Contents

[Create model 1](#_Toc461099458)

[Add validation messages 3](#_Toc461099459)

[Create a repository 3](#_Toc461099460)

[Create service 3](#_Toc461099461)

[Implement this service 4](#_Toc461099462)

[Expose the service 5](#_Toc461099463)

[Create a controller 5](#_Toc461099464)

To implement a new CRUD, follow the following steps:

# Create model in com.bitmakersbd.biyebari.server.model

**package** com.bitmakersbd.biyebari.server.model;  
  
**import** com.bitmakersbd.biyebari.server.util.State;  
**import** com.bitmakersbd.biyebari.server.validation.ValidateOnCreate;  
**import** com.bitmakersbd.biyebari.server.validation.ValidateOnUpdate;  
**import** com.fasterxml.jackson.annotation.JsonIgnore;  
**import** org.hibernate.validator.constraints.NotEmpty;  
  
**import** javax.persistence.\*;  
**import** javax.validation.constraints.NotNull;  
**import** java.util.ArrayList;  
**import** java.util.Date;  
**import** java.util.List;  
  
*/\*\*  
 \* This model holds the groups. Groups are dependent on {****@link*** *com.bitmakersbd.biyebari.server.model.District}.  
 \* A district can have many groups. Groups are used in {****@link*** *com.bitmakersbd.biyebari.server.model.Vendor} to  
 \* indicate the service groups of a vendor. A vendor can have multiple service groups and a group can have  
 \* multiple vendors.  
 \*  
 \** ***@see*** *com.bitmakersbd.biyebari.server.model.District  
 \** ***@see*** *com.bitmakersbd.biyebari.server.model.Vendor  
 \*/*@Entity  
@Table(name = **"tbl\_group"**)  
**public class** Group {  
 @Id  
 @GeneratedValue(strategy = GenerationType.***AUTO***)  
 **private** Long **id**;

// other fields…

@Temporal(TemporalType.***TIMESTAMP***)  
 @Column(name = **"created\_at"**, columnDefinition = **"timestamp"**, nullable = **true**, updatable = **false**)  
 **private** Date **createdAt** = **new** Date();  
  
 @NotNull(groups = ValidateOnCreate.**class**)  
 @Column(name = **"created\_by"**, columnDefinition = **"varchar"**, length = 50)  
 **private** Long **createdBy**;  
  
 @Temporal(TemporalType.***TIMESTAMP***)  
 @Column(name = **"updated\_at"**, columnDefinition = **"timestamp"**, nullable = **true**, insertable = **false**)  
 **private** Date **updatedAt**;  
  
 @NotNull(groups = ValidateOnUpdate.**class**)  
 @Column(name = **"updated\_by"**, columnDefinition = **"varchar"**, length = 50)  
 **private** Long **updatedBy**;  
  
 **public** Group() {  
 }  
  
 **public** Group(Long id) {  
 **this**.**id** = id;  
 }  
  
 **public** Long getId() {  
 **return id**;  
 }  
  
 **public void** setId(Long id) {  
 **this**.**id** = id;  
 }  
  
 **public** Date getCreatedAt() {  
 **return createdAt**;  
 }  
  
 **public** Long getCreatedBy() {  
 **return createdBy**;  
 }  
  
 **public void** setCreatedBy(Long createdBy) {  
 **this**.**createdBy** = createdBy;  
 }  
  
 **public** Date getUpdatedAt() {  
 **return updatedAt**;  
 }  
  
 **public** Long getUpdatedBy() {  
 **return updatedBy**;  
 }  
  
 **public void** setUpdatedBy(Long updatedBy) {  
 **this**.**updatedBy** = updatedBy;  
 }  
  
 @PreUpdate  
 **public void** onUpdate() {  
 **this**.**updatedAt** = **new** Date();  
 }  
}

# Add validation messages in the src\main\resources\validationMessages.properties file. Must follow this format: *[ConstraintName].[ClassName].[FieldName]=[Message]*

**NotEmpty.group.name**=**Name is required  
NotNull.group.cOrder**=**COrder is required  
NotNull.group.defaultImage**=**Default image is required  
NotNull.group.createdBy**=**Created by is required  
NotNull.group.updatedBy**=**Updated by is required**

# Create a repository interface in com.bitmakersbd.biyebari.server.repository

**package** com.bitmakersbd.biyebari.server.repository;  
  
**import** com.bitmakersbd.biyebari.server.model.Group;  
**import** com.bitmakersbd.biyebari.server.util.State;  
**import** org.springframework.data.domain.Page;  
**import** org.springframework.data.domain.Pageable;  
**import** org.springframework.data.jpa.repository.JpaRepository;  
  
