此文章已于 11:22:24 2018/3/7 发布到 HealerJean梦想博客

2、项目规范

类别 DDKJ ; 经验

# 1、前台交互，传递的对象

## 1、前段响应类

### 1、一般正常responseBean.setCode(ErrorCodeEnum.正常.code);//200

### 2、Result对象正常

### 3、一般错误

### 4、错误信息了，录入，code系统错误

### 5、错误、枚举code和desc录入

### 6、错误、code，message自己填充

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by Fernflower decompiler)  
//  
  
package com.duodian.admore.core.bean;  
  
import com.duodian.admore.enums.ErrorCodeEnum;  
  
public class ResponseBean {  
 private boolean success;  
 private Object result = "{}";  
 private String message = "";  
 private String code = "";  
 private String date;  
  
 private ResponseBean() {  
 }  
  
 public static ResponseBean buildSuccess() {  
 ResponseBean responseBean = new ResponseBean();  
 responseBean.setSuccess(true);  
 responseBean.setCode(ErrorCodeEnum.正常.code);//200  
 responseBean.setDate(System.currentTimeMillis() + "");//系统事件戳  
 return responseBean;  
 }  
  
 public static ResponseBean buildSuccess(Object result) {  
 ResponseBean responseBean = new ResponseBean();  
 responseBean.setSuccess(true);  
 responseBean.setCode(ErrorCodeEnum.正常.code);  
 responseBean.setResult(result);  
 responseBean.setDate(System.currentTimeMillis() + "");  
 return responseBean;  
 }  
  
 public static ResponseBean buildFailure() {  
 ResponseBean responseBean = new ResponseBean();  
 responseBean.setSuccess(false);  
 responseBean.setCode(ErrorCodeEnum.系统错误.code);  
 responseBean.setDate(System.currentTimeMillis() + "");  
 return responseBean;  
 }  
  
 public static ResponseBean buildFailure(String message) {  
 ResponseBean responseBean = new ResponseBean();  
 responseBean.setSuccess(false);  
 responseBean.setCode(ErrorCodeEnum.系统错误.code);  
 responseBean.setMessage(message);  
 responseBean.setDate(System.currentTimeMillis() + "");  
 return responseBean;  
 }  
  
 public static ResponseBean buildFailure(ErrorCodeEnum errorCodeEnum) {  
 ResponseBean responseBean = new ResponseBean();  
 responseBean.setSuccess(false);  
 responseBean.setCode(errorCodeEnum.code);  
 responseBean.setMessage(errorCodeEnum.desc);  
 responseBean.setDate(System.currentTimeMillis() + "");  
 return responseBean;  
 }  
  
 public static ResponseBean buildFailure(String code, String message) {  
 ResponseBean responseBean = new ResponseBean();  
 responseBean.setSuccess(false);  
 responseBean.setCode(code);  
 responseBean.setMessage(message);  
 responseBean.setDate(System.currentTimeMillis() + "");  
 return responseBean;  
 }  
  
 get set……  
}

## 2、枚举信息类

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by Fernflower decompiler)  
//  
  
package com.duodian.admore.enums;  
  
public enum ErrorCodeEnum {  
 正常("200", "访问正常"),  
 条件验证失败("201", "条件验证失败"),  
 参数解析失败("202", "参数解析失败"),  
 操作确认("203", "操作确认"),  
 逻辑错误("301", "逻辑错误"),  
 登录标识过期("400", "登录标识过期"),  
 强制升级("401", "强制升级"),  
 访问禁止("403", "访问禁止"),  
 页面丢失("404", "页面丢失"),  
 系统错误("500", "系统内部错误"),  
 未登录("204", "未登录"),  
 不是企业认证("205", "不是企业认证"),  
 UNKNOWN("999", "未知错误");  
  
 public String code;  
 public String desc;  
  
 private ErrorCodeEnum(String code, String desc) {  
 this.code = code;  
 this.desc = desc;  
 }  
  
 public static String getDesc(String code) {  
 if (code != null && !"".equals(code)) {  
 ErrorCodeEnum[] var1 = values();  
 int var2 = var1.length;  
  
 for(int var3 = 0; var3 < var2; ++var3) {  
 ErrorCodeEnum desc = var1[var3];  
 if (desc.code.equals(code)) {  
 return desc.desc;  
 }  
 }  
  
 return UNKNOWN.desc;  
 } else {  
 return UNKNOWN.desc;  
 }  
 }  
}

### 1、 保存操作，

#### 1、一般正常

#### 2、错误信息，录入

@RequestMapping("save")  
@ResponseBody  
public ResponseBean save(Customer customer){  
 try {  
 customerService.saveCustomer(customer);  
 return ResponseBean.buildSuccess(); //一般正常  
 } catch (AppException e) {  
 return ResponseBean.buildFailure(e.getMessage()); //将报错信息输入  
 }  
}

