# Visual Studio

MSDN库 帮助查看器

## 快捷键：

Ctrl+R+E 将字段封装为属性

prop+Tab\*2 get/set属性模板

Ctrl+K+C 注释代码

Ctrl+K+U 取消注释

Ctrl+K+D 代码对齐

Ctrl+J / Alt+-> 弹出只能提示（用于补全代码）

Ctrl+C 不用选择，就可以直接复制光标所在的当前行

Ctrl+D 删除整行代码

Ctrl+Enter 在光标上一行插入一行

Ctrl+Shift+Enter 在光标下一行插入一行

Ctrl+MM 快速隐藏代码

Shift+Home/End 跨选

#region 折叠代码

#endregion

F1 弹出帮助，弹出关键字的文档

F5 代码调试

F10 bug单步调试

F11 逐句调试

F12 查看关键字，对象源代码

.Net 指 .Net FrameWork 框架，是一种应用，一种技术，C#开发基于 .Net 平台的应用

.Net 交互模式：C/S 客户机/服务器、B/S 浏览器/服务器

C# 不支持多重继承（C++支持）

C#支持大小写

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Test

{

class Program

{

static vide Main(string[] args)

{

string str = Console.ReadLine(); //用于接收控制台的输入

Console.WriteLine("Hello World"); //向控制台输出内容

Console.Readkey(); //暂停当前程序、输入显示控制台

}

}

}

术语

padding 内边距

border 边框

margin 外边距

element 内容

sidebar 侧边栏

dotted 点线

solid 实线

dashed 虚线

margin-left 左外边距

C#

C#的独特之处

在类型的外部不能声明全局变量

没有全局函数声明在类型声明的外部

没有默认的返回类型，所有方法必须包含返回类型或void

不管嵌套级别如何，都不能在第一个名称的有效范围内声明另一个同名的局部变量

C#基础

基本语法

数据结构

类型转换

变量

常量

运算符

判断

循环

封装

方法

可空类型

数组

字符串

结构体

枚举

类

继承

多态性

运算和重载

接口

命名空间

预处理器指令

正则表达式

异常处理

文件的输入与输出

使用命令行编译器：csc SimpleProgram.cs

块后面不接分号

输出

System.Console.Write(“……”); 无换行

System.Console.WriteLine(“……”); 有换行

**替换标记**

System.Console.write(“Two words are {0} and {1}.”,3,6);

**字符串插值：**

int var1=3;

int var2=6;

System.Console.write($“Two words are {var1} and {var2}.”);

格式化数字字符串：

格式说明符=索引号+对齐说明符+格式字段

格式字段=格式说明符+精度说明符

C#程序就是一组类型声明

基础语法

类

类的声明

方法

创建变量和类的实例

分配内存

类的声明：为类变量的引用分配内存

New:为类变量的实际数据分配内存

实例成员

访问修饰符

private

public

protected

internal

protected internal

类型

类型=名称+结构+行为或约束

类型是“创建数据结构的模板”  
类型的对象/实例化

类型=预定义类型+用户自定义类型

= 值类型+引用类型

存储

栈

LIFO（后进先出）

堆

任意顺序存入和移除

不同类型数据的存储类型

变量

1. 局部变量
2. 字段：隶属于类的变量，必须在类型声明内部声明；
3. 参数
4. 数组元素

声明：命名+类型+内存

乱七八糟的知识

抽象方法

1. 只有声明，没有具体方法的实现
2. 必须在抽象类或接口中定义，需要被子类或实现类实现
3. 目的：将方法的设计和实现分离
4. 当一个类有抽象方法时，该类必须声明为抽象类
5. 将方法设置为抽象的原因：强制子类或实现类必须提供特定的方法实现
6. 抽象类不能被实例化，只能被继承

路由

1. 根据不同的请求路径，将请求导向不同的处理程序或代码逻辑的过程
2. 通过设置路由规则，可以实现URL与处理程序之间的映射关系，使得用户可以通过简单的URL访问到特定的功能。
3. 目的：实现页面之间的导航和跳转，以及对请求进行分发和处理
4. 流程：用户在浏览器输入一个URL，服务器根据该URL匹配到相应的路由规则，并调用相应的处理程序来处理请求。

**var关键字**

1. 使用var声明变量，编译器会根据右侧表达式的类型来推断变量的类型
2. 这意味着，var可以用于声明任何类型的变量（匿名类型、具名类型、基本数据类型）
3. 匿名对象：没有明确类型名称——临时存储一些数据，不必为其定义一个具体的类

**MIME类型**

1. 一种用于标识文件类型的标准，在互联网上传输数据时使用的一种约定
2. 在HTTP头部中指定，来告知客户端如何处理接收到的数据
3. 由主类型和子类型，“主类型/子类型”
4. “text/html”：HTML文档；“image/jpeg”JPEG图像；“application/pdf”PDF文件

**HTTP状态码**

BadRequest（400）

* 表示客户端的请求无效。（服务器收到一个不能被理解或处理的请求）
* 比如请求语法错误、参数缺失、格式不正确等

**HTTP请求**

* GET：
  + 获取数据（向服务端请求特定数据或资源，并将其返回到客户端进行处理和显示）
  + 将请求参数附在URL末尾，以查询字符串的形式发送给服务器
  + GET请求可以被缓存、保存为书签、在浏览器历史记录中回退
  + 对请求参数的长度有限制，适合发送少量的非敏感数据，如搜索关键字等
* POST
  + 提交数据（将客户端的数据发送到服务器进行处理）
  + 将请求参数放在请求的信息体中，而不是URL中
  + POST请求被缓存、保存为书签、在浏览器历史记录中回退
  + 对请求参数的长度没有限制，适合发送大量的数据，如表单提交、文件上传等

**泛型类（Generic Class）**

1. 在类定义中使用类型参数的类
2. 可以创建具有通用行为的类，以便在编译时指定具体的类。
3. 实现对不同类型的数据进行操作，而不需要为每个类型编写独立的类。
4. 目的是为了在类定义中引入一个或多个类型参数，并在类的方法、属性或字段中使用这些类型参数。
5. 好处：在使用泛型类时指定具体的类型，从而使类具有适应不同数据类型的能力。
6. 使用场景：
   1. 需要创建一个可以处理多种类型的数据结构或算法时
   2. 需要在运行时指定具体的类型，并且希望通过编译时类型检查来捕获类型错误时
   3. 希望在不同上下文中使用相同的代码逻辑来处理不同类型的数据时
7. 使用方法：通过实例化泛型类对象，传入具体的类型参数来创建对象。然后，调用泛型类中定义的方法、属性或字段来处理数据。
8. 泛型约束：使用where关键字；对泛型参数进行一些特定的限制，比如只能接受某个特定接口或类作为泛型参数。

**多态性（Polymorphism）**

1. 编译时多态性（静态多态）：函数重载和运行符重载
2. 运行时多态性（动态多态）：继承和方法重写
3. 父类类型的变量可以引用子类对象
4. 作用：先编写通用的代码，具体的实现细节延迟到运行时再确定

**Lambda表达式**

Expression<Func<TEntity,bool>>condition

1. Expression：用于表示表达树的类，System.Linq.Expressions命名空间的一部分，提供了创建、组合和分析表达式树的功能。
2. <Func<TEntity,bool>是一个委托类型，表示接收一个TEntity类型的参数，并返回一个bool类型的结果。
3. condition是一个变量名，表示一个用于条件查询的表达式树，它的类型是Expression<Func<TEntity,bool>>，即一个接收TEntity类型的参数，并返回一个bool类型的表达式树

**Jquery**

$(function()){……}：文档就绪函数，表示在文档加载完成后执行其中的代码。

$()：创建Jquery对象，可以传入CSS选择器字符串、DOM元素以及HTML代码等，与其他方法一起使用

**Jquery插件**

$('#gridList').DataTable()是DataTables插件提供的一个接口，用于将普通的HTML表格转换为交互式数据表格（排序、搜索、分页、自定义布局等）

**回调函数**

1. 作为参数传递给其他函数，并在特定事件发生时被调用
2. 允许我们在适当时间执行代码，以响应事件或完成任务。
3. function(){…}

**DOM**

1. DOM（文档对象模型）
2. 将HTML文档表示为由节点组成的树型结构
3. 定义了访问和操作HTML元素的标准方法和属性
4. 提供了一种统一的方式来处理文档的结构、内容和样式
5. 通过使用DOM，可以使用JavaScript在网页上选择、创建、修改和删除元素
6. 作用：
   1. 实时改变页面内容
   2. 响应用户操作
   3. 动态加载数据
7. 使用方法
   1. 获取元素：
      1. document.getElementById("id");
      2. document.getElementsByClassName();
      3. document.getElementsByTagName("");
      4. document.querySelector("#id")/("标签名");
   2. 操作元素：修改文本内容、添加样式、绑定事件
   3. 创建新元素:document.createElement()创建新元素，并通过appendChild()、insertBefore()等方法将其添加到页面中
   4. 删除元素：removeChild（）

**AJAX**

（Asynchronous JavaScript and XML）

1. 使用JavaScript、XML和HTTP请求与服务器进行异步通信的技术
2. 在不刷新整个页面的情况下，通过异步请求获取服务器数据，并根据数据更新页面的某个部分，从而提高用户体验。
3. 使用场景：
   1. 与服务器进行数据交互并实时更新页面内容时
   2. 需要通过异步请求获取数据或提交数据而无需刷新整个页面时
4. 使用方法：
   1. 创建一个XMLHttpRequest对象或使用JQuery的AJAX方法
   2. 设置请求的URL、请求方法类型、数据类型等参数
   3. 定义请求成功时的回调函数，处理服务器返回的数据
   4. 定义请求失败时的回调函数，处理错误情况
   5. 发送请求，并处理服务器返回的数据

**XMLHttpRequest对象**

1. 用于在后台与服务器进行数据交换的JavaScript API
2. 允许客户端发送HTTP请求并接收来自服务器的响应，从而实现异步通信
3. 使用步骤：
   1. 创建XMLHttpRequest对象：new XMLHttpRequest()
   2. 设置请求参数：open(请求类型，URL，是否使用异步通信)
   3. 注册事件监听器onreadystatechange
      1. 处理服务器响应：根据readystate和status属性判断请求状态，并操作服务器返回的数据
   4. 发送请求：send()

**AJAX和XMLHttpRequest对象**

XMLHttpRequest对象时用于发送请求和接受响应的JavaScript API，而AJAX是一种利用XMLHttpRequest对象实现异步通信的技术。（Ajax是技术，XMLHttpRequest对象

是实现技术的方法）

**$(function () {…})；**

$(function () {…})；==$(document).ready(function () {…}；

在JQuery中，当DOM树加载完成后，即开始执行JavaScript代码。为了确保代码在DOM加载完成后再执行，可以使用ready()方法。该方法在DOM加载完成后触发一个事件，从而执行相应的处理程序。