**public interface** GroupRepository **extends** JpaRepository<Group, Long> {  
 **public** Page<Group> findAllByIsActive(State isActive, Pageable pageable);  
}

# Create a service interface in com.bitmakersbd.biyebari.server.service

**package** com.bitmakersbd.biyebari.server.service;  
  
**import** com.bitmakersbd.biyebari.server.model.Group;  
**import** org.springframework.data.domain.Page;  
**import** org.springframework.data.domain.Pageable;  
  
**public interface** GroupService {  
 Page getAll(Pageable pageable) **throws** Exception;  
  
 Group get(Long id) **throws** Exception;  
  
 Group create(Group group) **throws** Exception;  
  
 **void** delete(Long id) **throws** Exception;  
  
 Group update(Group group) **throws** Exception;  
}

# Implement this service in the same com.bitmakersbd.biyebari.server.service package. Add any custom message in src\main\resources\messages.properties file.

**package** com.bitmakersbd.biyebari.server.service;  
  
**import** com.bitmakersbd.biyebari.server.model.Group;  
**import** com.bitmakersbd.biyebari.server.repository.GroupRepository;  
**import** com.bitmakersbd.biyebari.server.util.Messages;  
**import** com.bitmakersbd.biyebari.server.util.State;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.data.domain.Page;  
**import** org.springframework.data.domain.Pageable;  
**import** org.springframework.transaction.annotation.Transactional;  
  
**public class** GroupServiceImpl **implements** GroupService {  
 @Autowired  
 GroupRepository **groupRepository**;  
  
 @Autowired  
 Messages **messages**;  
  
 @Transactional(readOnly = **true**)  
 @Override  
 **public** Page getAll(Pageable pageable) **throws** Exception {  
 **return groupRepository**.findAllByIsActive(State.***active***, pageable);  
 }  
  
 @Transactional(readOnly = **true**)  
 @Override  
 **public** Group get(Long id) **throws** Exception {  
 Group group = **groupRepository**.findOne(id);  
  
 **if** (group.getIsActive() != State.***active***) {  
 **throw new** Exception(**messages**.getMessage(**"group.not.active"**));  
 }  
  
 **return** group;  
 }  
  
 @Transactional  
 @Override  
 **public** Group create(Group group) **throws** Exception {  
 **return groupRepository**.save(group);  
 }  
  
 @Transactional  
 @Override  
 **public void** delete(Long id) **throws** Exception {  
 Group groupInDb = **groupRepository**.findOne(id);  
 **if** (groupInDb == **null**) **throw new** Exception(**messages**.getMessage(**"group.not.found"**));  
  
 **groupRepository**.delete(id);  
 }  
  
 @Transactional  
 @Override  
 **public** Group update(Group group) **throws** Exception {  
 Group groupInDb = **groupRepository**.findOne(group.getId());  
 **if** (groupInDb == **null**) **throw new** Exception(**messages**.getMessage(**"group.not.found"**));  
  
 *// update only fields which are allowed to update* groupInDb.setName(group.getName());  
 groupInDb.setDescription(group.getDescription());  
 groupInDb.setIsActive(group.getIsActive());  
 groupInDb.setParent(group.getParent());  
 groupInDb.setcOrder(group.getcOrder());  
 groupInDb.setDefaultImage(group.getDefaultImage());  
 groupInDb.setInfoFormat(group.getInfoFormat());  
 groupInDb.setIsBudget(group.getIsBudget());  
 groupInDb.setUpdatedBy(group.getUpdatedBy());  
  
 **return groupRepository**.save(groupInDb);  
 }  
}

# Expose the service as a Spring bean in src\main\webapp\WEB-INF\dispatcher-servlet.xml in the services section

<**bean id="groupService" class="com.bitmakersbd.biyebari.server.service.GroupServiceImpl"**/>

# Create a controller in com.bitmakersbd.biyebari.server.controller

**package** com.bitmakersbd.biyebari.server.controller;  
  
**import** com.bitmakersbd.biyebari.server.model.Group;  
**import** com.bitmakersbd.biyebari.server.service.GroupService;  
**import** com.bitmakersbd.biyebari.server.util.Messages;  
**import** com.bitmakersbd.biyebari.server.util.RestResponse;  
**import** com.bitmakersbd.biyebari.server.validation.ValidateOnCreate;  
**import** com.bitmakersbd.biyebari.server.validation.ValidateOnUpdate;  
**import** org.apache.log4j.Logger;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.data.domain.Pageable;  
**import** org.springframework.validation.annotation.Validated;  
**import** org.springframework.web.bind.annotation.\*;  
  
