# Unit Test AI Karşılaştırması

Bu dokümanda ChatGPT ve KAI kullanılarak aşağıdaki servis için unit testler üretilmiştir.

package com.example.demo.services;  
  
import com.example.demo.entities.Brand;  
import com.example.demo.entities.Model;  
import com.example.demo.repositories.BrandRepository;  
import com.example.demo.repositories.ModelRepository;  
import com.example.demo.mappers.ModelMapper;  
import com.example.demo.dtos.requests.CreateModelRequest;  
import com.example.demo.dtos.requests.UpdateModelRequest;  
import com.example.demo.dtos.responses.GetModelDetailsResponse;  
import com.example.demo.dtos.responses.GetModelResponse;  
import com.example.demo.constants.CacheConstants;  
import com.example.demo.constants.UIMessages;  
import com.example.demo.utils.result.\*;  
import lombok.RequiredArgsConstructor;  
import org.springframework.cache.annotation.CacheEvict;  
import org.springframework.cache.annotation.Cacheable;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Isolation;  
import org.springframework.transaction.annotation.Propagation;  
import org.springframework.transaction.annotation.Transactional;  
  
import java.util.List;  
  
@Service  
@RequiredArgsConstructor  
@Transactional(isolation = Isolation.READ\_COMMITTED, propagation = Propagation.REQUIRED)  
public class ModelServiceImpl implements ModelService {

private final ModelRepository modelRepository;  
 private final BrandRepository brandRepository;  
 private final ModelMapper modelMapper;

@Cacheable(value = CacheConstants.MODELS, key = "#id")  
 @Override  
 public DataResult<GetModelDetailsResponse> getById(Long id) {  
 Model model = modelRepository.findById(id).orElse(null);  
 if (model == null) {  
 return new ErrorDataResult<>(null, UIMessages.NOT\_FOUND\_DATA);  
 }  
  
 GetModelDetailsResponse response = modelMapper.toDetailsDto(model);  
  
 return new SuccessDataResult<>(response, UIMessages.SUCCESS);  
 }

@Cacheable(value = CacheConstants.MODELS, key = CacheConstants.ALL\_KEY)  
 @Override  
 public DataResult<List<GetModelResponse>> getAll() {  
 List<GetModelResponse> response;  
 List<Model> models = modelRepository.findAll();  
  
 response = modelMapper.toDtoList(models);  
  
 return new SuccessDataResult<>(response, UIMessages.SUCCESS);  
 }

@CacheEvict(value = {CacheConstants.BRANDS, CacheConstants.MODELS}, allEntries = true)  
 @Override  
 public Result add(CreateModelRequest createModelRequest) {  
 Brand brand = brandRepository.findById(createModelRequest.getBrandId()).orElse(null);  
 if (brand == null) {  
 return new ErrorResult(UIMessages.ERROR);  
 }  
  
 Model model = modelMapper.toEntity(createModelRequest);  
 model.setBrand(brand);  
 modelRepository.save(model);  
  
 return new SuccessResult(UIMessages.SUCCESS);  
 }

@CacheEvict(value = {CacheConstants.BRANDS, CacheConstants.MODELS}, allEntries = true)  
 @Override  
 public Result update(UpdateModelRequest updateModelRequest) {  
 Model model = modelRepository.findById(updateModelRequest.getId()).orElse(null);  
 if (model == null) {  
 return new ErrorResult(UIMessages.NOT\_FOUND\_DATA);  
 }  
  