$()是JQuery()方法的简写形式，除了可以接收选择器字符串作为参数，还可以接收一个函数作为参数。

**sbyte**

数据类型，表示有符号的8位整数值，存储范围：-128~127

Sbyte ?：表示该属性可以存储空值（即可为null）

**Session.Abandon();**

结束当前用户会话并释放所有会话数据的方法。

**$().attr()**

$()：获取元素的函数

.attr()：对选中的元素的属性进行操作

.attr()方法只作用于第一个匹配的元素；如果要对多个元素进行操作，可以使用.each()方法来遍历匹配的元素并逐个执行相应的操作。

**$().modal()**

.modal()显示弹出窗口、对话框、提示框等，阻止用户与页面上的其他内容进行交互，直到用户对模态框进行操作或关闭它。

**委托**

1. 引用一个或多个具有特定参数列表和返回类型的方法
2. 对方法的抽象，类似于函数指针
3. 通过委托，可以将方法作为参数传递给其他方法，或者将方法存储在数据结构中。
4. 作用：
   1. 方法的引用
      1. 通过委托实例来间接调用关联的方法
   2. 回调（在某个条件满足时，通过委托调用相关方法）
   3. 事件处理
   4. 动态方法调用

**静态方法**

1. 不依赖于任何对象，可以在没有创建对象的情况下直接调用
2. 可以访问和修改类的静态成员变量，但不能访问和修改实例成员变量

**私有方法**

只能在定义它的类内部访问，只能被同一类中的其他方法所调用

**正则表达式**

/^(\d{4}-\d{1,2}-\d{1,2})(\s?\d{2}:\d{2}:\d{2})?$/

/：表示正则表达式的开始和结束

^：表示从输入字符串的开头进行匹配

?：前面的字符可以出现零次或一次

$：匹配以……结尾的字符串

/colou?r/匹配color、colour

**onblur**

onblur="checkdate()"在元素失去焦点时出发的JavaScript函数

失去焦点：用户在输入框或其他表单元素中输入内容后，将焦点从该元素移开时。

NFine框架

1. 路由规则的设置：
   1. Global.asax.cs文件中进行设置，在Application\_Start()方法中调用 RouteConfig.RegisterRoutes()方法来注册路由。
2. 区域注册：
   1. AreaRegistration：ASP.NET MVC 中的一个类，用于定义和注册区域（Area）。区域指在应用程序中将控制器和视图组织到单独的名称空间中，便于管理和组织代码。
3. URL（统一资源定位符）：
   1. @Url.Action(“About”,”Home”)” method=”get”

ASP.NET MVC 框架提供的一个辅助方法，用于根据指定的动作和控制器生成对应的URL，

通常用于在视图中生成链接或表单的目标URL。

“About”是动作方法,”Home”是控制器

* 1. 作用：1.定位资源；2.访问资源（浏览器通过URL向服务器发送请求并获得相应的资源）
  2. 使用方法：
     1. 设置控制器+方法
     2. 视图文件
     3. 使用生成的URL来设置表单的提交地址

1. @{}
   1. Razor语法中的代码块，表示“开始一个代码块”，用于在视图中执行C#代码，即一些与界面展示相关的逻辑操作，比如生成URL，设置变量等。
   2. @{}代码块内的C#代码会在服务器端执行
2. Razor语法：
   1. 将C#语言和HTML标记结合
   2. @\* \*@进行注释
3. ActionResult
   1. 抽象类，表示控制器方法的返回结果
   2. 作用：将控制器方法的执行结果包装为一个可返回给客户端的操作结果（如视图、重定向、JSON数据等多种返回类型），开发人员可以将处理结果传递给视图引擎，或者进行页面跳转、数据序列化等操作。
   3. 目的：设置ActionResult对象，是为了告诉MVC框架如何处理控制器方法的返回结果，执行什么操作（如视图、重定向、JSON数据）。
   4. 返回ActionResult对象的代码应该位于控制器方法中，并根据业务需求选择适当的ActionResult类型进行返回。
4. Entity Framework
   1. ORM（对象关系映射）框架，简化.NET应用程序和数据库之间的数据访问
   2. 将数据库中的表映射为.NET对象，并提供了一组API和工具，用于增删改查数据
   3. 将数据访问操作抽象为.NET类和方法的形式
   4. 在NFine框架中设置Entity Framework：
      1. 在项目中添加Entity Framework的引用
      2. 配置数据库连接字符串
         1. Web.config/App.config——NFine.Web/Configs/database.config
         2. 指定要链接的数据库和认证方式等
      3. 创建数据模型
         1. 使用Entity Framework提供的工具（如EF Designer、Code First）创建数据模型，将数据库的表映射为.NET的实体类。这些实体类可以通过LINQ查询进行操作。
5. 命名空间
   1. System ：常用的系统类型和功能
   2. System.Data.Entity：包含了Entity Framework的核心类和接口，用于实现ORM框架
   3. System.Linq：包含了LINQ的相关功能，用于对数据进行查询和操作
   4. NFine.Data：NFine框架自己定义的数据访问相关的类和接口
   5. NFine.Domain.Entity.SystemManage：包含了系统管理相关的领域实体类
6. 上下文
   1. Entity Framework中的重要概念，代表了数据库的上下文环境，负责连接到数据库、跟踪实体对象的变化以及执行数据操作等功能。
      1. 实体（Entity）：在Entity Framework中，实体表示数据库中的表或视图的映射类，通常是个C#类。实体中包含了表中的字段（属性）以及其他实体之间的关联关系。
      2. 实体集合（DbSet）：上下文中的属性，用于访问数据库中的表或视图。
      3. 上下文（Context）：继承自DbContext的类，用于表示数据库的上下文环境。它通常包含了与数据库相关的实体集合（DBSet）以及其他的配置信息。

上下文（school）——DbSet（student的集合、course的集合）——实体（student、course）

1. 仓储类（Repository Class）
   1. 用于实现数据访问层的设计模式
   2. 封装对数据存储的操作，提供高级别的数据访问接口给上层业务逻辑代码使用
   3. 将数据访问细节和业务逻辑分离
   4. 当应用程序需要进行数据库访问或其他数据存储操作时，就应该考虑设置仓储类
   5. 使用方法：
      1. 定义一个仓储类，继承基类
      2. 通过依赖注入将数据库上下文（IDbContext）传递给仓储类的构造函数，从而实现数据访问
      3. 在具体的仓储类中根据实际需求添加特定的方法
2. 构造函数和析构函数
   1. 构造函数（Constructor）
      1. 在创建对象时进行初始化操作（设置对象的初始状态、分配资源等）
      2. 在类中定义，且与类名相同。
      3. 没有返回类型，也不需要显式声明
      4. 在创建对象（使用new关键字）时自动调用与类名相同的构造函数
   2. 析构函数（Destructor）
      1. 在对象销毁之前被自动调用，进行资源释放或其他清理操作
      2. 与类名相同，但在名称前加上~符号。
      3. 没有参数，也没有返回类型

public class DicRepository : RepositoryBase<DicEntity>, IDicRepository

{

public DicRepository()

: base(CommonConfig.dupConnection)

//构造函数，用于初始化对象

//在这里可以进行一些初始化操作，例如设置默认值或分配资源

{

}

}

1. 虚方法（Virtual Method）
   1. 允许子类重写（Override）基类中定义的方法，以实现多态性，使基类定义的行为能够在子类中进行定制化
   2. 作用：
      1. 子类重写父类方法
      2. 父类访问子类变量（不同类型的子类可以以统一的父类方式进行访问）
2. SqlSugarClient

使用SqlSugar进行数据库操作时，需要首先创建一个SqlSugarClient实例，然后对该实例进行增删改查操作。但在某种情况下，我们可能需要在多个地方使用同一个SqlSugarClient实例，如果每次都通过New来创建，会造成代码冗余，且存在线程安全问题。为解决这一问题，使用GetInstance()，可以在全局范围内获取同一个SqlSugarClient实例。

注意，使用GetInstance()来获取实例，需要在程序入口处配置SqlSugar的初始化方法

1. InitKeyType.Attribute

将实体类的主键和自增列的初始化方式配置为使用特性来获取。

1. NFine层次结构
   1. **Common基础结构层**：通用的工具类、帮助类和扩展方法等，常用的功能模块（缓存、数据库操作、文件操作和日志记录）
   2. **Domain应用服务：**系统的核心业务逻辑实现层，负责处理业务需求、验证数据合法性、调用领域对象和协调不同的聚合根之间的关系。对外暴露面向业务的接口，以便其他层或客户端调用。
      1. NFine.Application：应用服务的实现
      2. NFine.Domain：领域模型的定义与实现
      3. NFine.Repository：仓储类的实现

NFine.Application调用 NFine.Domain中的领域对象和领域服务来处理业务逻辑，而NFine.Domain则通过 NFine.Repository中的仓储实现来持久化和检索数据。

* 1. **Web 应用程序：**系统的用户界面展示层，负责接受用户的请求、呈现数据和处理用户的输入等。Web 应用程序层通常包括控制器（Controller）、视图（View）等组件，用于将用户请求转化为领域操作，并将领域操作的结果反馈给用户。
  2. **Web应用服务：**系统的网络服务层，负责处理客户端的请求、协调不同的领域服务和领域对象之间的关系，以及返回合适的数据给客户端。

1. ConfigurationManager
   1. 用于管理应用程序配置信息的类
   2. 允许应用程序读取配置文件中的设置，并提供了方便的方法来访问和操作这些设置。
2. ORM框架—SqlSugar
3. AdminLTE框架
4. "@ViewData["userCode"]"
   1. 在MVC中，控制器负责处理用户请求，然后将数据传递给视图进行展示
   2. ViewData是一个动态对象，用于在控制器和视图之间传递数据。
   3. 可以存储任意类型的数据，并在视图中进行访问。
   4. 在视图中访问ViewData中名为"userCode"的数据，并将其值渲染到视图中相应的位置。

C#高级

特性(Attribute)：

适用于在运行时传递程序中个各种元素（比如类、方法、结构、枚举、组件等）的行为信息的声明性标签。可以通过特性向程序添加声明性信息。一个声明性标签是通过放置在它所应用的元素去前面的方括号来描述的。

用于添加元数据，如编译器指令和注释、描述、方法、类等其他信息。

反射(Reflection)：

是指程序可以访问、检测和修改它本身状态或行为的一种能力。程序集包含模块，而模块包含类型，类型又包含成员。反射则提供了封装程序集、模块和类型的对象。

可以使用反射动态地创建类型的实例，将类型绑定到现有对象，或从现有对象中获取类型。然后可以调用类型的方法或访问其字段和属性。

用途：它允许在运行时查看特性信息；它允许审查集合中的各种类型，以及实例化这些类型；它允许延迟绑定的方法和属性；它允许在运行时创建新类型，然后使用这些类型执行一些任务。