*/\*\*  
 \* REST API controller for {****@link*** *com.bitmakersbd.biyebari.server.model.Group}. This manages groups.  
 \*  
 \** ***@see*** *com.bitmakersbd.biyebari.server.model.Group  
 \** ***@see*** *com.bitmakersbd.biyebari.server.service.GroupService  
 \** ***@see*** *com.bitmakersbd.biyebari.server.service.GroupServiceImpl  
 \** ***@see*** *com.bitmakersbd.biyebari.server.repository.GroupRepository  
 \*/*@RestController  
@RequestMapping(value = **"/groups"**)  
**public class** GroupController {  
 **private static final** Logger ***logger*** = Logger.*getLogger*(GroupController.**class**);  
  
 @Autowired  
 GroupService **groupService**;  
  
 @Autowired  
 Messages **messages**;  
  
 */\*\*  
 \* Gets all active groups.  
 \*  
 \** ***@param pageable*** *the page parameters  
 \** ***@return*** *all active groups as {****@link*** *com.bitmakersbd.biyebari.server.util.RestResponse}  
 \*/* @RequestMapping(method = RequestMethod.***GET***)  
 **public** RestResponse getAll(Pageable pageable) {  
 RestResponse restResponse = **new** RestResponse();  
 **try** {  
 restResponse.setData(**groupService**.getAll(pageable));  
 } **catch** (Exception e) {  
 restResponse.setError(**true**);  
 restResponse.addMessage(e.getMessage());  
 }  
 **return** restResponse;  
 }  
  
 */\*\*  
 \* Get a single group with id.  
 \*  
 \** ***@param id*** *the id of the group  
 \** ***@return*** *the group as {****@link*** *com.bitmakersbd.biyebari.server.util.RestResponse}  
 \*/* @RequestMapping(value = **"/{id}"**, method = RequestMethod.***GET***)  
 **public** RestResponse get(@PathVariable(value = **"id"**) **long** id) {  
 RestResponse restResponse = **new** RestResponse();  
 **try** {  
 restResponse.setData(**groupService**.get(id));  
 } **catch** (Exception e) {  
 restResponse.setError(**true**);  
 restResponse.addMessage(e.getMessage());  
 }  
 **return** restResponse;  
 }  
  
 */\*\*  
 \* Create a group.  
 \*  
 \** ***@param group*** *the group object  
 \** ***@return*** *the created group as {****@link*** *com.bitmakersbd.biyebari.server.util.RestResponse}  
 \*/* @RequestMapping(method = RequestMethod.***POST***)  
 **public** RestResponse create(@Validated(ValidateOnCreate.**class**) @RequestBody Group group) {  
 RestResponse restResponse = **new** RestResponse();  
 **try** {  
 restResponse.setData(**groupService**.create(group));  
 restResponse.addMessage(**messages**.getMessage(**"created.successfully"**));  
 } **catch** (Exception e) {  
 restResponse.setError(**true**);  
 restResponse.addMessage(e.getMessage());  
 }  
 **return** restResponse;  
 }  
  
 */\*\*  
 \* Update a group.  
 \*  
 \** ***@param group*** *the group to update  
 \** ***@param id*** *the id of the group  
 \** ***@return*** *the updated group as {****@link*** *com.bitmakersbd.biyebari.server.util.RestResponse}  
 \*/* @RequestMapping(value = **"/{id}"**, method = RequestMethod.***PUT***)  
 **public** RestResponse update(@Validated(ValidateOnUpdate.**class**) @RequestBody Group group, @PathVariable(value = **"id"**) Long id) {  
 RestResponse restResponse = **new** RestResponse();  
 **try** {  
 group.setId(id);  
 restResponse.setData(**groupService**.update(group));  
 restResponse.addMessage(**messages**.getMessage(**"updated.successfully"**));  
 } **catch** (Exception e) {  
 restResponse.setError(**true**);  
 restResponse.addMessage(e.getMessage());  
 }  
 **return** restResponse;  
 }  
  
 */\*\*  
 \* Delete a group.  
 \*  
 \** ***@param id*** *the id of the group  
 \** ***@return*** *the deleted group as {****@link*** *com.bitmakersbd.biyebari.server.util.RestResponse}  
 \*/* @RequestMapping(value = **"/{id}"**, method = RequestMethod.***DELETE***)  
 **public** RestResponse delete(@PathVariable(value = **"id"**) **long** id) {  
 RestResponse restResponse = **new** RestResponse();  
 **try** {  
 **groupService**.delete(id);  
 restResponse.setData(**true**);  
 restResponse.addMessage(**messages**.getMessage(**"deleted.successfully"**));  
 } **catch** (Exception e) {  
 restResponse.setError(**true**);  
 restResponse.addMessage(e.getMessage());  
 }  
 **return** restResponse;  
 }  
}