 Brand brand = brandRepository.findById(updateModelRequest.getBrandId()).orElse(null);  
 if (brand == null) {  
 return new ErrorResult(UIMessages.ERROR);  
 }  
  
 modelMapper.toEntity(updateModelRequest, model);  
 model.setBrand(brand);  
 modelRepository.save(model);  
  
 return new SuccessResult(UIMessages.SUCCESS);  
 }

@CacheEvict(value = {CacheConstants.BRANDS, CacheConstants.MODELS}, allEntries = true)  
 @Override  
 public Result delete(Long id) {  
 Model model = modelRepository.findById(id).orElse(null);  
 if (model == null) {  
 return new ErrorResult(UIMessages.NOT\_FOUND\_DATA);  
 }  
  
 modelRepository.delete(model);  
  
 return new SuccessResult(UIMessages.SUCCESS);  
 }  
}

## ChatGPT

Ücretsiz ChatGPT 4o mini modeli kullanılmıştır. Unit testlerin hepsi başarılı çalışmıştır. Coverage:100

package com.example.demo.services;  
  
import com.example.demo.constants.UIMessages;  
import com.example.demo.dtos.requests.CreateModelRequest;  
import com.example.demo.dtos.requests.UpdateModelRequest;  
import com.example.demo.dtos.responses.GetModelDetailsResponse;  
import com.example.demo.dtos.responses.GetModelResponse;  
import com.example.demo.entities.Brand;  
import com.example.demo.entities.Model;  
import com.example.demo.mappers.ModelMapper;  
import com.example.demo.repositories.BrandRepository;  
import com.example.demo.repositories.ModelRepository;  
import com.example.demo.utils.result.\*;  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
  
import java.util.List;  
import java.util.Optional;  
  
import static org.junit.jupiter.api.Assertions.\*;  
import static org.mockito.Mockito.\*;  
  
class ModelServiceImplTest {  
  
 private ModelRepository modelRepository;  
 private BrandRepository brandRepository;  
 private ModelMapper modelMapper;  
 private ModelServiceImpl modelService;  
  
 @BeforeEach  
 void setUp() {  
 modelRepository = *mock*(ModelRepository.class);  
 brandRepository = *mock*(BrandRepository.class);  
 modelMapper = *mock*(ModelMapper.class);  
 modelService = new ModelServiceImpl(modelRepository, brandRepository, modelMapper);  
 }  
  
 @Test  
 void getById\_shouldReturnModelDetails\_whenModelExists() {  
 Model model = new Model();  
 model.setId(1L);  
 GetModelDetailsResponse response = new GetModelDetailsResponse();  
  
 *when*(modelRepository.findById(1L)).thenReturn(Optional.*of*(model));  
 *when*(modelMapper.toDetailsDto(model)).thenReturn(response);  
  
 DataResult<GetModelDetailsResponse> result = modelService.getById(1L);  
  
 *assertTrue*(result.isSuccess());  
 *assertEquals*(response, result.getData());  
 *assertEquals*(UIMessages.SUCCESS, result.getMessage());  
 }  
  
 @Test  
 void getById\_shouldReturnError\_whenModelDoesNotExist() {  
 *when*(modelRepository.findById(1L)).thenReturn(Optional.*empty*());  
  
 DataResult<GetModelDetailsResponse> result = modelService.getById(1L);  
  
 *assertFalse*(result.isSuccess());  
 *assertNull*(result.getData());  
 *assertEquals*(UIMessages.NOT\_FOUND\_DATA, result.getMessage());  
 }

@Test  
 void getAll\_shouldReturnAllModels() {  
 List<Model> models = List.*of*(new Model());  
 List<GetModelResponse> dtos = List.*of*(new GetModelResponse());  
  
 *when*(modelRepository.findAll()).thenReturn(models);  
 *when*(modelMapper.toDtoList(models)).thenReturn(dtos);  
  
 DataResult<List<GetModelResponse>> result = modelService.getAll();  
  
 *assertTrue*(result.isSuccess());  
 *assertEquals*(dtos, result.getData());  
 *assertEquals*(UIMessages.SUCCESS, result.getMessage());  
 }  
  
 @Test  
 void add\_shouldAddModel\_whenBrandExists() {  
 CreateModelRequest request = new CreateModelRequest();  
 request.setBrandId(10L);  
 Brand brand = new Brand();  
 Model model = new Model();  
  
