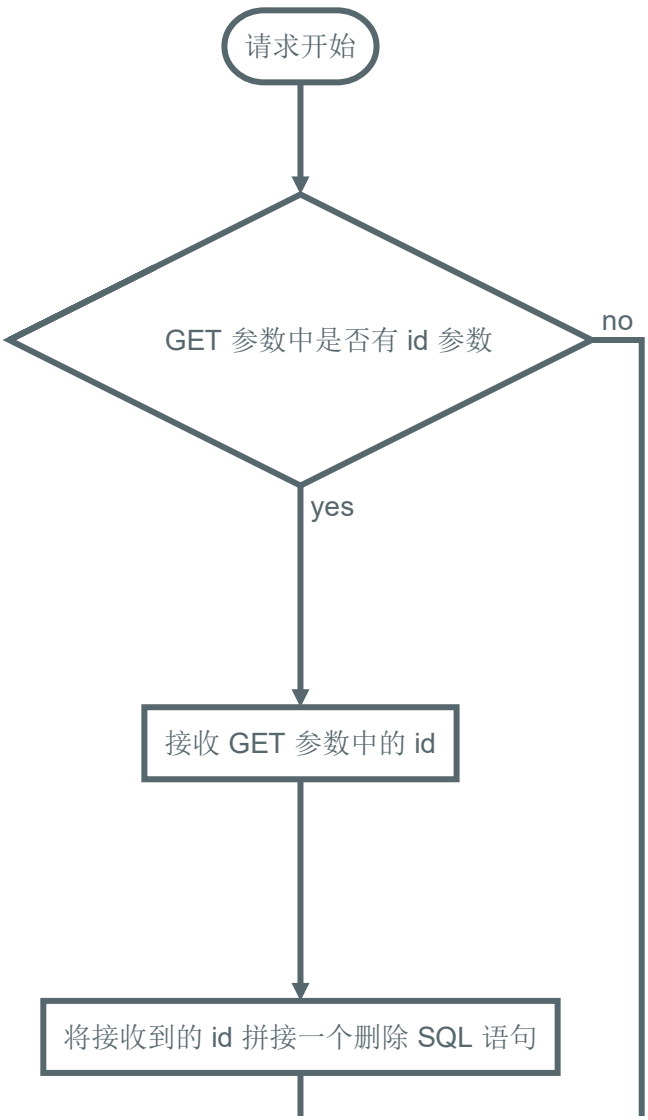
管理后台删除文章

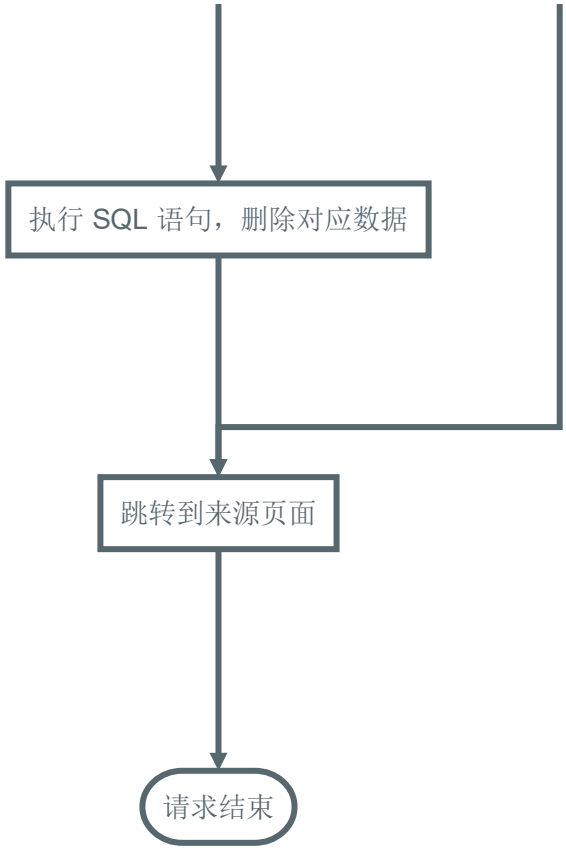
在 `admin` 目录下新建一个 `post-delete.php` 文件，专门处理文章的删除业务。