属性(Property)：

是类、结构(structure)和接口(interface)的命名成员。类或结构中的成员变量或方法称为域(Field)。属性是域的扩展，且可使用相同的语法来访问。他们使用访问器(accessors)让私有域的值可被读写或操作。

属性不会确定存储位置。相反，它们具有可读写或计算它们的值得访问器。

索引器(Indexer)：

允许一个对象可以像数组一样使用下标的方式来访问。

当类定义一个索引器时，该类的行为就会像一个虚拟数组一样。就可以使用数组访问运算符[]来访问该类的成员。

用途：索引器行为的声明在某种程度上类似于属性，就像属性。可以使用get和set访问器来定义索引器。但是属性返回或设置一个特定的数据成员，而索引器返回或设置对象实例的一个特定值。（把实例数据分为更小的部分，并索引每个部分，获取或设置每个部分）

委托(Delegate)：

类似C/C++的指针。是存在对某个方法的引用的一种引用类型变量，引用可在运行时被改变。特别用于实现时间和回调方法。

事件(Event)：

用户操作，按键、点击、鼠标移动等，或是一些提示信息，如系统生成的通知。应用程序需要在事件发生时响应时间。

事件在类中声明且生成，且通过使用同一个类或其他类中的委托与实践处理程序关联。

集合(Collection)：

集合类是专门用于数据存储和检索的类。这些类提供了对栈、队列、列表和哈哈希表的支持。大多数集合类实现了相同的接口。

集合类服务于不同的目的，如为元素动态分配内存，基于索引访问列表等。

泛型(Generic)：

允许延迟编写类或方法中的编程元素的数据类型的规范，直到实际在程序中使用的时候。可以通过数据类型的替代参数编写类或方法的规范。当编译器遇到类的构造函数或方法的函数调用时，它会产生代码来处理指定的数据类型。总结：泛型允许编写一个可以与任何数据类型一起工作的类或方法，当编辑器遇到类的结构函数或方法的函数调用时，他会生成代码来处理指定的数据结构。

匿名方法：

提供可一种传递代码作为委托参数的技术。匿名方法是没有名称只有主体的方法。在匿名方法中不需要指定返回类型，同时从方法主体内的 return 语句推断的。

不安全代码：

当一个代码块使用 unsafe 修饰符标记时，C# 允许在函数中使用指针变量。不安全代码或非托管代码指的是使用了指针变量的代码块。

指针变量是值为另一个变量的地址的变量，即，内存位置的直接地址。就像其他变量或常量，必须在使用指针存储其他变量地址之前声明指针。type\* ver-name;

int\* p p是指向整数的指针

int\*\* p p是指向整数的指针的指针

多线程：

线程被定义为程序的执行路径。每个线程都定义了一个独特的控制流。如果程序涉及到复杂和耗时的操作，那么设置不同的线程执行路径往往是有益的，每个线程执行特定的工作。

线程是轻量级进程。一个使用线程的常见实例的实现操作系统中并行编程的实现。使用线程节省了CPU周期的浪费，同时提高了应用程序的效率。

目前为止，我们大多数写的练习程序普遍是一个点县城作为应用程序的运行实例的单一的过程运行。但是这样的子应用程序同时只能执行一个任务。为了同时执行多个任务，它可以被划分为更小的线程。

数据库

表是相关的数据项的集合，它由列和行组成。

SQL 语法

SQL 分为两个部分：数据操作语言 (DML) 和 数据定义语言 (DDL)

对大小写不敏感

通过函数调用访问数据？ADO+PHP

DML：

Select

SELECT - 从数据库表中获取数据

SELECT 列名称 FROM 表名称

SELECT \* FROM 表名称

SELECT DISTINCT 列名称 FROM 表名称

SELECT 列名称 FROM 表名称 WHERE 列 运算符 值

SELECT \* FROM Persons WHERE (FirstName='Thomas' OR FirstName='William') AND LastName='Carter'

SELECT Company, OrderNumber FROM Orders ORDER BY Company DESC, OrderNumber ASC——默认按照升序

SELECT INTO

创建表的备份复件

1. 把所有的列插入新表

SELECT \*

INTO new\_table\_name [IN externaldatabase]

FROM old\_tablename

SELECT \*

INTO Persons\_backup

FROM Persons

SELECT \*

INTO Persons IN 'Backup.mdb'

FROM Persons

1. 只把希望的列插入新表

SELECT column\_name(s)

INTO new\_table\_name [IN externaldatabase]

FROM old\_tablename

SELECT LastName,FirstName

INTO Persons\_backup

FROM Persons

SELECT LastName,Firstname

INTO Persons\_backup

FROM Persons

WHERE City='Beijing'

SELECT Persons.LastName,Orders.OrderNo

INTO Persons\_Order\_Backup

FROM Persons

INNER JOIN Orders

ON Persons.Id\_P=Orders.Id\_P

Top

TOP子句：规定要返回的记录的数目

Mysql：SELECT column\_name(s) FROM table\_name LIMIT number

SELECT TOP 2 \* FROM Persons

SELECT TOP 50 PERCENT \* FROM Persons

运算符：

= 等于

<> 不等于 或者 !=

> 大于

< 小于

>= 大于等于

<= 小于等于

BETWEEN 在某个范围内

BETWEEN 操作符

SELECT column\_name(s)

FROM table\_name

WHERE column\_name

BETWEEN value1 AND value2

SELECT \* FROM Persons

WHERE LastName

BETWEEN 'Adams' AND 'Carter'

**包前不包后（不同数据库有差异）**

like操作符

LIKE 搜索某种模式

SELECT column\_name(s)

FROM table\_name

WHERE column\_name LIKE pattern

SELECT \* FROM Persons WHERE City LIKE 'N%'

"%" 可用于定义**通配符**（模式中缺少的字母）

通配符必须与 LIKE 运算符一起使用



SELECT \* FROM Persons WHERE City LIKE '%g'

SELECT \* FROM Persons WHERE City LIKE '%lon%'

SELECT \* FROM Persons WHERE City NOT LIKE '%lon%'

SELECT \* FROM Persons WHERE FirstName LIKE '\_eorge'

SELECT \* FROM Persons WHERE LastName LIKE 'C\_r\_er'

SELECT \* FROM Persons WHERE City LIKE '[ALN]%'

SELECT \* FROM Persons WHERE City LIKE '[!ALN]%’

IN 操作符

SELECT column\_name(s) FROM table\_name WHERE column\_name IN (value1,value2,...)

SELECT \* FROM Persons WHERE LastName IN ('Adams','Carter')

Update

UPDATE - 更新数据库表中的数据

UPDATE 表名称 SET 列名称 = 新值 WHERE 列名称 = 某值

UPDATE Person SET Address = 'Zhongshan 23', City = 'Nanjing' WHERE LastName = 'Wilson'

Delete

DELETE - 从数据库表中删除数据

DELETE FROM 表名称 WHERE 列名称 = 值

DELETE FROM Person WHERE LastName = 'Wilson'

在不删除表的情况下删除所有的行：

DELETE FROM table\_name

DELETE \* FROM table\_name

INSERT INTO - 向数据库表中插入数据

INSERT INTO 表名称 VALUES (值1, 值2,....)

INSERT INTO table\_name (列1, 列2,...) VALUES (值1, 值2,....)

INSERT INTO Persons (LastName, Address) VALUES ('Wilson', 'Champs-Elysees')

Alias

表：

SELECT column\_name(s)

FROM table\_name

AS alias\_name

SELECT po.OrderID, p.LastName, p.FirstName

FROM Persons AS p, Product\_Orders AS po

WHERE p.LastName='Adams' AND p.FirstName='John'

列：

SELECT column\_name AS alias\_name

FROM table\_name

SELECT LastName AS Family, FirstName AS Name

FROM Persons

Join

SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo

FROM Persons, Orders

WHERE Persons.Id\_P = Orders.Id\_P

SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo

FROM Persons

**INNER** **JOIN** Orders

ON Persons.Id\_P = Orders.Id\_P

ORDER BY Persons.LastName

连接类型：

JOIN/INNER JOIN: 如果表中有至少一个匹配，则返回行

LEFT JOIN: 即使右表中没有匹配，也从左表返回所有的行

RIGHT JOIN: 即使左表中没有匹配，也从右表返回所有的行

FULL JOIN: 只要其中一个表中存在匹配，就返回行

INNER JOIN 关键字

SELECT column\_name(s)

FROM table\_name1

INNER JOIN table\_name2

ON table\_name1.column\_name=table\_name2.column\_name

LEFT JOIN 关键字

SELECT column\_name(s)

FROM table\_name1

LEFT JOIN table\_name2

ON table\_name1.column\_name=table\_name2.column\_name

SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo

FROM Persons

LEFT JOIN Orders

ON Persons.Id\_P=Orders.Id\_P

ORDER BY Persons.LastName

RIGHT JOIN 关键字

SELECT column\_name(s)

FROM table\_name1

RIGHT JOIN table\_name2

ON table\_name1.column\_name=table\_name2.column\_name

SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo

FROM Persons

RIGHT JOIN Orders

ON Persons.Id\_P=Orders.Id\_P

ORDER BY Persons.LastName

FULL JOIN 关键字

SELECT column\_name(s)

FROM table\_name1

FULL JOIN table\_name2

ON table\_name1.column\_name=table\_name2.column\_name

SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo

FROM Persons

FULL JOIN Orders

ON Persons.Id\_P=Orders.Id\_P

ORDER BY Persons.LastName

UNION 操作符

合并两个或多个 SELECT 语句的结果集

UNION

SELECT column\_name(s) FROM table\_name1

UNION

SELECT column\_name(s) FROM table\_name2

UNION 不允许重复

允许重复用 UNION ALL

UNION ALL

SELECT column\_name(s) FROM table\_name1

UNION ALL

SELECT column\_name(s) FROM table\_name2

DDL：

CREATE DATABASE

CREATE DATABASE - 创建新数据库

CREATE DATABASE database\_name

CREATE TABLE

CREATE TABLE - 创建新表

CREATE TABLE 表名称

(

列名称1 数据类型,

列名称2 数据类型,

列名称3 数据类型,

....

