo"*
* xsi:schemaLocation=*"http://www.springframework.org/schema/context*
* *http://www.springframework.org/schema/context/spring-context.xsd*
* *http://www.springframework.org/schema/data/mongo*
* *http://www.springframework.org/schema/data/mongo/spring-mongo-2.0.xsd*
* *http://www.springframework.org/schema/beans*
* *http://www.springframework.org/schema/beans/spring-beans.xsd"*>
* <context:component-scan base-package=*"com.fsd.api"*/>
* <context:property-placeholder location =*"classpath:config/mongoDbConfig.properties"* />
* <mongo:mongo-client host=*"${spring.data.mongodb.host}"* port=*"${spring.data.mongodb.port}"*></mongo:mongo-client>
* <mongo:db-factory dbname=*"${spring.data.mongodb.database}"*/>
* <bean id=*"mongoOperation"*
* class=*"org.springframework.data.mongodb.core.MongoTemplate"*>
* <constructor-arg name=*"mongoDbFactory"*
* ref=*"mongoDbFactory"* />
* </bean>
* <bean id=*"taskService"* class=*"com.fsd.api.taskapi.service.TaskServiceImpl"*/>
* </beans>

# web.xml

* <?xml version=*"1.0"* encoding=*"UTF-8"*?>
* <!-- This web.xml file is not required when using Servlet 3.0 container,
* see implementation details http://jersey.java.net/nonav/documentation/latest/jax-rs.html -->
* <web-app version=*"2.5"* xmlns=*"http://java.sun.com/xml/ns/javaee"*
* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*
* xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*>
* <context-param>
* <param-name>contextConfigLocation</param-name>
* <param-value>classpath:application-context.xml</param-value>
* </context-param>
* <listener>
* <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
* </listener>
* <servlet>
* <servlet-name>Jersey Web Application</servlet-name>
* <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
* <init-param>
* <param-name>jersey.config.server.provider.packages</param-name>
* <param-value>com.fsd.api</param-value>
* </init-param>
* <init-param>
* <param-name>com.sun.jersey.api.json.POJOMappingFeature</param-name>
* <param-value>true</param-value>
* </init-param>
* <init-param>
* <param-name>com.sun.jersey.config.property.packages</param-name>
* <param-value>
* com.your.packages,
* org.codehaus.jackson.jaxrs
* </param-value>
* </init-param>
* <load-on-startup>1</load-on-startup>
* </servlet>
* <servlet-mapping>
* <servlet-name>Jersey Web Application</servlet-name>
* <url-pattern>/\*</url-pattern>
* </servlet-mapping>
* </web-app>

# pom.xml

* <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/maven-v4\_0\_0.xsd"*>
* <modelVersion>4.0.0</modelVersion>
* <groupId>com.fsd.api</groupId>
* <artifactId>taskapi</artifactId>
* <packaging>war</packaging>
* <version>0.0.1-SNAPSHOT</version>
* <name>taskapi</name>
* <build>
* <finalName>taskapi</finalName>
* <plugins>
* <plugin>
* <groupId>org.apache.maven.plugins</groupId>
* <artifactId>maven-compiler-plugin</artifactId>
* <version>2.5.1</version>
* <inherited>true</inherited>
* <configuration>
* <source>1.7</source>
* <target>1.7</target>
* </configuration>
* </plugin>
* </plugins>
* </build>
* <dependencyManagement>
* <dependencies>
* <dependency>
* <groupId>org.glassfish.jersey</groupId>
* <artifactId>jersey-bom</artifactId>
* <version>${jersey.version}</version>
* <type>pom</type>
* <scope>import</scope>
* </dependency>
* </dependencies>
* </dependencyManagement>
* <dependencies>
* <dependency>
* <groupId>org.glassfish.jersey.containers</groupId>
* <artifactId>jersey-container-servlet-core</artifactId>
* <!-- use the following artifactId if you don't need servlet 2.x compatibility -->
* <!-- artifactId>jersey-container-servlet</artifactId -->
* </dependency>
* <dependency>
* <groupId>org.glassfish.jersey.inject</groupId>
* <artifactId>jersey-hk2</artifactId>
* </dependency>
* <!-- uncomment this to get JSON support -->
* <dependency>
* <groupId>org.glassfish.jersey.media</groupId>
* <artifactId>jersey-media-json-binding</artifactId>
* </dependency>
* <dependency>
* <groupId>org.glassfish.jersey.media</groupId>
* <artifactId>jersey-media-json-jackson</artifactId>
* </dependency>
* <!-- https://mvnrepository.com/artifact/org.codehaus.jackson/jackson-mapper-asl -->
* <dependency>
* <groupId>org.codehaus.jackson</groupId>
* <artifactId>jackson-mapper-asl</artifactId>
* <version>1.9.13</version>
* </dependency>
* <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->
* <!-- <dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId>
* <version>2.9.6</version> </dependency> -->
* <!-- Spring framework -->
* <dependency>
* <groupId>org.springframework</groupId>
* <artifactId>spring-core</artifactId>
* <version>5.0.7.RELEASE</version>
* </dependency>
* <dependency>
* <groupId>org.springframework</groupId>
* <artifactId>spring-context</artifactId>
* <version>5.0.7.RELEASE</version>
* </dependency>
* <dependency>
* <groupId>org.springframework</groupId>
* <artifactId>spring-webmvc</artifactId>
* <version>5.0.3.RELEASE</version>
* </dependency>
* <dependency>
* <groupId>org.springframework</groupId>
* <artifactId>spring-web</artifactId>
* <version>5.0.3.RELEASE</version>
* </dependency>
* <!-- https://mvnrepository.com/artifact/org.mongodb/mongo-java-driver -->
* <dependency>
* <groupId>org.mongodb</groupId>
* <artifactId>mongo-java-driver</artifactId>
* <version>3.8.0</version>
* </dependency>
* <dependency>
* <groupId>org.springframework.data</groupId>
* <artifactId>spring-data-mongodb</artifactId>
* <version>2.0.8.RELEASE</version>
* </dependency>
* <dependency>
* <groupId>cglib</groupId>
* <artifactId>cglib</artifactId>
* <version>2.2.2</version>
* </dependency>
* </dependencies>
* <properties>
* <jersey.version>2.27</jersey.version>
* <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
* </properties>
* </project>

**SCREENSHOTS**

