学习时长记录小程序文档

本小程序虽然结构简单，但它在加深了我对xml和Web Service的理解的同时，也是一个我一直想要完成的，能够解决我学习需求的小程序。所以对我而言，写这个小程序的过程，是在乐趣中学习知识，让我对知识的理解深入了许多。

1 项目简介

为了规划好、分配好自己学习和生活的时间，同时为了弥补自己意志力的不足，以时长为一项指标监督自己学习的努力程度，我有着记录自己学习时长的需求。然后，手动记录具有不便捷性。

在大部分人的习惯中，我们的学习具有**地点固定**的特点，因此，我们可以利用这一特点，制作记录自己学习时长的自动化程序。

由此，我制作了这个小程序，下面将对该程序进行详细的介绍。

文件结构介绍

- 技术文档(md+pdf)
- server.py(服务器代码)
- tmplates
    - index.html(前端代码，包含用JavaScript撰写的客户端代码)

运行时只需要在运行server.py后，运行html文件即可

2 需求分析与功能实现

2.1 需求分析

经分析，本项目共需实现如下需求：

- 前端显示：显示计时结果
- 位置信息获取与判定：获取位置信息，并判定位置是否位于指定区域内，作为计时依据
- 计时逻辑实现：根据位置判定结果进行计时
- 各模块间的数据传输

2.2 功能实现

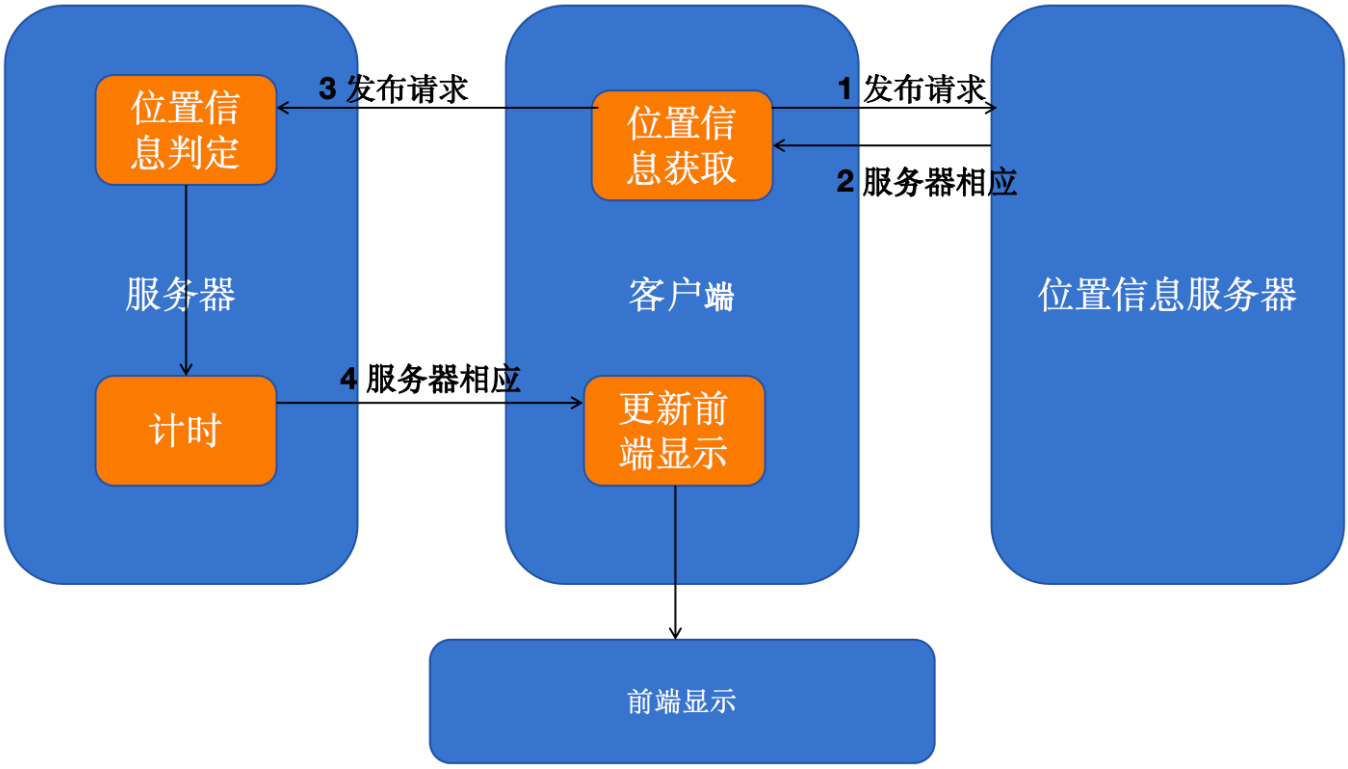
由于本程序的应用场景是记录学习时长，所以本程序并不需要过高的计时精度，以分钟为精度即可。因此，确定整体实现逻辑如下：**每分钟获取位置信息，并进行位置判断，若位于指定区域，计时器的值加一，计时器的值反应总学习分钟数，同时，在进行完如上逻辑后，更新前端显示内容。**

