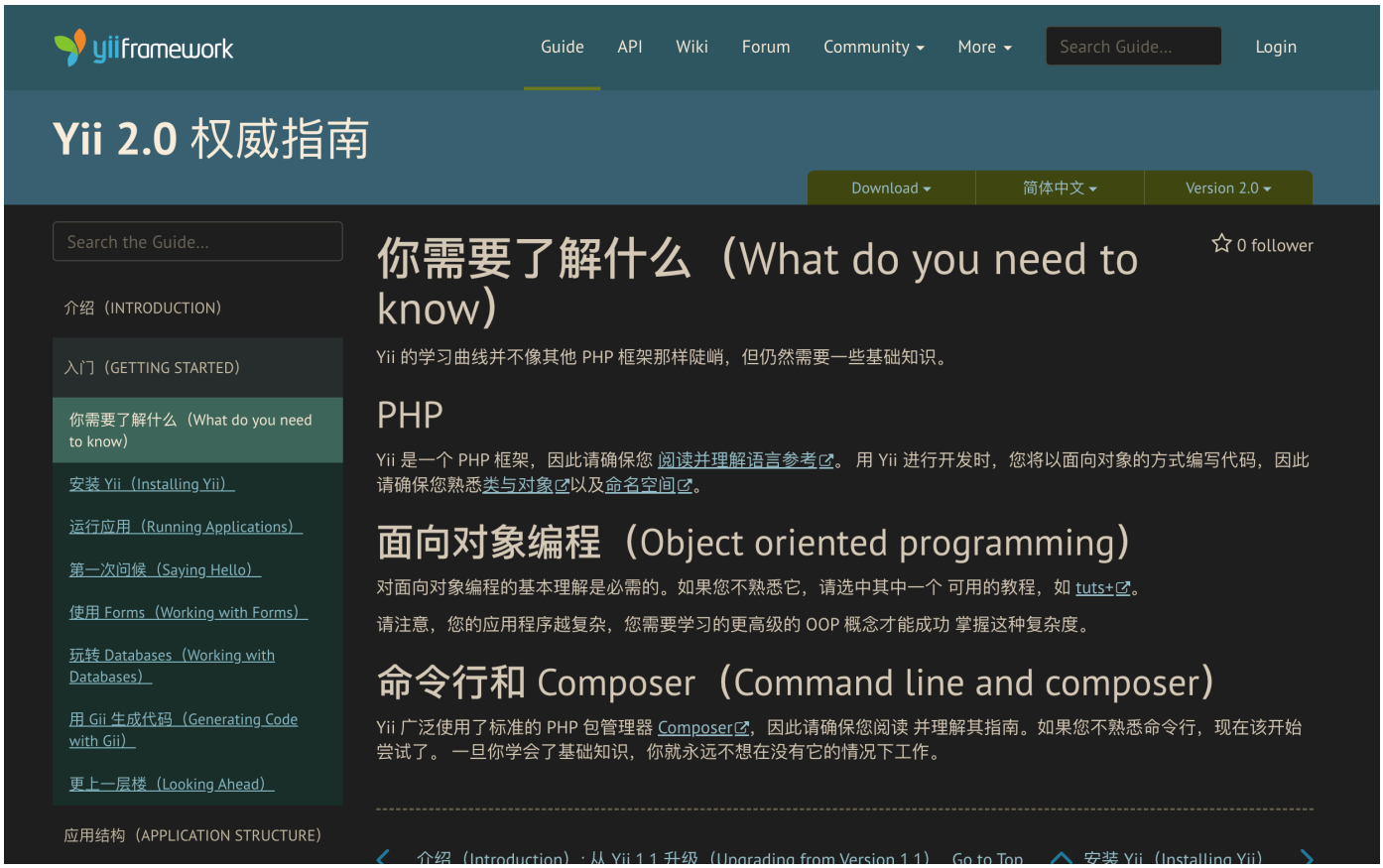
实现文档

环境搭建



搭建 Yii 项目需要按照以下步骤进行，确保已经安装了 PHP 和 Composer，因为 Yii 使用 Composer 进行依赖管理。

步骤 1：安装 Composer

Composer 是 PHP 的依赖管理工具，你可以从 Composer 官网 下载并安装。

步骤 2：安装 Yii

在命令行中运行以下命令来全局安装 Yii：

```
composer global require "yiisoft/yii2:^2.0"
```

步骤 3：创建 Yii 项目

在命令行中，进入你希望创建项目的目录，然后运行以下命令：

```
composer create-project --prefer-dist yiisoft/yii2-app-basic my_yii_project
```

