分类管理

1. 展示全部分类列表

在 `categories.php` 脚本一开始的时候：

```php
// 查询数据
// ========================================

// 查询全部分类信息
$categories = xiu_query('select * from categories');
```

数据绑定到表格中：

```php
<?php foreach ($categories as $item) { ?>
<tr data-id="<?php echo $item['id']; ?>">
  <td class="text-center"><input type="checkbox"></td>
  <td><?php echo $item['name']; ?></td>
  <td><?php echo $item['slug']; ?></td>
  <td class="text-center">
    <a href="javascript:;" class="btn btn-info btn-xs">编辑</a>
    <a href="javascript:;" class="btn btn-danger btn-xs">删除</a>
  </td>
</tr>
<?php } ?>
```

> 源代码: step-48

MAKE IT BETTER

2. 删除分类

2.1. category-delete.php 处理删除业务

在 `admin` 目录中添加 `category-delete.php` 脚本文件，处理分类的删除逻辑，具体实现参考删除文章的实现：

```php
<?php
/**
 * 删除分类
 */

require '../functions.php';

if (!empty($_GET['id'])) {
  // 拼接 SQL 并执行
  xiu_execute(sprintf('delete from categories where id in (%s)', $_GET['id']));
}

// 获取删除后跳转到的目标链接，优先跳转到来源页面，否则默认跳转到列表页
$target = empty($_SERVER['HTTP_REFERER']) ? '/admin/categories.php' : $_SERVER['HTTP_REFERER'];
header('Location: ' . $target);
```

2.2. 绑定单个删除按钮链接

回到 `categories.php` 文件中，在绑定表格数据的位置，修改最后一列的删除按钮的链接地址：

```php
<td class="text-center">
  <a href="javascript:;" class="btn btn-info btn-xs">编辑</a>
  <a href="/admin/category-delete.php?id=<?php echo $item['id']; ?>" class="btn btn-danger btn-xs">删除</a>
</td>
```

> 源代码: step-49

MAKE IT BETTER

3. 批量删除

类似于文章管理操作，分类也应该有批量删除操作，实现方式与文章管理的方式相同，都是借助 JavaScript 将选中的行数据 ID 记录起来，然后通过一个链接，将所有选中的 ID 全部作为参数传递上去。

直接拷贝之前的实现即可：

```
$(function () {
  // 获取所需操作的界面元素
  var $btnDelete = $('.btn-delete')
  var $thCheckbox = $('th > input[type=checkbox]')
  var $tdCheckbox = $('td > input[type=checkbox]')

  // 用于记录界面上选中行的数据 ID
  var checked = []

  /**
   * 表格中的复选框选中发生改变时控制删除按钮的链接参数和显示状态
   */
  $tdCheckbox.on('change', function () {
    var $this = $(this)

    // 为了可以在这里获取到当前行对应的数据 ID
    // 在服务端渲染 HTML 时，给每一个 tr 添加 data-id 属性，记录数据 ID
    // 这里通过 data-id 属性获取到对应的数据 ID
    var id = parseInt($this.parent().parent().data('id'))

    // ID 如果不合理就忽略
    if (!id) return

    if ($this.prop('checked')) {
      // 选中就追加到数组中
      checked.push(id)
    } else {
      // 未选中就从数组中移除
      checked.splice(checked.indexOf(id), 1)
    }

    // 有选中就显示操作按钮，没选中就隐藏
    checked.length ? $btnDelete.fadeIn() : $btnDelete.fadeOut()

    // 批量删除按钮链接参数
    // search 是 DOM 标准属性，用于设置或获取到的是 a 链接的查询字符串
    $btnDelete.prop('search', '?id=' + checked.join(','))
  })

  /**
   * 全选 / 全不选
   */
  $thCheckbox.on('change', function () {
    var checked = $(this).prop('checked')
    // 设置每一行的选中状态并触发 上面 👆 的事件
    $tdCheckbox.prop('checked', checked).trigger('change')
  })
})
```

> 源代码: step-50

MAKE IT BETTER

4. 新增分类

这里我们将新增和查询放在同一个页面中完成，但是实现方式也是一样的，只是在一个页面中执行过程中要同时做多件事，目的就是提升业务的复杂程度，锻炼逻辑思维。

4.1. 处理表单 HTML 属性

> form / input / label

同样是添加 `<form>` 的 `action` 和 `method` ：

```
<form action="/admin/categories.php" method="post">
  ...
</form>
```