)

CREATE TABLE Persons

( Id\_P int,

LastName varchar(255),

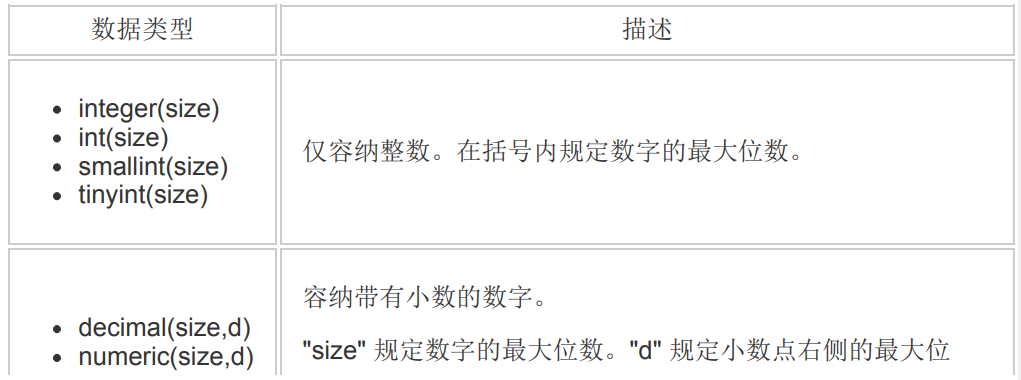
FirstName varchar(255),

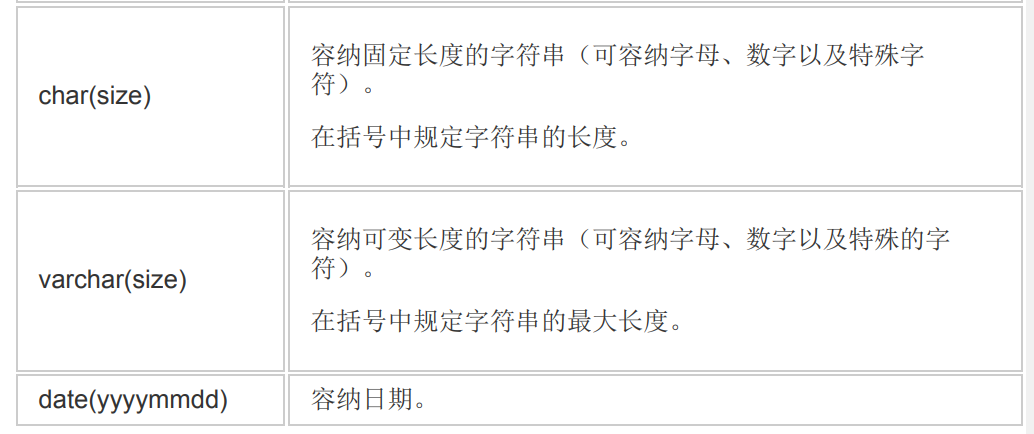
Address varchar(255),

City varchar(255)

)

数据类型





约束

NOT NULL 约束

不接受空值，不向字段添加值，就无法插入新记录或者更新记录。

CREATE TABLE Persons

(

Id\_P int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255)

)

UNIQUE 约束

UNIQUE 和 PRIMARY KEY 约束均为列或列集合提供了唯一性的保证。

PRIMARY KEY 拥有自动定义的 UNIQUE 约束

每个表可以有多个 UNIQUE 约束，但是每个表只能有一个 PRIMARY KEY 约束

1.创建表时：

MySQL:

CREATE TABLE Persons

( Id\_P int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255),

UNIQUE (Id\_P)

)

CREATE TABLE Persons

(

Id\_P int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255),

CONSTRAINT **uc\_PersonID** UNIQUE (Id\_P,LastName)

)

2.对已有的表进行修改

ALTER TABLE Persons

ADD UNIQUE (Id\_P)

ALTER TABLE Persons

ADD CONSTRAINT **uc\_PersonID** UNIQUE (Id\_P,LastName)

3.撤销 UNIQUE 约束：

MySQL:

ALTER TABLE Persons

DROP **INDEX** uc\_PersonID

PRIMARY KEY 约束

唯一的值；非空；每个表必须有且仅有一个

1.创建表时：

CREATE TABLE Persons

(

Id\_P int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255),

PRIMARY KEY (Id\_P)

)

多列：

CREATE TABLE Persons

(

Id\_P int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255),

CONSTRAINT pk\_PersonID PRIMARY KEY (Id\_P,LastName)

)

2.对已有的表进行修改

ALTER TABLE Persons

ADD PRIMARY KEY (Id\_P)

ALTER TABLE Persons

ADD CONSTRAINT pk\_PersonID PRIMARY KEY (Id\_P,LastName)

3.撤销 PRIMARY KEY 约束：

MySQL：

ALTER TABLE Persons

DROP PRIMARY KEY

FOREIGN KEY 约束

外键，防止破坏表间连接，防止非法数据插入外键列（必须是它指向的那个表中的值之一）

1.创建表时：

MySQL:

CREATE TABLE Orders

(

Id\_O int NOT NULL,

OrderNo int NOT NULL,

Id\_P int,

PRIMARY KEY (Id\_O),

FOREIGN KEY (Id\_P) REFERENCES Persons(Id\_P)

)

多列：

CREATE TABLE Orders

(

Id\_O int NOT NULL,

OrderNo int NOT NULL, Id\_P int,

PRIMARY KEY (Id\_O),

CONSTRAINT fk\_PerOrders FOREIGN KEY (Id\_P)

REFERENCES Persons(Id\_P)

)

2.对已有的表进行修改

ALTER TABLE Orders

ADD FOREIGN KEY (Id\_P)

REFERENCES Persons(Id\_P)

3.撤销 FOREIGN KEY 约束：

MySQL:

ALTER TABLE Orders

DROP FOREIGN KEY fk\_PerOrders

CHECK 约束

限制列中的值的范围；单列check，该列只允许特定的值；表check，在特定的列中对值进行限制

1.创建表时：

My SQL:

CREATE TABLE Persons

(

Id\_P int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255),

**CHECK (Id\_P>0)**

)

多列：

CREATE TABLE Persons

(

Id\_P int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255),

CONSTRAINT chk\_Person CHECK (Id\_P>0 AND City='Sandnes')

)

2.对已有的表进行修改

ALTER TABLE Persons

ADD CHECK (Id\_P>0)

多列

ALTER TABLE Persons

ADD CONSTRAINT chk\_Person

CHECK (Id\_P>0 AND City='Sandnes')

3.撤销 CHECK 约束：

MySQL：

ALTER TABLE Persons

DROP CHECK chk\_Person

DEFAULT 约束

向列中插入默认值；若没有规定其他的值，那么会将默认值添加到所有的新记录

1.创建表时：

CREATE TABLE Persons

(

Id\_P int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255) DEFAULT 'Sandnes'

)

GETDATE()：DEFAULT 约束也可以用于插入系统值

CREATE TABLE Orders

(

Id\_O int NOT NULL,

OrderNo int NOT NULL,

Id\_P int,

**OrderDate date DEFAULT GETDATE()**

)

2.对已有的表进行修改

ALTER TABLE Persons

ALTER City SET DEFAULT 'SANDNES'

3.撤销 DEFAULT 约束

ALTER TABLE Persons

ALTER City DROP DEFAULT

DROP

DROP TABLE - 删除表 （表的结构、属性以及索引也会被删除）

DROP TABLE 表名称

TRUNCATE TABLE—**删除表中数据，不删除表本身**

TRUNCATE TABLE 表名称

DROP DATABASE - 删除数据库

DROP DATABASE 数据库名称

INDEX

索引：不读取整个表，查找数据

CREATE INDEX - 创建索引（搜索键）

允许使用重复的值：

CREATE INDEX index\_name

ON table\_name (column\_name)

不能拥有相同的索引值：

CREATE UNIQUE INDEX index\_name

ON table\_name (column\_name)

降序

CREATE INDEX PersonIndex

ON Person (LastName DESC)

多列

CREATE INDEX PersonIndex

ON Person (LastName, FirstName)

DROP INDEX - 删除索引

删除索引：

ALTER TABLE table\_name

DROP INDEX index\_name

ALTER

ALTER DATABASE - 修改数据库

ALTER TABLE - 变更（改变）数据库表

添加列

ALTER TABLE table\_name

ADD column\_name datatype

删除列（有些数据库不支持）

ALTER TABLE table\_name

**DROP COLUMN** column\_name

改变列的数据类型

ALTER TABLE table\_name

**ALTER COLUMN** column\_name datatype

AUTO INCREMENT 字段

在新记录插入表中时生成一个唯一的数字——自动地创建主键字段的值

CREATE TABLE Persons

(

P\_Id int NOT NULL **AUTO\_INCREMENT**,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Address varchar(255),

City varchar(255),

PRIMARY KEY (P\_Id)

)

以其他值开始：

ALTER TABLE Persons AUTO\_INCREMENT=100

VIEW

视图是可视化的表；视图总是显示最近的数据

CREATE VIEW

CREATE VIEW view\_name AS

SELECT column\_name(s)

FROM table\_name

WHERE condition

CREATE VIEW [Products Above Average Price] AS

SELECT ProductName,UnitPrice

FROM Products

WHERE UnitPrice>(SELECT AVG(UnitPrice) FROM Products)

CREATE VIEW [Category Sales For 1997] AS

SELECT DISTINCT CategoryName,Sum(ProductSales) AS CategorySales

FROM [Product Sales for 1997]

GROUP BY CategoryName

查询视图：

SELECT \* FROM [Products Above Average Price]

更新视图

//SQL CREATE OR REPLACE VIEW Syntax

CREATE OR REPLACE VIEW view\_name AS

SELECT column\_name(s)

FROM table\_name

WHERE condition

CREATE VIEW [Current Product List] AS

SELECT ProductID,ProductName,Category

FROM Products

WHERE Discontinued=No

撤销视图

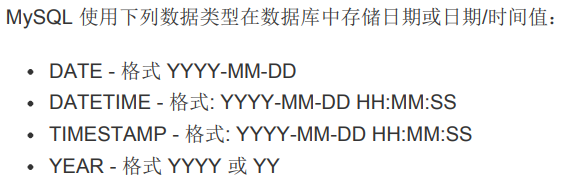
//SQL DROP VIEW Syntax

DROP VIEW view\_name

函数

**Date 函数**





**GROUP BY 语句**

结合合计函数，根据一个或多个列对结果集进行分组

SELECT Customer,OrderDate,**SUM(OrderPrice)**

FROM Orders

**GROUP BY** Customer,OrderDate

**HAVING 子句**

增加 HAVING 子句原因是，WHERE 关键字无法与合计函数一起使用

SELECT column\_name, aggregate\_function(column\_name)

FROM table\_name

WHERE column\_name operator value

GROUP BY column\_name

**HAVING** aggregate\_function(column\_name) operator value

SELECT Customer,SUM(OrderPrice)

FROM Orders

GROUP BY Customer

HAVING SUM(OrderPrice)<2000

MySQL 教程

语言测试：

dynamic 动态类型

int 2147483647

string // 字符串

char // 字符 不为空

decimal // 金钱类型，用于存储金钱的值，在值后加m，否则会被认为是double（精度15-16位），原因，小数点有效精度28到29

占位符 {0}，n1

swap a=a-b b=a+b a=b-a

### 自动类型转换、隐式类型转换

参与运算的操作数和结果类型必须一致，如果不一致，系统自动完成类型转换。（同种类型兼容，目标类大于原类型）

强制类型转换、显示类型转换

Convert 不再仅仅是内存级别的转换，而是考虑数据意义的转换，是一个加工、改造的过程。

类型如果相兼容的两个变量，可以使用自动类型转换或者强制转换类型转换，但如果两个类型的变量不兼容，比如 string 与 int 与 double，这个时候我们可以使用一个叫做 Convert 的转换工厂进行转换。