### 2、查询操作findCustomerById(Long id)

#### 1、result的正常返回

#### 2、错误信息，录入

@RequestMapping("findCustomerById")  
@ResponseBody  
public ResponseBean findCustomerById(Long id){  
 try {  
 Customer customer = customerService.findCustomerById(id);  
 return ResponseBean.buildSuccess(customer);  
 } catch (AppException e) {  
 return ResponseBean.buildFailure(e.getMessage());  
 }  
}

## 3、数据字典 （用到了上面的Object 对象）

### 1、result的正常返回

@RequestMapping("getStatusList")  
@ResponseBody  
public ResponseBean getStatusList(){  
 try {  
 return ResponseBean.buildSuccess(EnumCrmUserStatus.getTypeList());  
 } catch (AppException e) {  
 return ResponseBean.buildFailure(e.getMessage());  
 }  
}

### 2、枚举如下

package com.duodian.admore.enums.customer;  
  
  
import java.util.Arrays;  
import java.util.List;  
  
/\*\*  
 \* 类名称：EnumCrmUserType  
 \* 类描述：crm客户来源  
 \* 创建人：shiwei  
 \* 修改人：  
 \* 修改时间：2015/12/19 17:47  
 \* 修改备注：  
 \*  
 \* @version 1.0.0  
 \*/  
public enum EnumCrmUserStatus {  
  
 ***潜在客户***(1,"潜在客户"),  
 ***有意向***(2,"有意向"),  
 ***失败***(3,"失败"),  
 ***已成交***(4,"已成交"),  
 ***老客户***(5,"老客户"),  
  
 ***无***(88,"无"),  
  
 ***未知***(99,"未知")  
 ;  
  
 public Integer type;  
 public String des;  
  
 EnumCrmUserStatus(Integer type, String des) {  
 this.type = type;  
 this.des = des;  
 }  
  
 public static List<EnumCrmUserStatus> getTypeList() {  
 return Arrays.asList(values());  
 }  
  
  
 public Integer getType() {  
 return type;  
 }  
  
 public String getDes() {  
 return des;  
 }  
  
 public static boolean checkExist(Integer type){  
 if(type == null) return false;  
 for (EnumCrmUserStatus value : EnumCrmUserStatus.values()){  
 if (value.type.compareTo(type) == 0){  
 return true;  
 }  
 }  
 return false;  
 }  
  
  
  
 public static EnumCrmUserStatus getEnumCashFlowType(Integer type){  
 if(type == null) return EnumCrmUserStatus.***未知***;  
 for (EnumCrmUserStatus value : EnumCrmUserStatus.values()){  
 if (value.type.compareTo(type) == 0){  
 return value;  
 }  
 }  
 return EnumCrmUserStatus.***未知***;  
 }  
  
 public static String getDes(Integer type){  
 if(type == null) return EnumCrmUserStatus.***未知***.des;  
 for (EnumCrmUserStatus value : EnumCrmUserStatus.values()){  
 if (value.type.compareTo(type) == 0){  
 return value.des;  
 }  
 }  
 return EnumCrmUserStatus.***未知***.des;  
 }  
}

## 4、根据id删除

### 1、一般正常

### 2、信息错误录入

@RequestMapping("del")  
@ResponseBody  
public ResponseBean delCustomer(Long id){  
 try {  
 customerService.delCustomer(id);  
 return ResponseBean.buildSuccess();  
 } catch (AppException e) {  
 return ResponseBean.buildFailure(e.getMessage());  
 }  
}

# 2、删除的正确使用方式

### 1、正常情况下不应该从数据库表中直接删除，而是应该添加字段，让它变成不可见。（这种不可见需要再查询的时候特别 注意）

#### 1、删除代码，

@Override  
public void delContact(Long ContactId) {  
 Contact Contact = ContactDAO.findOne(ContactId);  
 Contact.setIsVisible(Contact.***DISABLE***);  
 ContactDAO.save(Contact);  
}