 *when*(brandRepository.findById(10L)).thenReturn(Optional.*of*(brand));  
 *when*(modelMapper.toEntity(request)).thenReturn(model);  
  
 Result result = modelService.add(request);  
  
 *verify*(modelRepository).save(model);  
 *assertSame*(brand, model.getBrand());  
 *assertTrue*(result.isSuccess());  
 *assertEquals*(UIMessages.SUCCESS, result.getMessage());  
 }  
  
 @Test  
 void add\_shouldReturnError\_whenBrandNotFound() {  
 CreateModelRequest request = new CreateModelRequest();  
 request.setBrandId(10L);  
  
 *when*(brandRepository.findById(10L)).thenReturn(Optional.*empty*());  
  
 Result result = modelService.add(request);  
  
 *verify*(modelRepository, *never*()).save(*any*());  
 *assertFalse*(result.isSuccess());  
 *assertEquals*(UIMessages.ERROR, result.getMessage());  
 }  
  
 @Test  
 void update\_shouldUpdateModel\_whenModelAndBrandExist() {  
 UpdateModelRequest request = new UpdateModelRequest();  
 request.setId(1L);  
 request.setBrandId(2L);  
  
 Model model = new Model();  
 Brand brand = new Brand();  
  
 *when*(modelRepository.findById(1L)).thenReturn(Optional.*of*(model));  
 *when*(brandRepository.findById(2L)).thenReturn(Optional.*of*(brand));  
  
 Result result = modelService.update(request);  
  
 *verify*(modelMapper).toEntity(request, model);  
 *verify*(modelRepository).save(model);  
 *assertSame*(brand, model.getBrand());  
 *assertTrue*(result.isSuccess());  
 *assertEquals*(UIMessages.SUCCESS, result.getMessage());  
 }

@Test  
 void update\_shouldReturnError\_whenModelNotFound() {  
 UpdateModelRequest request = new UpdateModelRequest();  
 request.setId(1L);  
  
 *when*(modelRepository.findById(1L)).thenReturn(Optional.*empty*());  
  
 Result result = modelService.update(request);  
  
 *verify*(modelRepository, *never*()).save(*any*());  
 *assertFalse*(result.isSuccess());  
 *assertEquals*(UIMessages.NOT\_FOUND\_DATA, result.getMessage());  
 }  
  
 @Test  
 void update\_shouldReturnError\_whenBrandNotFound() {  
 UpdateModelRequest request = new UpdateModelRequest();  
 request.setId(1L);  
 request.setBrandId(2L);  
 Model model = new Model();  
  
 *when*(modelRepository.findById(1L)).thenReturn(Optional.*of*(model));  
 *when*(brandRepository.findById(2L)).thenReturn(Optional.*empty*());  
  
 Result result = modelService.update(request);  
  
 *verify*(modelRepository, *never*()).save(*any*());  
 *assertFalse*(result.isSuccess());  
 *assertEquals*(UIMessages.ERROR, result.getMessage());  
 }  
  
 @Test  
 void delete\_shouldRemoveModel\_whenModelExists() {  
 Model model = new Model();  
 model.setId(1L);  
  
 *when*(modelRepository.findById(1L)).thenReturn(Optional.*of*(model));  
  
 Result result = modelService.delete(1L);  
  
 *verify*(modelRepository).delete(model);  
 *assertTrue*(result.isSuccess());  
 *assertEquals*(UIMessages.SUCCESS, result.getMessage());  
 }  
  
 @Test  
 void delete\_shouldReturnError\_whenModelNotFound() {  
 *when*(modelRepository.findById(1L)).thenReturn(Optional.*empty*());  
  
 Result result = modelService.delete(1L);  
  
 *verify*(modelRepository, *never*()).delete(*any*());  
 *assertFalse*(result.isSuccess());  
 *assertEquals*(UIMessages.NOT\_FOUND\_DATA, result.getMessage());  
 }  
}

## KAI