<!-- 约定这个页面处理两种类型的删除业务：

1. 单条删除，接收要删除的文章 ID
2. 批量删除，接收以逗号分隔的多条文章 ID 字符串 -->

1. 处理流程





MAKE IT BETTER

2. 单条删除

2.1. 接收参数

```php
if (!empty($_GET['id'])) {
  // 接收到要删除的 ID
}
```

2.2. 拼接 SQL 语句

```php
$sql = sprintf('delete from posts where id = %d', $_GET['id']);
```

> 提示：如果需要做软删除可以通过将指定文章的状态设置为 `trashed`

```php
$sql = sprintf("update posts set status='trashed' where id = %d", $items);
```

2.3. 执行 SQL 语句

由于之前封装的 `xiu_query()` 函数只适用于查询操作，对于增删改一类非查询的操作还是需要重新封装一个函数，具体代码如下：

```php
/**
 * 执行一个非查询语句，返回执行语句后受影响的行数
 * @param  string  $sql 非查询语句
 * @return integer      受影响的行数
 */
function xiu_execute ($sql) {
  // 获取与数据库之间的连接
  $connection = xiu_connect();

  // 执行 SQL 语句，获取一个查询对象
  if ($result = mysqli_query($connection, $sql)) {
    // 查询成功，获取执行语句后受影响的行数
    $affected_rows = mysqli_affected_rows($connection);
  }

  // 关闭数据库连接
  mysqli_close($connection);

  // 返回受影响的行数
  return isset($affected_rows) ? $affected_rows : 0;
}
```

2.4. 通过 `xiu_execute()` 函数执行 SQL 语句

```php
if (!empty($_GET['id'])) {
  // 拼接 SQL 并执行
  xiu_execute(sprintf('delete from posts where id = %d', $_GET['id']));
}
```