这将在当前目录下创建一个名为 `my_yii_project` 的 Yii 项目。你可以将 `my_yii_project` 替换为你想要的项目名称。

步骤 4：配置数据库

Yii 使用 `config/db.php` 文件来配置数据库连接。打开该文件，根据你的数据库设置修改以下部分：

```
return [
    'class' => 'yii\db\Connection',
    'dsn' => 'mysql:host=localhost;dbname=mydatabase',
    'username' => 'root',
    'password' => 'your_password',
    'charset' => 'utf8',
];
```

确保修改 `dsn`、`username` 和 `password` 分别为你的数据库地址、用户名和密码。

步骤 5：运行迁移

Yii 使用迁移来创建数据库表。在命令行中，进入你的 Yii 项目目录并运行以下命令：

```
php yii migrate
```

这将应用项目中的所有迁移，创建数据库表。

步骤 6：运行应用

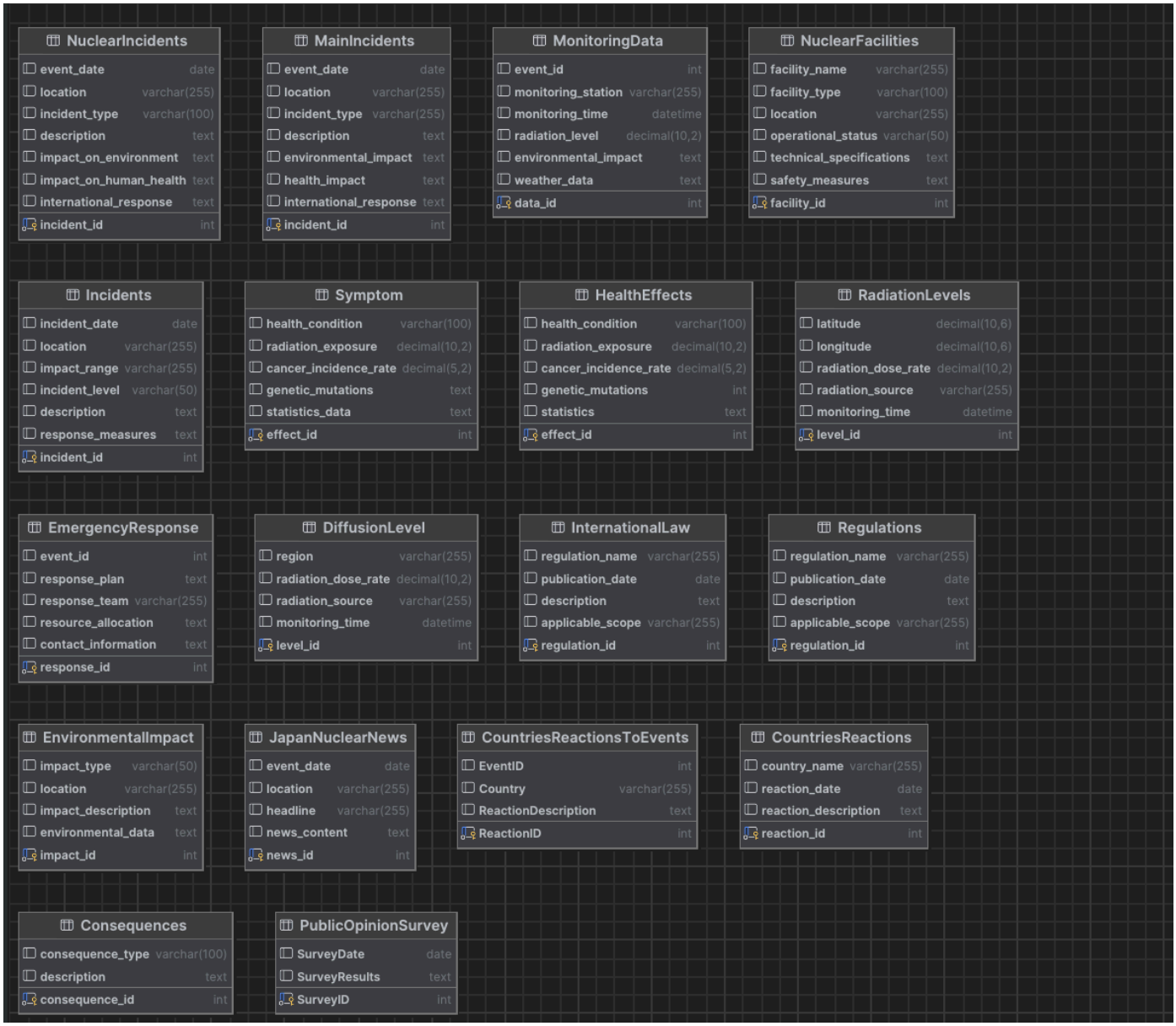
在命令行中，进入你的 Yii 项目目录并运行以下命令：

```
php yii serve
```