#### 2、对应实体类字段

@Entity  
@Table(name = "crm\_contact\_record")  
public class Contact implements Serializable {  
  
 public static final int ***ENABLE*** = 1; //可见  
 public static final int ***DISABLE*** = 0; //不可见  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.***IDENTITY***)  
 private Long id;  
 private Long adminId;  
 private Long customerId;  
 private String title;  
 private String contactName;  
 private Integer type;  
 private String content;  
 private Integer isVisible;  
 private Date contactTime;  
 private Date cdate;

# 3、自定义异常的使用

## 1、业务异常

/\*\*  
 \* Created by j.sh on 2015/5/9.  
 \* 业务异常  
 \*/  
public class AppException extends RuntimeException {  
  
 private String code;  
  
 public AppException(String message) {  
 super(message);  
 this.code = ErrorCodeEnum.***逻辑错误***.code;  
 }  
  
 public AppException(String code,String message) {  
 super(message);  
 this.code = code;  
 }  
  
 public String getCode() {  
 return code;  
 }  
  
 public void setCode(String code) {  
 this.code = code;  
 }  
}

## 2、异常错误的枚举值

/\*\*  
 \* 类名称：ErrorCodeEnum  
 \* 类描述：错误状态枚举值  
 \* 创建人：qingxu  
 \* 修改人：  
 \* 修改时间：2015/11/5 11:27  
 \* 修改备注：  
 \*  
 \* @version 1.0.0  
 \*/  
public enum ErrorCodeEnum {  
  
 ***正常***("200","访问正常"),  
 ***条件验证失败***("201","条件验证失败"),  
 ***参数解析失败***("202","参数解析失败"),  
 ***操作确认***("203","操作确认"),  
 ***逻辑错误***("301","逻辑错误"),  
 ***登录标识过期***("400","登录标识过期"),  
 ***强制升级***("401","强制升级"),  
 ***访问禁止***("403","访问禁止"),  
 ***页面丢失***("404","页面丢失"),  
 ***系统错误***("500","系统内部错误"),  
 ***未登录***("204","未登录"),  
 ***不是企业认证***("205","不是企业认证"),  
 ***UNKNOWN***("999","未知错误")  
 ;  
  
 public String code;  
  
 public String desc;  
  
 ErrorCodeEnum(String code, String desc) {  
 this.code = code;  
 this.desc = desc;  
 }  
  
 public static String getDesc(String code){  
 if(code == null || "".equals(code)){  
 return ErrorCodeEnum.***UNKNOWN***.desc;  
 }  
 for(ErrorCodeEnum desc: ErrorCodeEnum.values()){  
 if(desc.code.equals(code)){  
 return desc.desc;  
 }  
 }  
 return ErrorCodeEnum.***UNKNOWN***.desc;  
 }  
}

## 3、正确使用

@RequestMapping("config/loadDockedQuery")  
@ResponseBody  
public ResponseBean loadDockedQuery(Integer dockedType,String validate){  
 try {  
 AppsDockedConfigData data = appsDockedBufferService.loadAppsDockedConfigData(validate,dockedType);  
 return ResponseBean.buildSuccess(data);  
 } catch (AppException e) {  
 return ResponseBean.buildFailure(e.getCode(),e.getMessage());  
 } catch (Exception e) {  
 logger.error(e.getMessage(),e);  
 return ResponseBean.buildFailure(ErrorCodeEnum.***系统错误***); // 不想暴露给前台  
 }  
}

# 4、js前端调用

$http.post(***ctx*** + '/skin/enable', {  
 appId: item.trackId,  
 channelListsEnableValue: $scope.channelListsEnableValue  
}).success(function (data, status) {  
 $scope.processing = false;  
 if (data.success) {  
 toastr.success('操作成功', '提示');  
 $scope.skinData();  
 } else {  
 ngDialog.alert(data.message);  
 }  
 })  
 .error(function (data, status) {  
 $scope.processing = false;  
 ngDialog.alert('操作失败');  
 });

# 5、小程序找不同学来的 ，也是和前端交互的

## 1、code枚举

package com.duodian.admore.zhaobutong.bean;  
  
/\*\*  
 \* @author fengchuanbo  
 \*/  
public enum Code {  
  
 ***OK***("0", "OK"),  
 ***ERROR***("500", "系统错误"),  
 ***illegalArgument***("40001", "参数错误"),  
 ***SESSION\_CHECK\_ERROR***("40002","认证失败"),  
 ***TOKEN\_CHECK\_ERROR***("40003","token认证失败"),  
 ;  
  
 private String code;  
  
 private String desc;  
  