2.5. 在列表页中绑定单条删除链接

post.php

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

> 源代码: step-34

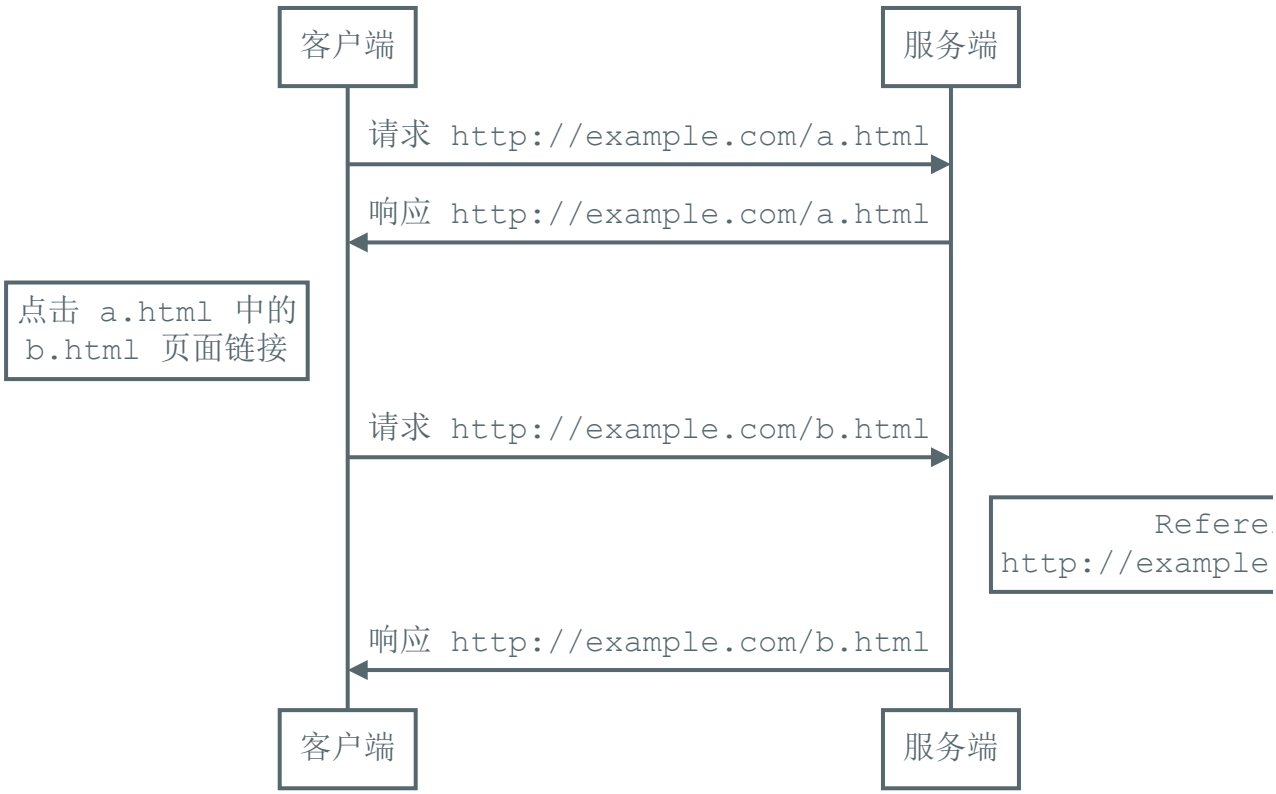
2.6. 跳转到来源

目前这种情况，用户在点击删除链接删除指定内容后，需要手动返回到之前的页面，体验不佳，最好是能在 `post-delete.php` 执行完成过后，自动跳转到之前访问的页面，也就是来源页面。

那么如何获取来源页面的地址，就是我们接下来的重点：

2.6.1. HTTP Referer

在 HTTP 协议中约定：在请求头中包含一个 `Referer` 字段，值就是上一次访问的 URL



2.6.2. 获取 Referer 并跳转

在 PHP 中可以通过 `$_SERVER['HTTP_REFERER']` 获取到 `Referer`

```php
// 获取删除后跳转到的目标链接，优先跳转到来源页面，否则跳转到文章列表
$target = isset($_SERVER['HTTP_REFERER']) ? $_SERVER['HTTP_REFERER'] : 'posts.php';
// 跳转
header('Location: ' . $target);
```