这将启动 PHP 内置服务器，并默认监听 `localhost:8080`。打开浏览器并访问 `http://localhost:8080`，你应该能够看到 Yii 默认生成的欢迎页面。

Yii 官方文档。

数据库设计



步骤 1：需求分析

1. **明确需求：** 确定项目的需求，包括数据存储和检索的需求，以及与其他组件的集成需求。
2. **收集数据：** 收集所有与项目相关的数据，包括实体、属性、关系等。

步骤 2：概念设计

1. **标识实体：** 根据需求确定项目中的实体，这可以是人、地点、物品等。
2. **确定属性：** 为每个实体确定属性，这些属性描述了实体的特征。
3. **标识关系：** 确定实体之间的关系，包括一对一、一对多、多对多等关系。

步骤 3：逻辑设计

1. **规范化：** 将数据表规范化，以减少数据冗余和维护表之间的一致性。
2. **选择主键：** 为每个表选择适当的主键，确保唯一标识每条记录。
3. **选择数据类型：** 为每个字段选择适当的数据类型，确保数据的完整性和节省存储空间。

步骤 4：物理设计

1. **确定存储引擎：** 选择适当的数据库管理系统（DBMS）和存储引擎。
2. **表空间分配：** 设计表的存储结构，包括索引、分区等。

步骤 5：实施设计

1. **创建表：** 使用 SQL 脚本或图形界面工具创建数据库表。
2. **添加约束：** 添加必要的约束，如主键、外键、唯一性约束等。
3. **创建索引：** 为加速检索操作，创建适当的索引。