 Code(String code, String desc) {  
 this.code = code;  
 this.desc = desc;  
 }  
  
 public String getCode() {  
 return code;  
 }  
  
 public String getDesc() {  
 return desc;  
 }  
}

## 2、response

package com.duodian.admore.zhaobutong.bean;  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
/\*\*  
 \* @author fengchuanbo  
 \*/  
@Setter  
@Getter  
@AllArgsConstructor  
@NoArgsConstructor  
public class Response<T> {  
  
 /\*\*  
 \* 返回code  
 \*/  
 private String code;  
 /\*\*  
 \* 返回描述  
 \*/  
 private String desc;  
 /\*\*  
 \* 返回数据  
 \*/  
 private T data;  
  
 public Response(String code,String desc) {  
 this.code = code;  
 this.desc = desc;  
 }  
  
 public Response(Code code) {  
 this.code = code.getCode();  
 this.desc = code.getDesc();  
 }  
  
 public Response(Code code, T data) {  
 this.code = code.getCode();  
 this.desc = code.getDesc();  
 this.data = data;  
 }  
  
 /\*\*  
 \* 成功响应  
 \* @param t  
 \* @param <T>  
 \* @return  
 \*/  
 public static <T> Response<T> success(T t){  
 return new Response<>(Code.***OK***, t);  
 }  
  
 /\*\*  
 \* 成功响应，date为空  
 \* @return  
 \*/  
 public static Response success(){  
 return new Response(Code.***OK***);  
 }  
  
 /\*\*  
 \* 参数错误  
 \* @return  
 \*/  
 public static Response illegalArgument(){  
 return new Response(Code.***illegalArgument***);  
 }  
  
  
 /\*\*  
 \* 自定义返回 T code 和描述  
 \* @param code  
 \* @param desc  
 \* @return  
 \*/  
 public static <T> Response of(String code,String desc, T t){  
 return new Response(code,desc,t);  
 }  
  
  
 /\*\*  
 \* 自定义返回  
 \* @param code  
 \* @param desc  
 \* @return  
 \*/  
 public static Response of(String code,String desc){  
 return new Response(code,desc);  
 }  
  
  
 /\*\*  
 \* 自定义返回  
 \* @param code  
 \* @param t  
 \* @return  
 \*/  
 public static <T> Response of(Code code, T t){  
 return new Response(code,t);  
 }  
  
 /\*\*  
 \* 自定义返回  
 \* @param code  
 \* @return  
 \*/  
 public static Response of(Code code){  
 return new Response(code);  
 }  
  
  
 /\*\*  
 \* 系统错误  
 \* @return  
 \*/  
 public static Response error() {  
 return new Response(Code.***ERROR***);  
 }  
  
}

## 3、使用

## 1、全局异常的处理

@Slf4j  
@ControllerAdvice  
public class GlobalExceptionHandler {  
  
 @ExceptionHandler(value = Throwable.class)  
 @ResponseBody  
 public Response<?> handler(Throwable t){  
 ***log***.error("系统错误", t);  
 return Response.error();  
 }  
  
}

## 2、普通成功的返回 比如save

@PostMapping("getToken")  
public Response<?> getToken(@RequestParam String code,@RequestParam String encryptedData,@RequestParam String iv){

if (openid == null){  
 return Response.of(Code.***SESSION\_CHECK\_ERROR***);  
 }

String token = AesUtils.LoginEncrypt(zbtUser.getId() + "#" + zbtUser.getOidMd5());  
 return Response.success(token);  
}

# 6、状态码返回中文

@Data  
@Accessors(chain = true)  
@JsonInclude(JsonInclude.Include.***NON\_NULL***)  
public class UserInfoToAdminData {  
  
 public static final Integer ***NUMBER\_0*** = 0;  
 public static final Integer ***NUMBER\_1*** = 1;  
 public static final Integer ***NUMBER\_2*** = 2;  
 public static final Integer ***NUMBER\_3*** = 3;

@ApiModelProperty(value = "删除状态 0 删除 1可用，2 异常")  
private Integer status ;

@Transient  
private String statusDes ;

public String getStatusDesc() {  
  
 if(status==null){  
 return "未知";  
 }  
 if(status.compareTo(***NUMBER\_0***)==0){  
 return "禁止";  
 }else if(status.compareTo(***NUMBER\_1***)==0){  
 return "正常";  
 }else if(status.compareTo(***NUMBER\_2***)==0){  
 return "异常";  
 }else {  
 return "未知";  
 }  
}