> 源代码: step-35

MAKE IT BETTER

3. 多选批量删除

3.1. 让 `post-delete.php` 支持批量删除

目前 `post-delete.php` 只能删除指定的单个数据，如果需要支持批量删除，可以稍加改造：

1. 约定接收的 `id` 参数是一个以英文半角逗号分隔的 ID
2. 将删除 SQL 语句的 `where` 子句改为 `where id in (%s)`

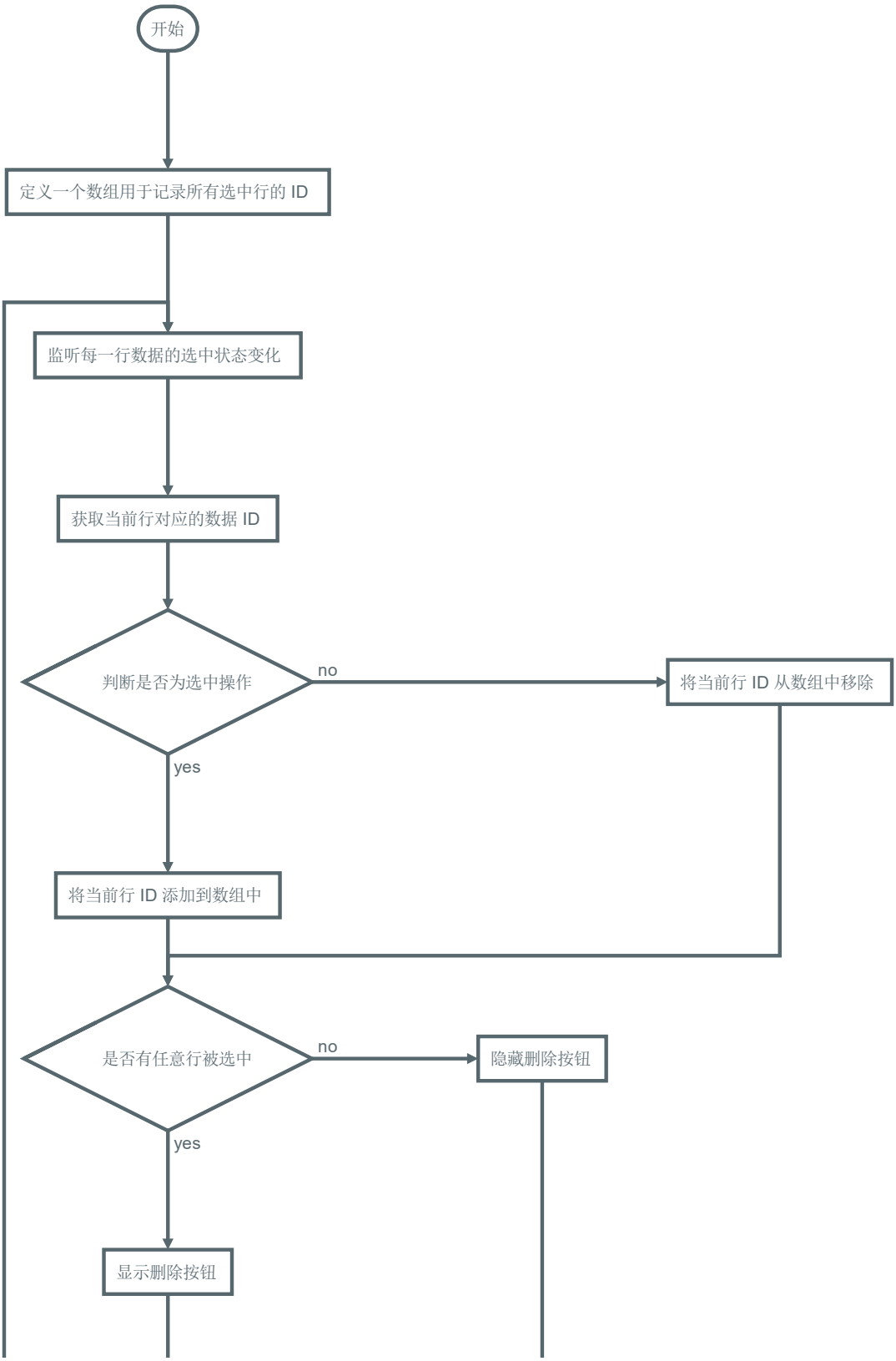
```php
// $id => 22 / 22,23,24
xiu_execute(sprintf('delete from posts where id in (%s)', $_GET['id']));
// $sql => delete from posts where id in (22)
// $sql => delete from posts where id in (22,23,24)
```

> 源代码: step-36

3.2. 选中状态切换过程删除按钮的变换

1. 当选中任意行过后，显示删除按钮
2. 当任意行改变选中状态过后，改变删除按钮的链接地址

对于界面功能需求，可以通过 JavaScript 实现，具体逻辑如下：


开始
定义一个数组用于记录所有选中行的 ID
监听每一行数据的选中状态变化
获取当前行对应的数据 ID
判断是否为选中操作
no
将当前行 ID 从数组中移除
yes
将当前行 ID 添加到数组中
是否有任意行被选中
no
隐藏删除按钮
yes
显示删除按钮

以下是我提供的一种实现，在页面中加入一段 JavaScript 完成上述逻辑：

```
// 获取所需操作的界面元素
var $btnDelete = $('.btn-delete')
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(','))
})
```

3.3. 全选和全不选

```
/**
 * 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-37

> 作业： 完成上述功能模块的实现，思考如何改进上述功能的体验？