由此，将项目划分为三个具体模块：

- 前端显示
- 客户端模块
    - 调用api获取位置信息
    - 发布位置信息给服务器
    - 获取服务器模块返回的时间信息

    - 更新前端显示的学习时长数据
- 服务器模块
    - 接收客户端模块发布的位置信息
    - 根据位置信息判断是否位于指定区域
    - 根据位置判断结果计时
    - 发布时间数据给客户端模块

实现逻辑如下，旨在借鉴**“客户端-服务器模型”**实现该项目：



3 分模块详细介绍

3.1 前端显示模块

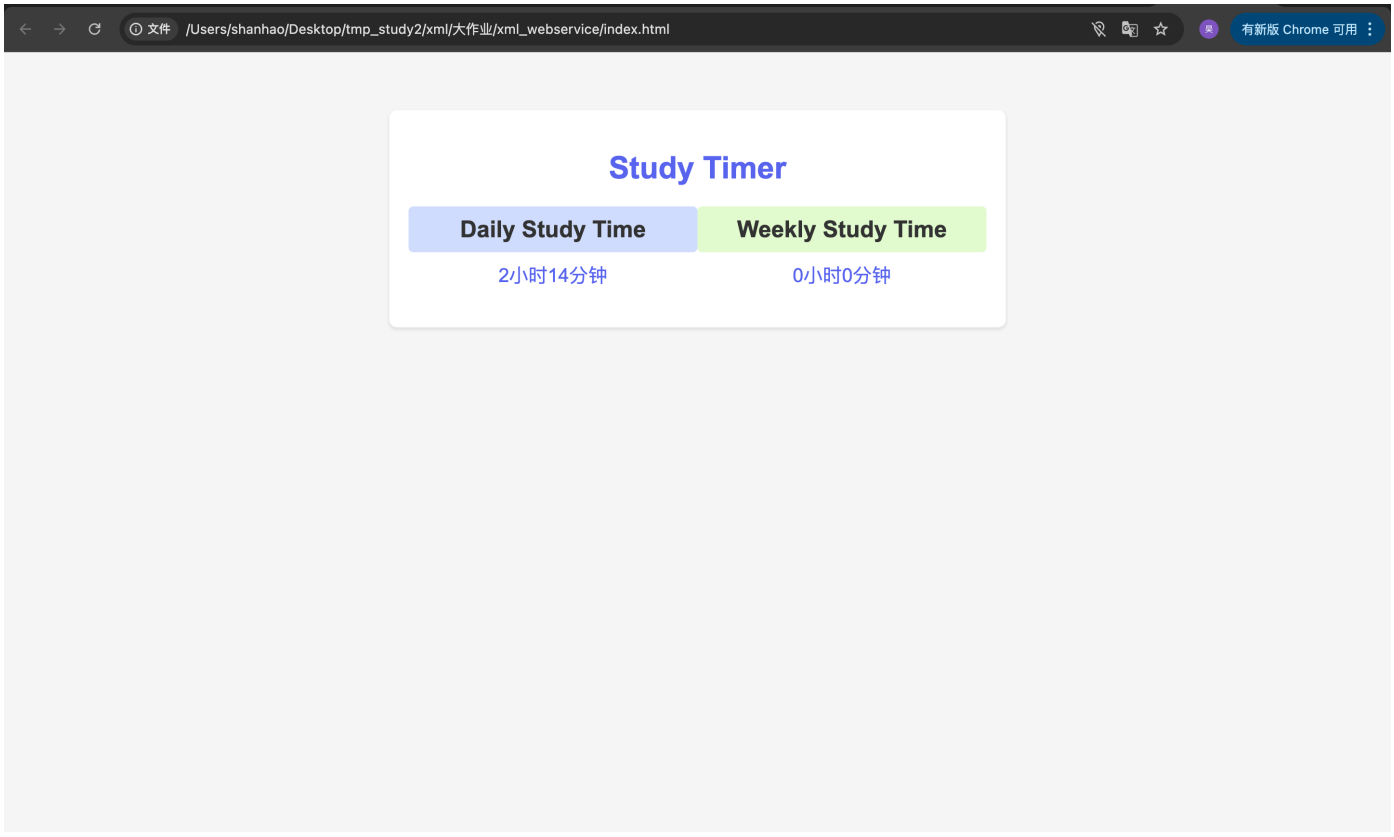
通过html实现，主要用于规范前端显示界面，代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Study Timer</title>
    <style>
        body {
            font-family: 'Arial', sans-serif;
            background-color: #f7f7f7;
            color: #333;
            margin: 0;
            padding: 20px;
        }
```