使用 Convert 进行转换时也需要满足一个条件

break、continue

递归 阶层 result=factorial(num - 1) \* num;

实参

interesting a = 100;

interesting b = 200;

形参

swap

temp = x;

x = y;

y = temp;

按值传递参数：时调用一个方法时，为每一个参数创建一个新的存储位置。所以不会影响到原本的实参。

public void sum (int a, int b) { a += b}

按引用传递参数（关键字：ref int x）：引用参数是一个对变量的内存位置的引用，它不会为这些参数创建一个新的存储位置。所以在发生参数变幻时，原实参也会收到改变。

public void sum (ref int a, ref int b) { a += b}

按输出传递参数（关键字：out int x）：return 语句只从函数中返回一个值，但是，可以使用输出函数来从函数中返回两个值。输出参数会把方法输出的数据赋给自己。

public void sum (out int a, int b) { a = b + 2}

区别 and 注意事项

ref 和 out 传输的都是数据的地址，所以才能对源数据进行修改。

ref型传递变量前，变量必须初始化，否则编译器会报错，而out 型则不需要初始化（但是在方法中必须要赋值）。

ref传递变量，数值可以传入方法中，而out无法将数据传入方法中。

可空类型(Nullable)：在处理数据库和其他包含可能未赋值的元素的数据类型时，将null赋值给数值类型或布尔类型的功能。

单问号与双问号

单问号用于对int，double，bool等无法直接赋值为null的数据类型进行null赋值，意思是这个数据类型是Nullable。

int? a = 3; 等同于 Nullable<int> i = new Nullable<int>(3);

int i; 表示默认值为0、int? i; 表示默认值为null。

双问号则是用于判断一个变量为null的时候返回一个指定的值。

如果第一个操作数的值为null，则运算符返回第二个操作数的值，否则返回第一个操作数的值。

a = b ?? c

b为空，则a=c，b不为空，a=b

String方法代码实践

当使用 static 关键字声明一个类成员为静态时，意味着无论有多少个类对象被创建，只会有一个该静态成员的副本，只有一个该成员的实例。他们的值可以通过直接调用类而不需要创建类的实例来获取。反之，不声明为static，即使和main方法同一个类，也必须要通过实例化。

继承

class 派生类：基类

C#不支持多继承，但是可以通过接口进行操作

多态：一个接口多个功能

动态多态性，函数的响应是在运行时发生的。

静态多态性，函数的响应是在编译时发生的，编译时函数和对象的连接机制。

两种技术实现静态多态：函数重载（在同一范围内对相同函数名有多个定义，可以是参数类型或参数个数当然不同，但不许只有返回值类型不同），

运算符重载 opertor

接口 interface 以 I 字母开头命名 与继承类似 ： 来引用

异常处理

try（引起异常的语句）、catch（错误处理代码）、finally（无论是否异常都会执行的代码）、throw（当问题出现，程序抛出一个异常）

System.IO.IOException 处理I/O错误

System.IndexOutOfRangException 处理当方法指向超出范围的数组索引时生成的错误

System.ArrayTypeMsimatchException 处理当数组类型不匹配时生成的错误

System.NullReferenceException 处理当依从一个空对象时生成的错误

System.DivideByZeroException 处理当除以零时生成的错误

System.InvalidCastException 处理在类型转换期间生成的错误

System.OutOfMemoryException 处理空闲内存不足生成的错误

System.StackOverflowException 处理栈溢出生成的错误

ApplicationException 自定义异常处理

文件的输出与输入

一个文件是一个存储在磁盘中带有指定名称和目录路径的数据集合。当打开文件进行读写时，它会变成一个流。

流是通过通讯路径传输的字节序列。

System.IO 命名空间中常用的非抽象类

BinaryReader 从二进制流读取原始数据

BinaryWriter 以二进制格式写入原始数据

BufferedStream 字节流的临时存储

Directory 有助于操作目录结构

DirectoryInfo 用于对目录执行操作

DriveInfo 提供驱动器的信息

File 有助于处理文件

FileInfo 用于对文件执行操作

FileStream 用于文件中任何位置的读写

MemoryStream 用于随机访问存储在内存中的数据流

Path 对路径信息执行操作

StreamReader 用于从字节流中读取字符

StreamWriter 用于向一个流中写入字符

StringReader 用于读取字符串缓冲区

StringWriter 用于写入字符串缓冲区

FileStream类有助于文件的读写于关闭

需要创建一个FileStream对象来创建一个新的文件，或打开一个已有的文件。

FileStream <Object\_name> = new FileStream(<file\_name>,<FileMode Enumerator>,<FileAccess Enumerator>,<FileShare Enumerator>);

FileMode：枚举定义了各种打开文件的方法

FileAccess：枚举的成员有Read、ReadWrite、Write

FileShare：枚举定义了操作文件进程的方法

前端

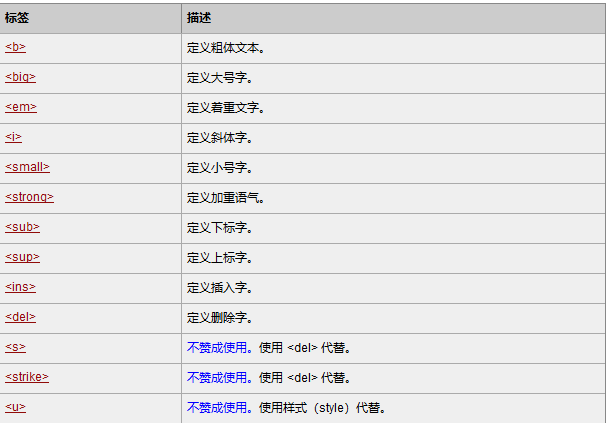
HTML

* 一种**标记语言**
* HTML 文档 = 网页
* 对大小写不敏感
* HTML 代码中的所有连续的空行（换行）也被显示为一个空格

标签

* <html> 与 </html> 之间的文本描述网页
* <body> 与 </body> 之间的文本是可见的页面内容
* <h1> 与 </h1> 之间的文本被显示为标题
* <p> 与 </p> 之间的文本被显示为段落
* <br /> 标签定义换行
* .htm 也可以使用 .html 文件后缀
* <hr /> 标签在 HTML 页面中创建水平线
* 注释：<!--This is a comment-->
* <a> </a>用来创建锚

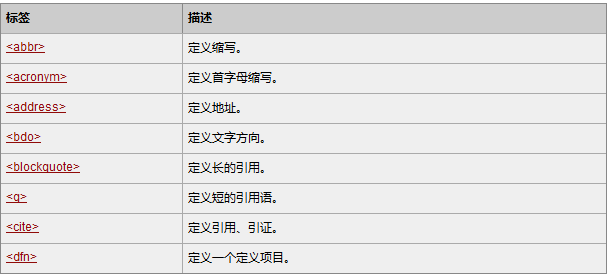
**文本格式化标签**



**“计算机输出”标签**



**引用、引用和术语定义**



**表格标签**

****

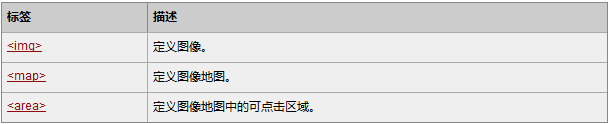
**列表标签**

****

**表单标签**

****

**图像标签**



HTML 链接

<a href="http://www.w3school.com.cn">This is a link</a>

HTML 图像

<img src="w3school.jpg" width="104" height="142" />

HTML 元素

HTML 元素指的是从开始标签（start tag）到结束标签（end tag）的所有代码。

HTML 属性

* 总是以名称/值对的形式出现，比如：name="value"。
* 属性总是在 HTML 元素的开始标签中规定。

实例

**定义水平线**

<html>

<body>

<p>hr 标签定义水平线：</p>

<hr />

<p>这是段落。</p>

<hr />

<p>这是段落。</p>

<hr />

<p>这是段落。</p>

</body>

</html>

**同段换行**

<p>This is<br />a para<br />graph with line breaks</p>

**格式化**

<html>

<body>

<b>This text is bold</b>

<br>

<strong>This text is strong</strong>

<br>

<big>This text is big</big>

<br>

<em>This text is emphasized</em>

<br>

<i>This text is italic</i>

<br>

<small>This text is small</small>

<br>

This text contains

<sub>subscript</sub>

<br>

This text contains

<sup>superscript</sup>

</body>

</html>

**预格式文本（保留空格和换行）**

<html>

<body>

<pre>

这是

预格式文本。

它保留了 空格

和换行。

</pre>

<p>pre 标签很适合显示计算机代码：</p>

<pre>

for i = 1 to 10

print i

next i

</pre>

</body>

</html>

**地址**

<html>

<body>

<p>唐老鸭的地址：</p>

<address>

Donald Duck<br />

BOX 555<br />

Disneyland<br />

USA<br />

</address>

<br />

<p>W3School 的地址：</p>

<address>

<a href="mailto:service@w3school.com.cn">用户服务信箱</a><br />

上海赢科投资有限公司<br />

金桥开发区 789 号<br />

</address>

</body>

</html>

**缩写和首字母缩写**

<html>

<body>

<abbr title="United Nations">UN</abbr>

<br>

<acronym title="World Wide Web">WWW</acronym>

<p>在某些浏览器中，当您把鼠标移至缩略词语上时，title 可用于展示表达的完整版本。</p>

<p>仅对于 IE 5 中的 acronym 元素有效。</p>

<p>对于 Netscape 6.2 中的 abbr 和 acronym 元素都有效。</p>

</body>

</html>

**逆转文字方向**

<html>

<body>

<p>

如果您的浏览器支持 bi-directional override (bdo)，下一行会从右向左输出 (rtl)；

</p>

<bdo dir="rtl">

Here is some Hebrew text

</bdo>

</body>

</html>

**删除和下划线**

<html>

<body>

<p>

一打有

<del>二十</del>

<ins>十二</ins>

件。

</p>

<p>

大多数浏览器会改写为删除文本和下划线文本。

</p>

<p>

一些老式的浏览器会把删除文本和下划线文本显示为普通文本。

</p>

</body>

</html>

**样式**

<html>

<body style="background-color:PowderBlue;">

<h1>Look! Styles and colors</h1>

<p style="font-family:verdana;color:red">

This text is in Verdana and red</p>

<p style="font-family:times;color:green">

This text is in Times and green</p>

<p style="font-size:30px">This text is 30 pixels high</p>

</body>

</html>

<p style="font-family:arial;color:red;font-size:20px;">A paragraph.</p>

对齐方式：

<h1 style="text-align:center">This is a heading</h1>

font-family

font-size

color

text-align

链接

**创建超级链接**

<html>

<body>

<p>

<a href="../index.html" tppabs="http://www.w3school.com.cn/index.html">本文本</a> 是一个指向本网站中的一个页面的链接。</p>