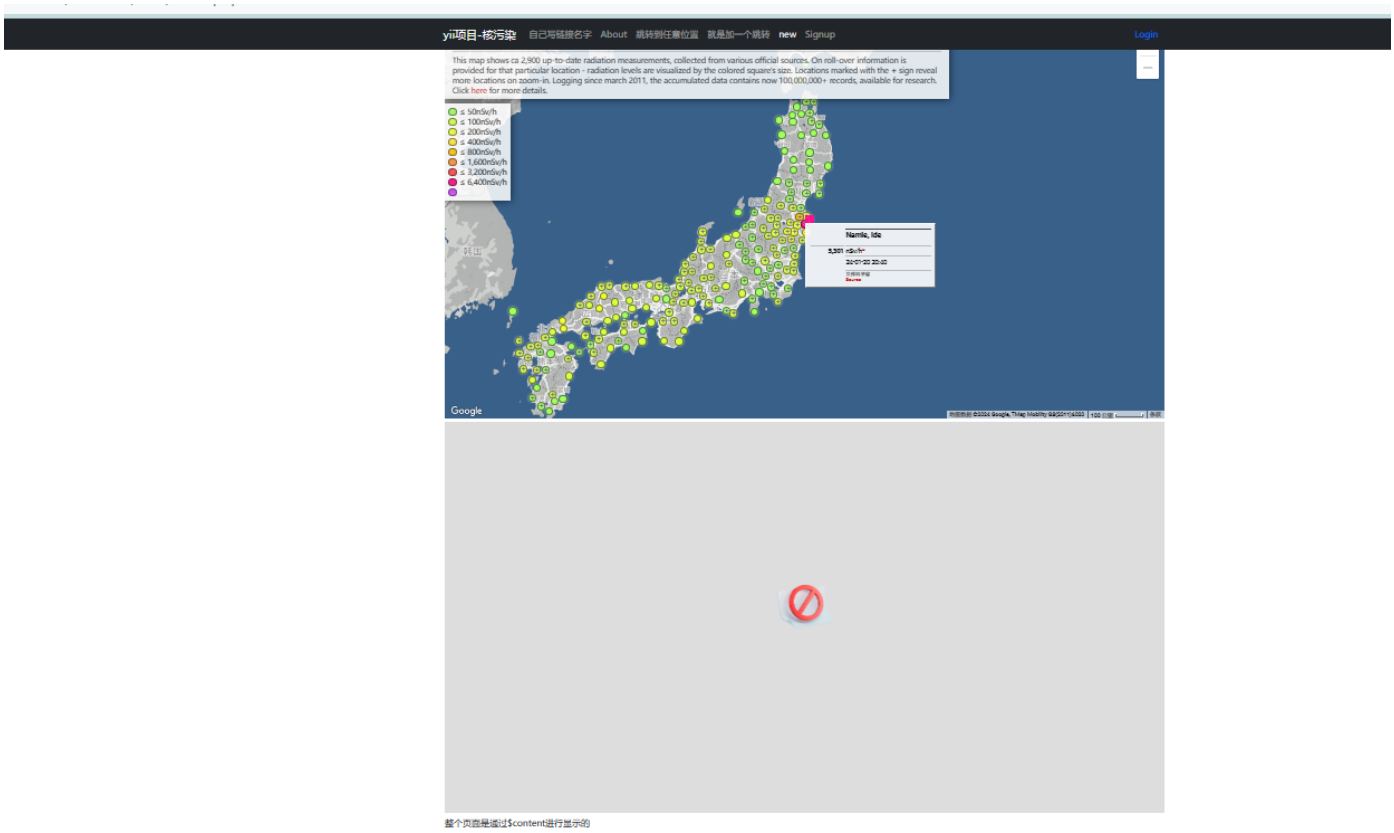
步骤 6：测试和优化

1. **测试：** 对数据库进行测试，确保表现符合预期，并且满足所有需求。
2. **优化：** 根据测试结果进行数据库性能优化，包括查询优化、索引优化等。

步骤 7：文档化设计

1. **文档化：** 编写数据库设计文档，包括表结构、关系图、约束等信息，以便未来维护和参考。
2. **版本控制：** 如果可能，将数据库设计纳入版本控制系统，以便跟踪和管理变更。

数据爬取



使用 Python 进行数据爬取的一般流程包括以下步骤：

步骤 1：确定目标

1. **选择爬取目标：** 确定你希望爬取的数据来源，可能是网页、API、数据库等。
2. **了解目标结构：** 分析目标网站或数据源的结构，确定数据的位置、格式和获取方式。

步骤 2：选择爬取工具

1. **选择爬虫框架：** 根据目标选择合适的爬虫框架，如 Scrapy、Beautiful Soup、Requests 等。
2. **安装相关库：** 根据选定的爬虫框架，安装相应的 Python 库。

步骤 3：编写爬虫代码

1. **发送请求：** 使用库发送 HTTP 请求获取网页内容。

```
import requests

url = 'https://example.com'
response = requests.get(url)
```

2. **解析页面：** 使用 Beautiful Soup 或其他解析库解析 HTML 或 XML 页面。

```
from bs4 import BeautifulSoup

soup = BeautifulSoup(response.text, 'html.parser')
```

3. **定位数据：** 使用选择器或解析方法定位目标数据。

```
data = soup.find('div', class_='target-class').text
```

步骤 4：存储数据

1. **选择存储方式：** 确定数据存储方式，可以是文本文件、数据库、CSV 文件等。
2. **编写存储代码：** 将爬取到的数据存储到选定的位置。

```
with open('output.txt', 'w', encoding='utf-8') as file:
    file.write(data)
```

步骤 5：处理异常

1. **异常处理：** 添加适当的异常处理机制，防止爬虫因为网络问题或其他原因中断。

```
try:
    # 爬取代码
except Exception as e:
    print(f'Error: {e}')
```