4.2. 接收表单数据并保存

在页面执行一开始的位置，增加逻辑接收表单提交过来的数据并保存：

> 提问：处理表单数据的逻辑应该加载什么位置（查询数据之前 / 查询数据之后）会更合适？

处理逻辑任然是经典的三部曲：接收并校验；持久化；响应

```php
// 处理表单提交
// ========================================

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
  // 表单校验
  if (empty($_POST['slug']) || empty($_POST['name'])) {
    // 表单不合法，提示错误信息（可以分开判断，提示更加具体的信息）
    $message = '完整填写表单内容';
  } else {
    ...
  }
}
```

> 提问：为什么不用 JavaScript 校验表单，或者说有了 JavaScript 校验为什么还要在服务端校验？

保存数据并响应结果：

```php
// 表单合法，数据持久化（通俗说法就是保存数据）
$sql = sprintf("insert into categories values (null, '%s', '%s')", $_POST['slug'], $_POST['name']);
// 响应结果
$message = xiu_execute($sql) > 0 ? '保存成功' : '保存失败';
```

4.3. 操作结果展示

还是和之前的方式相同：

```php
<?php if (isset($message)) : ?>
<div class="alert alert-danger">
  <strong>错误! </strong><?php echo $message; ?>
</div>
<?php endif; ?>
```

> 源代码: step-51

但是，仔细一点的同学都会发现，我们提示的信息有两种类别：一种是成功的提示，另一种是失败的提示。

> 作业： 都用红色的提示框提示消息不是很友好，该用什么方式解决？

> 源代码: step-52

4.4. Slug 预览

与文章发布时的实现方式相同，直接复制：

```
/**
 * slug 预览
 */
$('#slug').on('input', function () {
  $(this).next().children().text($(this).val())
})
```