<p><a href="javascript:if(confirm('http://www.microsoft.com/ \n\nThis file was not retrieved by Teleport Pro, because it is addressed on a domain or path outside the boundaries set for its Starting Address. \n\nDo you want to open it from the server?'))window.location='http://www.microsoft.com/'" tppabs="http://www.microsoft.com/">本文本</a> 是一个指向万维网上的页面的链接。</p>

</body>

</html>

**图像作为链接**

<html>

<body>

<p>

您也可以使用图像来作链接：

<a href="../example/html/lastpage.html" tppabs="http://www.w3school.com.cn/example/html/lastpage.html">

<img border="0" src="../i/eg\_buttonnext.gif" tppabs="http://www.w3school.com.cn/i/eg\_buttonnext.gif" />

</a>

</p>

</body>

</html>

**新的浏览窗口打开链接**

<html>

<body>

<a href="../example/html/lastpage.html" tppabs="http://www.w3school.com.cn/example/html/lastpage.html" **target="\_blank"**>最后一页</a>

<p>

如果把链接的 target 属性设置为 "\_blank"，该链接会在新窗口中打开。

</p>

</body>

</html>

**[链接到同一个页面的不同位置](mk:@MSITStore:D:\\Download\\学习资源\\js文档\\完整版W3CSchool线下教程.chm::/www.w3school.com.cn/html/../tiy/t.asp-f=html_link_locations.htm)**

<html>

<body>

<p>

<a href="#C4">查看 Chapter 4。</a>

</p>

<h2>Chapter 1</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 2</h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 3</h2>

<p>This chapter explains ba bla bla</p>

<h2><a name="C4">Chapter 4</a></h2>

<p>This chapter explains ba bla bla</p>

<h2>Chapter 5</h2>

<p>This chapter explains ba bla bla</p>

</body>

</html>

**[跳出框架](mk:@MSITStore:D:\\Download\\学习资源\\js文档\\完整版W3CSchool线下教程.chm::/www.w3school.com.cn/html/../tiy/t.asp-f=html_frame_getfree.htm)**

<html>

<body>

<p>被锁在框架中了吗？</p>

<a href="../index.html" tppabs="http://www.w3school.com.cn/index.html"

target="\_top">请点击这里！</a>

</body>

</html>

**[创建电子邮件链接](mk:@MSITStore:D:\\Download\\学习资源\\js文档\\完整版W3CSchool线下教程.chm::/www.w3school.com.cn/html/../tiy/t.asp-f=html_mailto.htm)**

<html>

<body>

<p>

这是邮件链接：

<a href="mailto:someone@microsoft.com?cc=someoneelse@microsoft.com&bcc=andsomeoneelse2@microsoft.com&subject=Summer%20Party&body=You%20are%20invited%20to%20a%20big%20summer%20party!">发送邮件！</a>

</p>

<b>注意：</b>应该使用 %20 来替换单词之间的空格，这样浏览器就可以正确地显示文本了。

</p>

</body>

</html>

表格

<html>

<body>

<p>每个表格由 table 标签开始。</p>

<p>每个表格行由 tr 标签开始。</p>

<p>每个表格数据由 td 标签开始。</p>

<h4>一列：</h4>

<table border="1">

<caption>我的标题</caption>

<tr>

<td>100</td>

</tr>

</table>

<h4>一行三列：</h4>

<table border="1">

<tr>

<td>100</td>

<td>200</td>

<td>300</td>

</tr>

</table>

<h4>两行三列：</h4>

<table border="1">

<tr>

<th>Heading</th>

<th>Another Heading</th>

<th>Heading 3</th>

</tr>

<tr>

<td>400</td>

<td>500</td>

<td>600</td>

</tr>

</table>

</body>

</html>

**在空单元格中添加一个空格占位符，就可以将边框显示出来**

<td>&nbsp**;**</td>

**表头**

<html>

<body>

<h4>表头：</h4>

<table border="1">

<tr>

<th>姓名</th>

<th>电话</th>

<th>电话</th>

</tr>

<tr>

<td>Bill Gates</td>

<td>555 77 854</td>

<td>555 77 855</td>

</tr>

</table>

<h4>垂直的表头：</h4>

<table border="1">

<tr>

<th>姓名</th>

<td>Bill Gates</td>

</tr>

<tr>

<th>电话</th>

<td>555 77 854</td>

</tr>

<tr>

<th>电话</th>

<td>555 77 855</td>

</tr>

</table>

</body>

</html>

**跨行或跨列的表格单元格**

<html>

<body>

<h4>横跨两列的单元格：</h4>

<table border="1">

<tr>

<th>姓名</th>

<th colspan="2">电话</th>

</tr>

<tr>

<td>Bill Gates</td>

<td>555 77 854</td>

<td>555 77 855</td>

</tr>

</table>

<h4>横跨两行的单元格：</h4>

<table border="1">

<tr>

<th>姓名</th>

<td>Bill Gates</td>

</tr>

<tr>

<th rowspan="2">电话</th>

<td>555 77 854</td>

</tr>

<tr>

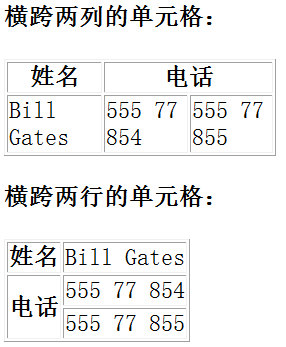
<td>555 77 855</td>

</tr>

</table>

</body>

</html>

****

**表格内的标签**

<html>

<body>

<table border="1">

<tr>

<td>

<p>这是一个段落。</p>

<p>这是另一个段落。</p>

</td>

<td>这个单元包含一个表格：

<table border="1">

<tr>

<td>A</td>

<td>B</td>

</tr>

<tr>

<td>C</td>

<td>D</td>

</tr>

</table>

</td>

</tr>

<tr>

<td>这个单元包含一个列表：

<ul>

<li>苹果</li>

<li>香蕉</li>

<li>菠萝</li>

</ul>

</td>

<td>HELLO</td>

</tr>

</table>

</body>

</html>

**单元格边距**

<table border="1"

cellpadding="10">

**单元格间距：**

<table border="1"

cellspacing="10">

**在表格单元中排列内容**

<html>

<body>

<table width="400" border="1">

<tr>

<th align="left">消费项目....</th>

<th align="right">一月</th>

<th align="right">二月</th>

</tr>

<tr>

<td align="left">衣服</td>

<td align="right">$241.10</td>

<td align="right">$50.20</td>

</tr>

<tr>

<td align="left">化妆品</td>

<td align="right">$30.00</td>

<td align="right">$44.45</td>

</tr>

<tr>

<td align="left">食物</td>

<td align="right">$730.40</td>

<td align="right">$650.00</td>

</tr>

<tr>

<th align="left">总计</th>

<th align="right">$1001.50</th>

<th align="right">$744.65</th>

</tr>

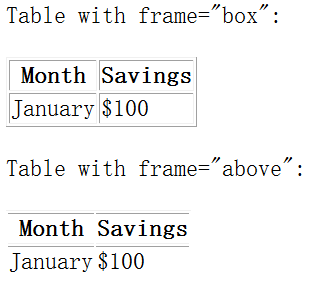
</table>

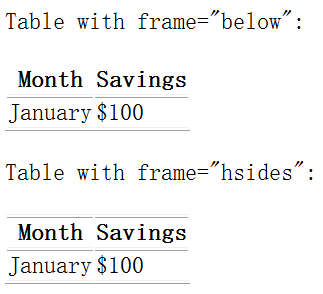
</body>

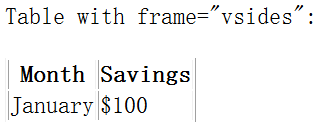
</html>

**框架属性：**

**<table frame="box">**

****

****

****

列表

**无序列表**

<ul>

<li>Coffee</li>

<li>Milk</li>

</ul>

<ul type="disc">

<ul type="circle">

<ul type="square">

有序列表

<ol>

<li>Coffee</li>

<li>Milk</li>

</ol>

默认：数字列表

<ol type="A">

<ol type="a">

<ol type="I">

<ol type="i">

自定义列表

<dl>

<dt>Coffee</dt>

<dd>Black hot drink</dd>

<dt>Milk</dt>

<dd>White cold drink</dd>

</dl>

嵌套列表

<html>

<body>

<h4>一个嵌套列表：</h4>

<ul>

<li>咖啡</li>

<li>茶

<ul>

<li>红茶</li>

<li>绿茶

<ul>

<li>中国茶</li>

<li>非洲茶</li>

</ul>

</li>

</ul>

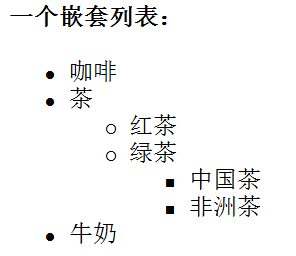
</li>

<li>牛奶</li>

</ul>

</body>

</html>

****

表单

文本域

<html>

<body>

<form>

名：

<input type="text" name="firstname">

<br>

姓：

<input type="text" name="lastname">

</form>

</body>

</html>

多行文本输入

<textarea rows="10" cols="30">

The cat was playing in the garden.