步骤 6：设置爬取策略

1. **设置延迟：** 避免对目标服务器造成过大的负载，设置合适的爬取延迟。

```
import time

time.sleep(2)  # 休眠2秒
```

2. **设置用户代理：** 有些网站可能对爬虫进行检测，可以设置合适的用户代理。

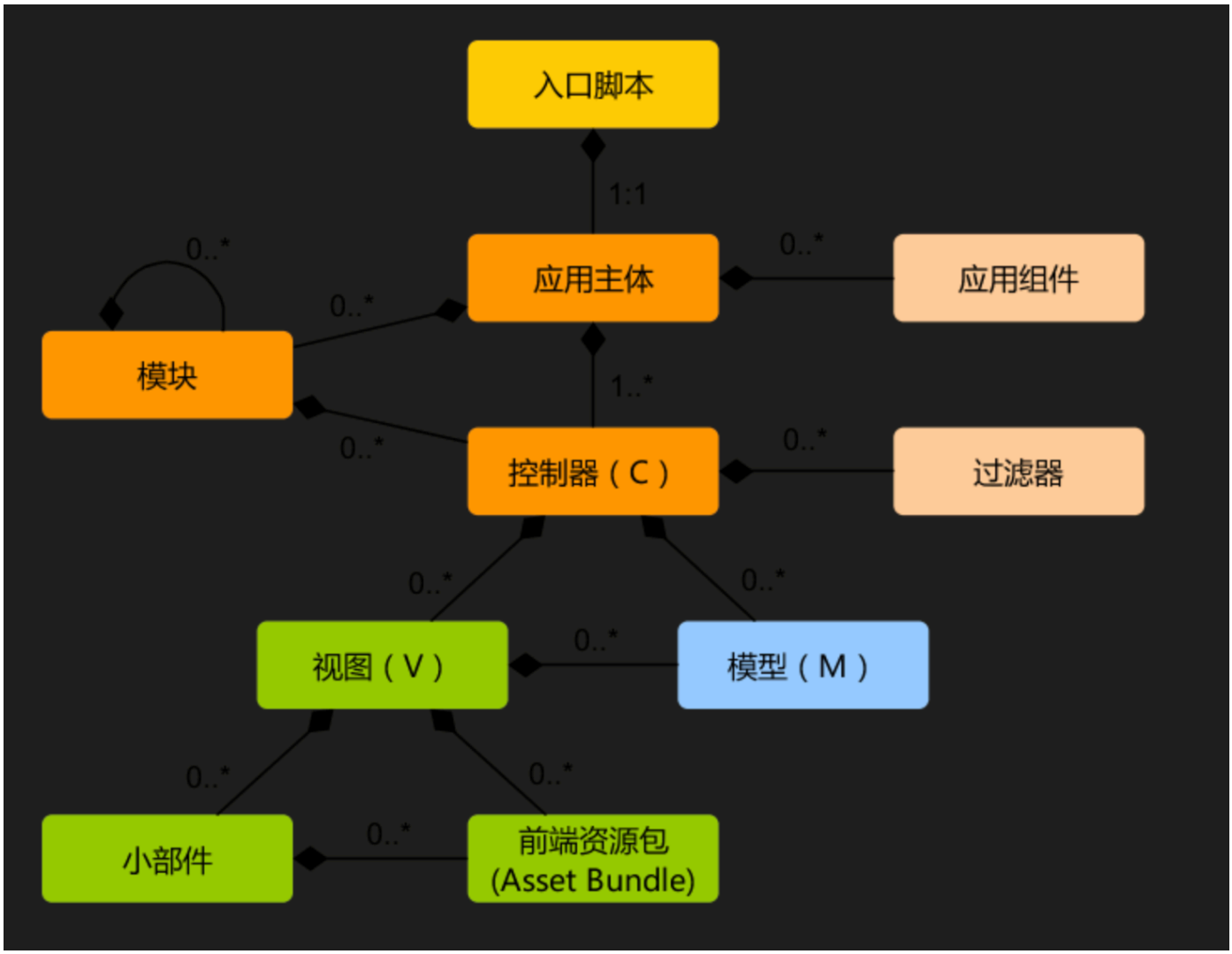
```
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'}
response = requests.get(url, headers=headers)
```