```
        .container {
            max-width: 600px;
            margin: 40px auto;
            background: #fff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 2px 4px rgba(0,0,0,0.1);
        }
        h1 {
            text-align: center;
            color: #5c67f2;
        }
        .timer-box {
            display: flex;
            justify-content: space-between;
            align-items: center;
            margin: 20px 0;
        }
        .timer-item {
            text-align: center;
            flex: 1;
        }
        .timer-label {
            font-size: 24px;
            font-weight: bold;
            padding: 10px;
            border-radius: 5px;
            margin-bottom: 10px;
        }
        #dailyLabel {
            background-color: #d0e0ff;
        }
        #weeklyLabel {
            background-color: #e0ffd0;
        }
        .time-display {
            font-size: 20px;
            color: #5c67f2;
        }
        button {
            display: block;
            width: 100%;
            padding: 10px;
            background: #5c67f2;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            font-size: 16px;
            margin-top: 20px;
        }
        button:hover {
```

```
            background: #4a54e1;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Study Timer</h1>
        <div class="timer-box">
            <div class="timer-item">
                <div id="dailyLabel" class="timer-label">Daily Study Time</div>
                <div class="time-display" id="dailyTime">0小时0分钟</div>
            </div>
            <div class="timer-item">
                <div id="weeklyLabel" class="timer-label">Weekly Study Time</div>
                <div class="time-display" id="weeklyTime">0小时0分钟</div>
            </div>
        </div>
    </div>
</body>
</html>
```

效果如下：



3.2 客户端模块

通过JavaScript实现

3.2.1 主要功能实现

通过fetchStudyTime函数实现：

```
function fetchStudyTime() {
    navigator.geolocation.getCurrentPosition(function(position) {
        const userPosition = {
            latitude: position.coords.latitude,
            longitude: position.coords.longitude
        };

        fetch('/webservice/timer', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify(userPosition)
        }).then(response => response.json())
        .then(data => {
            updateDisplay(data.dailyTime, data.weeklyTime);
        }).catch(error => {
            console.error('Error:', error);
        });
    }, function(error) {
        console.error("Error Code = " + error.code + " - " + error.message);
    });
}
```

3.2.2 更新前端显示

通过updateDisplay函数实现：

```
function updateDisplay(dailyTime, weeklyTime) {
    const dailyHours = Math.floor(dailyTime / 60);
    const dailyMinutes = dailyTime % 60;
    const weeklyHours = Math.floor(weeklyTime / 60);
    const weeklyMinutes = weeklyTime % 60;

    document.getElementById("dailyTime").textContent = `${dailyHours}小时${dailyMinutes}分
钟`;
    document.getElementById("weeklyTime").textContent = `${weeklyHours}小时
${weeklyMinutes}分钟`;
}
```

3.2.3 整体逻辑

每分钟更新一次位置数据，并根据上述逻辑完成前端显示学习时长的更新，代码架构如下：

```
document.addEventListener("DOMContentLoaded", function() {
            const updateInterval = 60000; // 更新频率为一分钟一次
            let timerInterval;

            function fetchStudyTime() {

            }
```

```
            function updateDisplay(dailyTime, weeklyTime) {

            }

            function startFetching() {
                fetchStudyTime(); // 立即执行一次获取
                timerInterval = setInterval(fetchStudyTime, updateInterval);
            }

            startFetching();
        });
```

3.3 服务器模块

通过Python实现

3.3.1 获取客户端模块发布的位置数据

```
data = request.get_json()  # 解析JSON格式的数据
latitude = data['latitude']
longitude = data['longitude']
```

3.3.1 判断位置是否位于指定区域

通过is_within_bounding_box函数实现