密码域

<html>

<body>

<form>

用户：

<input type="text" name="user"/>

<br>

密码：

<input type="password" name="password"/>

</form>

<p>

请注意，当您在密码域中键入字符时，浏览器将使用项目符号来代替这些字符。

</p>

</body>

</html>

单选按钮（Radio Buttons）

当用户点击一个单选按钮时，该按钮会变为选中状态，其他所有按钮会变为非选中状态。

<form>

**<input type="radio" name="sex" value="male" /> Male**

<br />

<input type="radio" name="sex" value="female" /> Female

</form>

复选框（Checkboxes）

<form>

**<input type="checkbox" name="bike" />**

**I have a bike**

<br />

<input type="checkbox" name="car" />

I have a car

</form>

动作属性和确认按钮

<form name="input" action="html\_form\_action.asp" method="get">

Username:

<input type="text" name="user" />

<input type="submit" value="Submit" />

</form>

<html>

<body>

<form action="http://www.w3school.com.cn/example/html/form\_action.asp" method="get">

<p>First name: <input type="text" name="fname" /></p>

<p>Last name: <input type="text" name="lname" /></p>

<input type="submit" value="Submit" />

</form>

<p>请单击确认按钮，输入会发送到服务器上名为 "form\_action.asp" 的页面。</p>

</body>

</html>

下拉列表：

<html>

<body>

<form>

<select name="cars">

<option value="volvo">Volvo</option>

<option value="saab">Saab</option>

<option value="fiat" selected="selected">Fiat</option>

<option value="audi">Audi</option>

</select>

</form>

</body>

</html>

创建按钮：

<html>

<body>

<form>

<input type="button" value="Hello world!">

</form>

</body>

</html>

带标题的框

<html>

<body>

<fieldset>

<legend>健康信息：</legend>

<form>

<label>身高：<input type="text" /></label>

<label>体重：<input type="text" /></label>

</form>

</fieldset>

<p>如果表单周围没有边框，说明您的浏览器太老了。</p>

</body>

</html>

从表单发送邮件

<html>

<body>

<form action="http:///someone@w3school.com.cn" method="post" enctype="text/plain">

<h3>这个表单会把电子邮件发送到 W3School。</h3>

姓名：<br>

<input type="text" name="name" value="yourname" size="20">

<br>

电邮：<br>

<input type="text" name="mail" value="yourmail" size="20">

<br>

内容：<br>

<input type="text" name="comment" value="yourcomment" size="40">

<br><br>

<input type="submit" value="发送">

<input type="reset" value="重置">

</form>

</body>

</html>

图像

插入图像

<html>

<body>

<p>

一幅图像：

<img src="../i/eg\_mouse.jpg"

tppabs="http://www.w3school.com.cn/i/eg\_mouse.jpg"

width="128" height="128">

</p>

<p>

一幅动画图像：

<img src="../i/eg\_cute.gif" tppabs="http://www.w3school.com.cn/i/eg\_cute.gif"

width="50" height="50">

</p>

<p>请注意，插入动画图像的语法与插入普通图像的语法没有区别。</p>

</body>

</html>

替换文本属性（Alt）

在浏览器无法载入图像时，替换文本属性告诉读者她们失去的信息

<img src="boat.gif" alt="Big Boat">

图像排列

<html>

<body>

<p>

图像

<img src="../i/eg\_cute.gif" tppabs="http://www.w3school.com.cn/i/eg\_cute.gif" align="bottom">

在文本中

</p>

<p>

图像

<img src ="../i/eg\_cute.gif" tppabs="http://www.w3school.com.cn/i/eg\_cute.gif" align="middle">

在文本中

</p>

<p>

图像

<img src ="../i/eg\_cute.gif" tppabs="http://www.w3school.com.cn/i/eg\_cute.gif" align="top">

在文本中

</p>

<p>请注意，bottom 对齐方式是默认的对齐方式。</p>

<p>

图像

<img src ="../i/eg\_cute.gif" tppabs="http://www.w3school.com.cn/i/eg\_cute.gif">

在文本中

</p>

<p>

<img src ="../i/eg\_cute.gif" tppabs="http://www.w3school.com.cn/i/eg\_cute.gif">

图像在文本前面

</p>

<p>

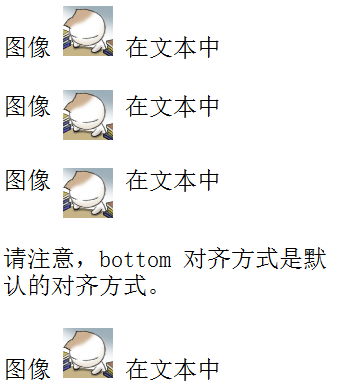
图像在文本后面

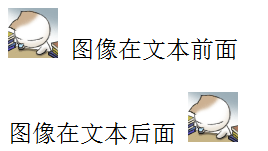
<img src ="../i/eg\_cute.gif" tppabs="http://www.w3school.com.cn/i/eg\_cute.gif">

</p>

</body>

</html>





align=top/middle/bottom;left/right

图像上的文本（Alt）

无法显示图像，将显示 "alt" 属性中的文本

把鼠标指针移动到图像上，大多数浏览器会显示 "alt" 文本

<img src="../i/eg\_goleft.gif" tppabs="http://www.w3school.com.cn/i/eg\_goleft.gif" alt="向左转" />

图片作为链接

（图片上有链接）

<html>

<body>

<p>

您也可以把图像作为链接来使用：

<a href="../example/html/lastpage.html" tppabs="http://www.w3school.com.cn/example/html/lastpage.html">

<img border="0" src="../i/eg\_buttonnext.gif" tppabs="http://www.w3school.com.cn/i/eg\_buttonnext.gif" />

</a>

</p>

</body>

</html>

图片映射

<html>

<body>

<p>请点击图像上的星球，把它们放大。</p>

<img

src="../i/eg\_planets.jpg" tppabs="http://www.w3school.com.cn/i/eg\_planets.jpg"

border="0" usemap="#planetmap"

alt="Planets" />

<map name="planetmap" id="planetmap">

<area

shape="circle"

coords="180,139,14"

href ="../example/html/venus.html" tppabs="http://www.w3school.com.cn/example/html/venus.html"

target ="\_blank"

alt="Venus" />

<area

shape="circle"

coords="129,161,10"

href ="../example/html/mercur.html" tppabs="http://www.w3school.com.cn/example/html/mercur.html"

target ="\_blank"

alt="Mercury" />

<area

shape="rect"

coords="0,0,110,260"

href ="../example/html/sun.html" tppabs="http://www.w3school.com.cn/example/html/sun.html"

target ="\_blank"

alt="Sun" />

</map>

<p><b>注释：</b>img 元素中的 "usemap" 属性引用 map 元素中的 "id" 或 "name" 属性（根据浏览器），所以我们同时向 map 元素添加了 "id" 和 "name" 属性。</p>

</body>

</html>

背景

<body bgcolor="#000000">

<body bgcolor="rgb(0,0,0)">

<body bgcolor="black">

相对地址：<body background="clouds.gif">

绝对地址：<body background="http://www.w3school.com.cn/clouds.gif">

已被废弃，早点说呀！真的会谢，用css

* 背景图像是否增加了页面的加载时间。小贴士：图像文件不应超过 10k。
* 背景图像是否与页面中的其他图象搭配良好。
* 背景图像是否与页面中的文字颜色搭配良好。
* 图像在页面中平铺后，看上去还可以吗？
* 对文字的注意力被背景图像喧宾夺主了吗？

CSS

* 层叠样式表 (Cascading Style Sheets)
* 解决内容与表现分离的问题
* 当使用 **RGB 百分比**时，即使当值为 0 时也要写百分比符号。但是在其他的情况下就不需要这么做了
* CSS 对大小写不敏感。不过存在一个例外：如果涉及到与 HTML 文档一起工作的话，class 和 id 名称对大小写是敏感的。

样式优先级：

4 拥有最高的优先权

1. 浏览器缺省设置
2. 外部样式表
3. 内部样式表（位于 <head> 标签内部）
4. 内联样式（在 HTML 元素内部）

CSS=选择器+属性+值

selector {property: value}

格式：

p {

text-align: center;

color: black;

font-family: arial;

}

组与继承

body {

font-family: Verdana, sans-serif;

}

td, ul, ol, ul, li, dl, dt, dd {

font-family: Verdana, sans-serif;

}

p {

font-family: Times, "Times New Roman", serif;

}

派生选择器：

根据文档的上下文关系来确定某个标签的样式

strong {

color: red;

}

h2 {

color: red;

}

h2 strong {

color: blue;

}

下面是它施加影响的 HTML：

<p>The strongly emphasized word in this paragraph is<strong>red</strong>.</p>

<h2>This subhead is also red.</h2>

<h2>The strongly emphasized word in this subhead is<strong>blue</strong>.</h2>

id 选择器

标有特定 id 的 HTML 元素指定特定的样式

以 "#" 来定义

id 属性只能在每个 HTML 文档中出现一次

建立派生选择器

#sidebar p {

font-style: italic;

text-align: right;

margin-top: 0.5em;

}

#sidebar h2 {

font-size: 1em;

font-weight: normal;

font-style: italic;

margin: 0;

line-height: 1.5;

text-align: right;

}

单独使用：

#sidebar {

border: 1px dotted #000;

padding: 10px;

}

类选择器

以一个点号显示

类名的第一个字符不能使用数字

用作派生选择器

.center {text-align: center}

<h1 class="center">

This heading will be center-aligned

</h1>

<p class="center">

This paragraph will also be center-aligned.

</p>

用作派生选择器

类名为 fancy 的表格单元将是带有灰色背景的橙色

.fancy td {

color: #f60;

background: #666;

}

td.fancy {

color: #f60;

background: #666;

}

<td class="fancy">

创建CSS

插入样式表的方法有三种：

* 外部样式表
* 内部样式表
* 内联样式

外部样式表

样式需要应用于很多页面时

可以通过改变一个文件来改变整个站点的外观

每个页面使用 <link> 标签链接到样式表。<link> 标签在（文档的）头部

文件不能包含任何的 html 标签

以 .css 扩展名进行保存

不要在属性值与单位之间留有空格

页面链接到样式表

<head>

<**link** rel="stylesheet" type="text/css" href="**mystyle.css**" />

</head>

样式表文件

hr {color: sienna;}

p {margin-left: 20px;}

body {background-image: url("images/back40.gif");}

内部样式表

单个文档需要特殊的样式时

<style> 标签在文档头部定义内部样式表

<head>

<style type="text/css">

hr {color: sienna;}

p {margin-left: 20px;}

body {background-image: url("images/back40.gif");}

</style>

</head>

内联样式

当样式仅需要在一个元素上应用一次时

在相关的标签内使用样式（style）属性

<p style="color: sienna; margin-left: 20px">

This is a paragraph

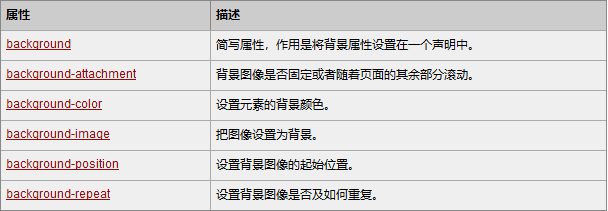
</p>

多重样式：

继承于外部样式表，但如果内部样式表也有，则会被内部样式表中的规则取代。

背景

CSS背景属性



纯色背景

<html>

<head>

<style type="text/css">

body {background-color: yellow}

h1 {background-color: #00ff00}

h2 {background-color: transparent}

p {background-color: rgb(250,0,255)}

p.no2 {background-color: gray; padding: 20px;}

</style>

</head>

<body>

<h1>这是标题 1</h1>

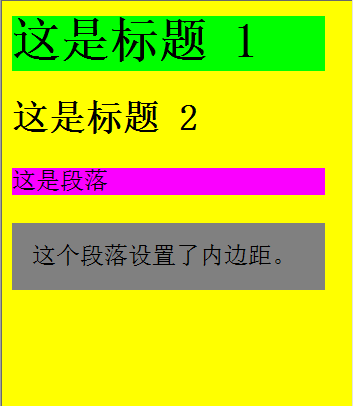
<h2>这是标题 2</h2>

<p>这是段落</p>

<p class="no2">这个段落设置了内边距。</p>

</body>

</html>



背景图像

**background-image 也不能继承。事实上，所有背景属性都不能继承**

<html>

<head>

<style type="text/css">

body {background-image:url("../i/eg\_bg\_04.gif"/\*tpa=http://www.w3school.com.cn/i/eg\_bg\_04.gif\*/);}

p.flower {background-image: url("../i/eg\_bg\_03.gif"/\*tpa=http://www.w3school.com.cn/i/eg\_bg\_03.gif\*/); padding: 20px;}

a.radio {background-image: url("../i/eg\_bg\_07.gif"/\*tpa=http://www.w3school.com.cn/i/eg\_bg\_07.gif\*/); padding: 20px;}