> 源代码: step-53

> 思考： 到目前为止，我们已经有好几段 JavaScript 代码复制的情况了，以后肯定也会更多，应该如何处理？

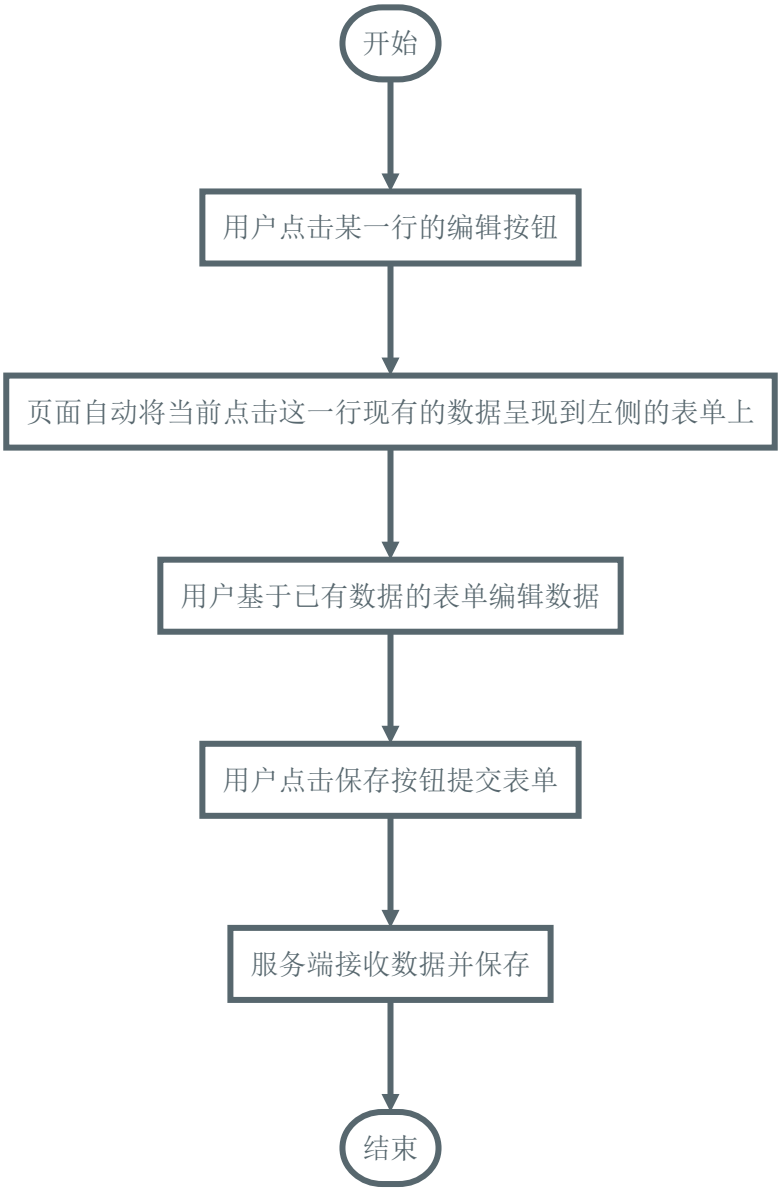
MAKE IT BETTER

5. 编辑分类

> 作为一个软件开发者，提高经验最有效的办法是多尝试，而最快速的方法就是多看看“👀”是怎么跑的（要思考）

作为一个用户，你觉得咱们目前这个页面的设计下，编辑应该是一个怎样的流程？

以下是我的分析结果：



这个过程中除了**服务端接收数据并保存**这一环节，其余的全部是在**客户端**完成的，所以还是 **JavaScript** 出场：

思路：给界面上每一行的编辑按钮注册点击事件，在事件触发过后拿到当前点击行的数据，设置到表单元素上：

```
/**
 * 编辑分类
 */
$('.btn-edit').on('click', function () {
  // 变量本地化（效率）
  var $tr = $(this).parent().parent()
  var $tds = $tr.children()

  // 拿到当前行数据
  var name = $tds.eq(1).text()
  var slug = $tds.eq(2).text()

  // 将数据放到表单中
  $('#name').val(name)
  $('#slug').val(slug)

  // 界面显示变化
  $('form > h2').text('编辑分类')
  $('form > div > .btn-save').text('保存')
})
```

> 源代码: step-54

但是只是添加这样一段 JavaScript 是不可以的，因为我们界面上的表单提交目前都是去新增，想要支持更新，肯定需要调整。

另外作为更新业务，在执行的时候必须明确更新的对象是谁，也就是说我们除了更新过后的 `name` 和 `slug` 还需要将更新的数据 ID 传到服务端。正好利用客户端是否提交了 ID 来判断当前是更新还是新增。

5.1. 服务端更新分类

明确了思路过后，开始改造之前的服务端保存分类逻辑，让其支持更新已有分类：

我们在表单中添加一个隐藏域:

```
<input id="id" name="id" type="hidden">
```

在编辑的时候才加上隐藏域的 `value`

> 隐藏域是专门用来提交数据到服务端的表单元素，但是界面上看不见，也可以用一个普通文本框，然后通过样式，隐藏达到相同效果

服务端接收并保存数据逻辑改造：

```php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
  // 表单校验
  if (empty($_POST['slug']) || empty($_POST['name'])) {
    // 表单不合法，提示错误信息（可以分开判断，提示更加具体的信息）
    $message = '完整填写表单内容';
  } else if (empty($_POST['id'])) {
    // 表单合法，数据持久化（通俗说法就是保存数据）
    // 没有提交 ID 代表新增，则新增数据
    $sql = sprintf("insert into categories values (null, '%s', '%s')", $_POST['slug'], $_POST['name']);
    // 响应结果
    $message = xiu_execute($sql) > 0 ? '保存成功' : '保存失败';
  } else {
    // 提交 ID 就代表是更新，则更新数据
    $sql = sprintf("update categories set slug = '%s', name = '%s' where id = %d", $_POST['slug'], $_POST['name'], $_POST['id']);
    // 响应结果
    $message = xiu_execute($sql) > 0 ? '保存成功' : '保存失败';
  }
}
```

5.2. 客户端设置更新 ID

在编辑按钮的点击事件中，加上 id 的操作：

```
/**
 * 编辑分类
 */
$('.btn-edit').on('click', function () {
  // 变量本地化（效率）
  var $tr = $(this).parent().parent()
  var $tds = $tr.children()

  // 拿到当前行数据
+ var id = $tr.data('id')
  var name = $tds.eq(1).text()
  var slug = $tds.eq(2).text()

  // 将数据放到表单中
+ $('#id').val(id)
  $('#name').val(name)
  $('#slug').val(slug)

  // 界面显示变化
  $('form > h2').text('编辑分类')
  $('form > div > .btn-save').text('保存')
})
```

> 源代码: step-55

5.3. 取消编辑功能

目前情况下，当点击编辑按钮过后，如果希望取消编辑，只能重新刷新页面，体验不佳，最好是能够有个取消编辑的按钮，点击恢复新增表单

> 作业： 实现取消编辑按钮功能

> 源代码: step-56

5.4. Slug 预览同步

由于之前在处理 Slug 预览时，只是监听了 `input` 事件，在后续代码执行过程中改变 `value` 是捕获不到的，所以需要手动 `trigger` 一下：

```
$('#slug').val('').trigger('input')
```

> 源代码: step-57

> 作业： 参考分类编辑，实现文章的编辑（文章更新）