```
def is_within_bounding_box(lat, lon):
  return (bounding_box['south'] <= lat <= bounding_box['north']) and
(bounding_box['west'] <= lon <= bounding_box['east'])
```

3.3.3 计时逻辑

由于我们的实现逻辑中，学习时长在每分钟更新，因此我们只需要以类似实现计数器的思路，以记录学习的“分钟数”的当时记录学习时长，代码如下：

```
if is_within_bounding_box(latitude, longitude):
        # 用户在区域内，更新学习时长
        study_time_data['daily'] += 1
        study_time_data['weekly'] += 1
```

3.3.4 返回学习时长数据给客户端模块

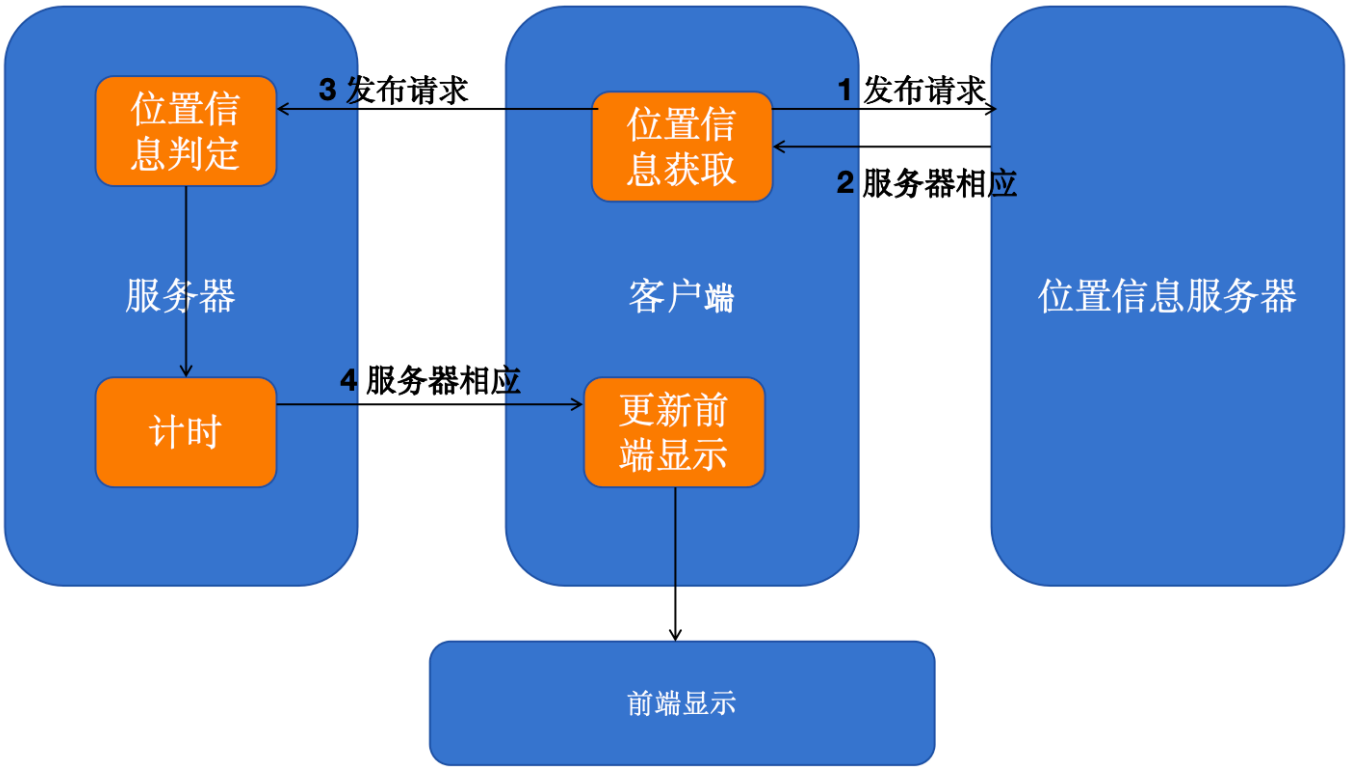
```
return jsonify({
    'dailyTime': study_time_data['daily'],
    'weeklyTime': study_time_data['weekly']