步骤 7：运行爬虫

在命令行或脚本中运行编写的爬虫代码：

```
python your_spider_script.py
```

开发细节



在 Yii 2 中，实现简单的增删改查（CRUD）操作涉及到创建模型（Model）、视图（View）和控制器（Controller）。以下是一个简单的 Yii 2 项目示例，包括一个 "Task" 模型用于表示任务，并实现了增加、删除、修改和查询任务的基本功能。

步骤 1：创建数据库表

```
php yii migrate/create create_task_table
```

```
// 文件路径：migrations/mXXXXXXXXXXXX_create_task_table.php

use yii\db\Migration;

/**
 * Handles the creation of table `{{%task}}`.
 */
class mXXXXXXXXXXXX_create_task_table extends Migration
{
    /**
     * {@inheritdoc}
     */
    public function safeUp()
    {
        $this->createTable('{{%task}}', [
            'id' => $this->primaryKey(),
            'title' => $this->string()->notNull(),
            'description' => $this->text(),
            'created_at' => $this->dateTime(),
            'updated_at' => $this->dateTime(),
        ]);
    }

    /**
     * {@inheritdoc}
     */
    public function safeDown()
    {
        $this->dropTable('{{%task}}');
    }
}
```

运行迁移：

```
php yii migrate
```

步骤 2：创建模型

创建一个 Task 模型，用于表示任务。在命令行中运行：

```
php yii gii/model --tableName=task --modelClass=Task
```

步骤 3：创建控制器

创建一个 TaskController 控制器，用于处理与任务相关的操作。在命令行中运行：

```
php yii gii/controller --controllerClass=TaskController
```

编辑生成的控制器代码，添加以下方法：

```php
// 文件路径：controllers/TaskController.php

use yii\web\Controller;
use app\models\Task; // 根据实际路径调整命名空间

class TaskController extends Controller
{
    public function actionIndex()
    {
        $tasks = Task::find()->all();
        return $this->render('index', ['tasks' => $tasks]);
    }

    public function actionCreate()
    {
        $task = new Task();

        if ($task->load(Yii::$app->request->post()) && $task->save()) {
            return $this->redirect(['index']);
        }

        return $this->render('create', ['task' => $task]);
    }

    public function actionUpdate($id)
    {
        $task = Task::findOne($id);

        if ($task->load(Yii::$app->request->post()) && $task->save()) {
            return $this->redirect(['index']);
        }

        return $this->render('update', ['task' => $task]);
    }

    public function actionDelete($id)
    {
```

```
        Task::findOne($id)->delete();
        return $this->redirect(['index']);
    }
}
```

步骤 4：创建视图

创建与控制器中的操作对应的视图文件：

- `views/task/index.php`

```
<?php foreach ($tasks as $task): ?>
    <p><?= $task->title ?></p>
<?php endforeach; ?>

<a href="<?= Yii::$app->urlManager->createUrl(['task/create']) ?>">Add Task</a>
```

- `views/task/create.php`

```
<?php $form = ActiveForm::begin(); ?>

    <?= $form->field($task, 'title')->textInput() ?>
    <?= $form->field($task, 'description')->textarea() ?>

    <div class="form-group">
        <?= Html::submitButton('Create', ['class' => 'btn btn-primary']) ?>
    </div>

<?php ActiveForm::end(); ?>
```

- `views/task/update.php`

```
<?php $form = ActiveForm::begin(); ?>

    <?= $form->field($task, 'title')->textInput() ?>
    <?= $form->field($task, 'description')->textarea() ?>

    <div class="form-group">
        <?= Html::submitButton('Update', ['class' => 'btn btn-primary']) ?>
    </div>

<?php ActiveForm::end(); ?>
```

步骤 5：配置路由

```
// 文件路径：config/web.php

return [
    'id' => 'basic',
    // ...
```

```
    'components' => [
        'urlManager' => [
            'enablePrettyUrl' => true,
            'showScriptName' => false,
            'rules' => [
                // Add your rules here
                // ...
                'task' => 'task/index',
                'task/create' => 'task/create',
                'task/update/<id:\d+>' => 'task/update',
                'task/delete/<id:\d+>' => 'task/delete',
            ],
        ],
    ],
    // ...
];
```