# 实体：

不就是个记录单元的表吗？用户表？商品表？，，，，，

# 实体注解

@ContentEntityType(

\* id = "user",实体类型唯一标识符。

\* label = @Translation("User"),实体标题用于UI，@Translation表明其可被翻译。

\* handlers = {处理器，实现实体的各种功能。

\* "storage" = "Drupal\user\UserStorage",

\* "storage\_schema" = "Drupal\user\UserStorageSchema",

\* "access" = "Drupal\user\UserAccessControlHandler",

\* "list\_builder" = "Drupal\user\UserListBuilder",

\* "views\_data" = "Drupal\user\UserViewsData",

\* "route\_provider" = {

\* "html" = "Drupal\user\Entity\UserRouteProvider",

\* },

\* "form" = {

\* "default" = "Drupal\user\ProfileForm",

\* "cancel" = "Drupal\user\Form\UserCancelForm",

\* "register" = "Drupal\user\RegisterForm"

\* },

\* "translation" = "Drupal\user\ProfileTranslationHandler"

\* },

\* admin\_permission = "administer users",实体的管理权限。

\* base\_table = "users",实体的基础表。

\* data\_table = "users\_field\_data",实体的数据表。

\* label\_callback = "user\_format\_name",

\* translatable = TRUE,  实体是否可被翻译。

\* entity\_keys = {实体键的数组。基础表的字段就是这些

\* "id" = "uid",

\* "langcode" = "langcode",

\* "uuid" = "uuid"

\* },

\* links = {  使用URI模板语法的Link模板，它是一个数组。

\* "canonical" = "/user/{user}",

\* "edit-form" = "/user/{user}/edit",

\* "cancel-form" = "/user/{user}/cancel",

\* "collection" = "/admin/people",

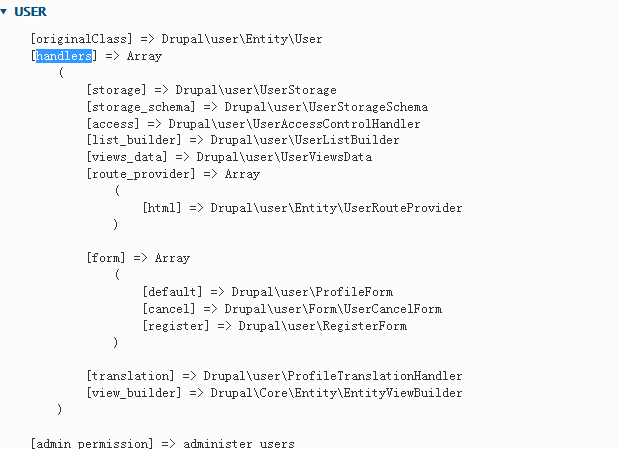
\* },

\* field\_ui\_base\_route = "entity.user.admin\_form",字段UI的基本路由。

\* common\_reference\_target = TRUE

\* )

# 实体处理器Handlers



### 1、存储(Storage)

存储处理器实现了EntityStorageInterface接口,EntityStorageInterface继承ContentEntityStorageBase类，这个类中提供了标准的CRUD方法。

### 2、访问(Access)

访问处理器实现了EntityAccessControlHandlerInterface接口，默认使用EntityAccessControlHandler类。

### 3、管理列表(List Builder)

列表实现了EntityListBuilderInterface接口，默认使用EntityListBuilder类。

### 4、视图(View Builder)

View Builder实现了EntityViewBuilderInterface接口，默认使用EntityViewBuilder类。

### 5、表单(Form)

表单处理器实现了EntityFormInterface接口。因为EntityForm类已经实现了这个接口，你可以直接对该类进行扩展，以实现如add、edit和delete等操

# 实体字段

* boolean 布尔型
* changed 修改日期
* created 创建日期
* decimal 数字
* email 电子邮件
* entity\_reference 实体引用
* float 浮点数
* integer 整型
* language 语言
* map 映射
* password 密码
* string\_long 长文本
* timestamp 时间戳
* uri 统一资源定位符
* uuid 唯一id
* comment 评论
* datetime 时间日期
* file 文件
* image 图像
* link 链接
* list\_float 浮点数列表
* list\_integer 整数列表
* list\_string 字符串列表
* path 路径
* telephone 电话
* text 文本
* text\_long 长文本
* text\_with\_summary 带摘要的文本

# 实体API ——CRUD

## 实体检测

检测对象是否是一个实体的实例

if ($object instanceof \Drupal\Core\Entity\EntityInterface) {

内容实体实例检测

if ($entity instanceof \Drupal\Core\Entity\ContentEntityInterface){

获取实体类型

$entity->getEntityTypeId();

节点实例检测

if($entity instanceof \Drupal\node\NodeInterface){

使用entityType获取动态实体类型

$needed\_type = ‘node’;

if ($entity->getEntityTypeId() == $needed\_type){

// 检测实体是否为新实体

$entity->isNew();

// 检测实体的查看权限，默认会对当前登录用户进行这一检测

if ($entity->access(‘view’)){

// 检测一个用户是否可以删除一个实体

if ($entity->access(‘delete’,$account)){

## 2. 获取实体或实体方法的信息(R)

// 使用静态方法

$node = Node::load(1);

// 动态实体类型，entity\_load()加载新单个实体。

$entity = entity\_load($entity\_type,$id);

// 使用存储控制器

$entity = \Drupal::entityTypeManager()->getStorage($entity\_type)->load(1);

// 加载多个实体使用entity\_load\_multiple()

$entity = \Drupal::entityTypeManager()->getStorage($entity\_type)->loadMultiple(array(1,2,3));

// 获取实体id

$entity->id();

// 获取实体bundle

$entity->bundle();

// 获取实体的标签

$entiry->label();

// 获取实体URI

$entity->uri();

// 获取实体的副本

$duplicate = $entity->createDuplicate();

## 3. 创建实体(C)

$node = entity\_create('node', [

'title' => 'My node',

'body' => 'The body content. This just works like this due to the new Entity Field API. It will be assigned as the value of the first field item in the default language.'

]);

// 你也可以使用静态的创建方法

$node = Node::create(['title' => 'The node title']);

// 使用实体管理器(entity manager)

$node = \Drupal::entityTypeManager()->getStorage(‘node’)->create(['type' => 'article', 'title' => 'Another node'));

保存新实体：

为了保存一个新实体(例如导入数据)，可以强制使用isNew标志，请看以下代码。

// 下面的代码试图插入一个节点ID为5的节点，如果这个节点ID已存在则会失败

$node->nid->value = 5;

$node->enforceIsNew(TRUE);

$node->save();

## 4. 修改实体(U)

// 保存一个实体, 这一方法既可用于新实体，也可用于已存在的实体

$entity->save();

## 5. 删除实体(D)

// 删除单个实体

$entity = \Drupal::entityTypeManager()->getStorage(‘node’)->load(1);

$entity->delete();

// 一次删除多个实体

\Drupal::entityTypeManager()->getStorage($entity\_type)->delete(array($id1 => $entity1,$id2 => $entity2));