})
```

4 与课程关系

本项目的数据通路灵活运用了课程中的互联网技术。

在调用API获取位置数据时，是加上经历了向服务提供者发布请求、借助服务注册者获取服务的具体信息，获取服务提供者返回信息的过程中，只不过该过程被JavaScript封装为简单的函数，但实际应用中运用到了WebService技术

同时，在我的数据通路中，简单借鉴了分布式计算的思想，在我的项目中，运用到了**“客户端-服务器模型”**，构建如下主体逻辑：

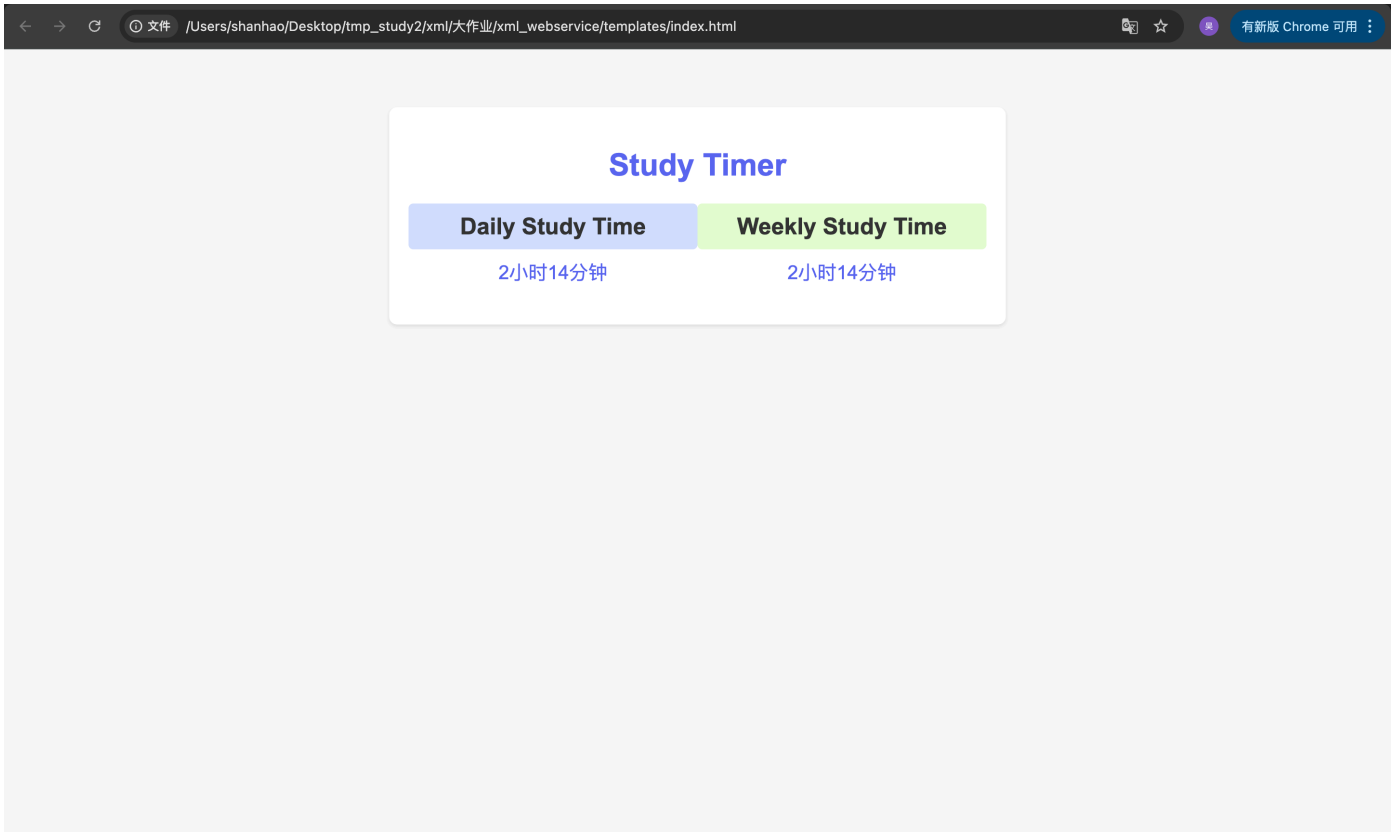


5 反思与不足

由于近期考试频繁，本程序目前并没有得到最好的优化， 仍存在不足：

- 使用浏览器直接打开html无法正确解析，该错误的原理目前正在学习
- 服务器与前端代码的交互并不稳定，部分情况下报错，目前仍在寻找问题
- 计时不稳定，或许是用经纬度记录位置精度不够
- 没有得到封装，使用并不友好

6 效果展示


文件
/Users/shanhao/Desktop/tmp_study2/xml/大作业/xml_webservice/templates/index.html
有新版 Chrome 可用
Study Timer
Daily Study Time
Weekly Study Time
2小时14分钟
2小时14分钟