</style>

</head>

<body>

<p class="flower">我是一个有花纹背景的段落。<a href="#" class="radio">我是一个有放射性背景的链接。</a></p>

<p><b>注释：</b>为了清晰地显示出段落和链接的背景图像，我们为它们设置了少许内边距。</p>

</body>

</html>

背景重复（图像平铺）

属性值 repeat 导致图像在水平垂直方向上都平铺

repeat-x 和 repeat-y 分别导致图像只在水平或垂直方向上重复，no-repeat 则不允许图像在任何方向上平铺

body

{

background-image: url(/i/eg\_bg\_03.gif);

background-repeat: repeat-y;

}

背景定位

body

{

background-image:url('/i/eg\_bg\_03.gif');

background-repeat:no-repeat;

background-position:center;

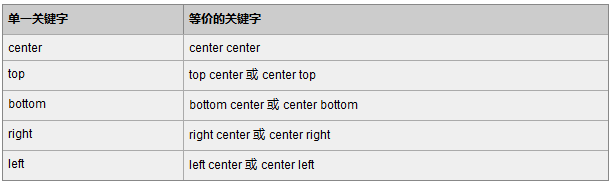
}

top、bottom、left、right 和 center

top right 使图像放置在元素内边距区的右上角

保证不超过两个关键字 - 一个对应水平方向，另一个对象垂直方向。

如果只出现一个关键字，则认为另一个关键字是 center



百分数值

水平方向 2/3、垂直方向 1/3 处： background-position:66% 33%;

只提供一个百分数值，所提供的这个值将用作水平值，垂直值将假设为 50%

长度值，如 100px 或 5cm

图像的左上角将在元素内边距区左上角向右 50 像素、向下 100 像素的位置

background-position:50px 100px;

背景关联

文档向下滚动，图像相对于可视区是固定的

body

{

background-image:url(/i/eg\_bg\_02.gif);

background-repeat:no-repeat;

background-attachment:fixed

}

文本

文本属性



文本缩进

块级元素：text-indent

行内元素的第一行：左内边距或外边距

“悬挂缩进”：text-indent 设置为负值，为了避免出现首行的某些文本可能会超出浏览器窗口的左边界，建议针对负缩进再设置一个外边距或一些内边距

p {text-indent: -5em; padding-left: 5em;}

使用百分比值：百分数要相对于缩进元素父元素的宽度

div {width: 500px;}

p {text-indent: 20%;}

<div>

<p>this is a paragragh</p>

</div>

缩进值是父元素的 20%，即 100 个像素：

继承：

text-indent 属性可以继承

div#outer {width: 500px;}

div#inner {text-indent: 10%;}

p {width: 200px;}

<div id="outer">

<div id="inner">some text. some text. some text.

<p>this is a paragragh.</p>

</div>

</div>

标记中的段落也会缩进 50 像素，这是因为这个段落继承了 id 为 inner 的 div 元素的缩进值

水平对齐

<CENTER> 不仅影响文本，还会把整个元素居中。text-align 不会控制元素的对齐，而只影响内部内容。元素本身不会从一段移到另一端，只是其中的文本受影响。

将块级元素或表元素居中，要通过在这些元素上适当地设置左、右外边距来实现。

两端对齐：justify

字间隔

word-spacing

<html>

<head>

<style type="text/css">

p.spread {word-spacing: 30px;}

p.tight {word-spacing: -0.5em;}

</style>

</head>

<body>

<p class="spread">This is some text. This is some text.</p>

<p class="tight">This is some text. This is some text.</p>

</body>

</html>

字母间隔

letter-spacing

字符转换

text-transform

* none
* uppercase 全大写
* lowercase 全小写
* capitalize 首字母大写

文本装饰

text-decoration：

* none 关闭原本应用到一个元素上的所有装饰
* underline 加下划线
* overline 顶端画一个上划线
* line-through 在文本中间画一个贯穿线
* blink 让文本闪烁

链接默认地会有下划线。如果您希望去掉超链接的下划线

a {text-decoration: none;}

text-decoration 值会替换而不是累积起来

h2.stricken {text-decoration: line-through;}

h2 {text-decoration: underline overline;}

所有 class 为 stricken 的 h2 元素都只有一个贯穿线装饰，而没有下划线和上划线

处理空白符

空格、换行和 tab 字符

默认（normal）：把所有空白符合并为一个空格（换行字符（回车）会转换为空格，一行中多个空格的序列也会转换为一个空格。）

white-space 属性的值为 pre，浏览器将会注意额外的空格，甚至回车

p {white-space: pre;}



文本方向

对于行内元素，只有当 [unicode-bidi 属性](mk:@MSITStore:D:\\Download\\学习资源\\js文档\\完整版W3CSchool线下教程.chm::/www.w3school.com.cn/css/pr_unicode-bidi.asp.htm" \o "CSS unicode-bidi 属性)设置为 embed 或 bidi-override 时才会应用 direction 属性。

direction 属性有两个值：ltr 和 rtl。大多数情况下，默认值是 ltr，显示从左到右的文本。如果显示从右到左的文本，应使用值 rtl

字体

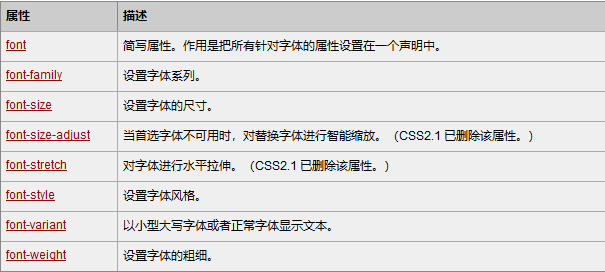
继承性

在所有 font-family 规则中都提供一个通用字体系列

h1 {font-family: Georgia, serif;}

只有当一个字体名中有一个或多个空格（比如 New York），或者如果字体名包括 # 或 $ 之类的符号，才需要在 font-family 声明中加引号

字体属性



所有字体属性写在一个声明里：

p.ex2

{

font:italic bold 12px/30px arial,sans-serif;

}

列表

修改用于列表项的标志类型：

ul {list-style-type : square}

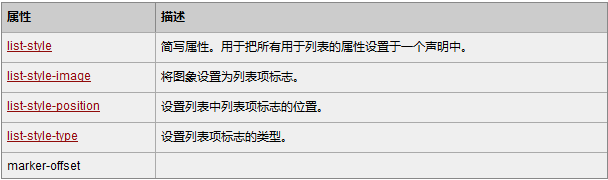
对各标志使用一个图像

ul li {list-style-image : url(xxx.gif)}

简写列表样式

li {list-style : url(example.gif) square inside}

列表属性



无序列表

<html>

<head>

<style type="text/css">

ul.disc {list-style-type: disc}

ul.circle {list-style-type: circle}

ul.square {list-style-type: square}

ul.none {list-style-type: none}

</style>

</head>

<body>

<ul class="disc">

<li>咖啡</li>

<li>茶</li>

<li>可口可乐</li>

</ul>

……

有序列表

<html>

<head>

<style type="text/css">

ol.decimal {list-style-type: decimal}

ol.lroman {list-style-type: lower-roman}

ol.uroman {list-style-type: upper-roman}

ol.lalpha {list-style-type: lower-alpha}

ol.ualpha {list-style-type: upper-alpha}

</style>

</head>

<body>

<ol class="decimal">

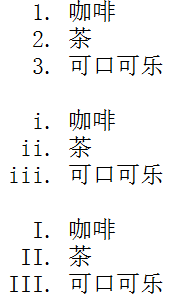
<li>咖啡</li>

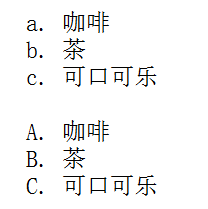
<li>茶</li>

<li>可口可乐</li>

</ol>

…………





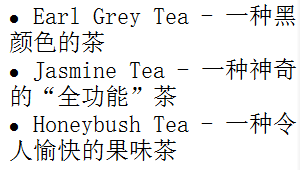
标记是否缩进

ul.inside

{

list-style-position: inside

}

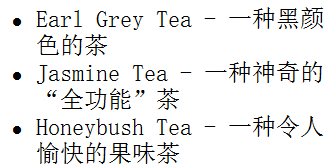


ul.outside

{

list-style-position: outside

}



表格

表格属性



轮廓/边框

p

{

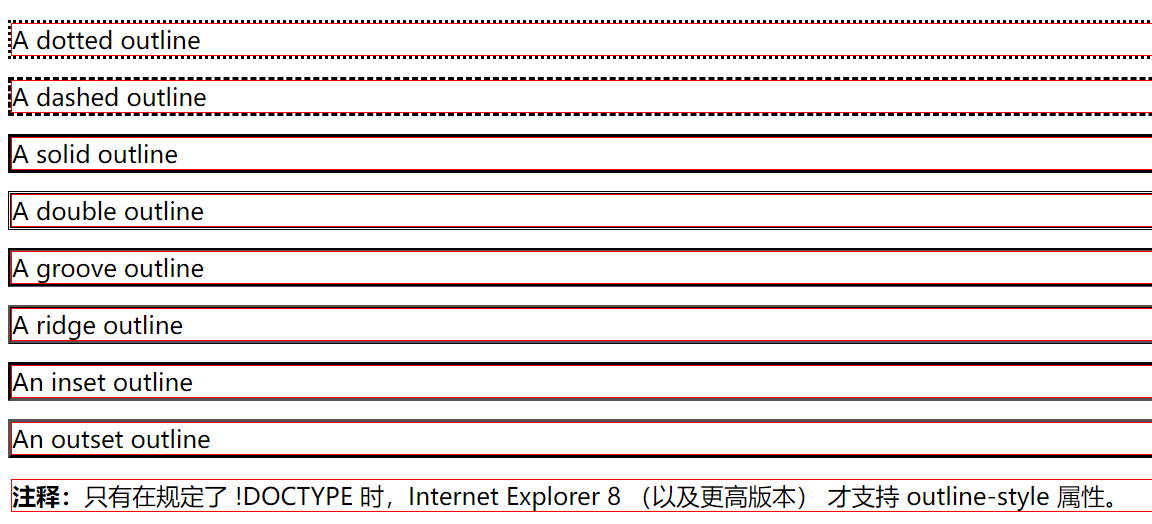
border:red solid thin;

outline:#00ff00 dotted thick;

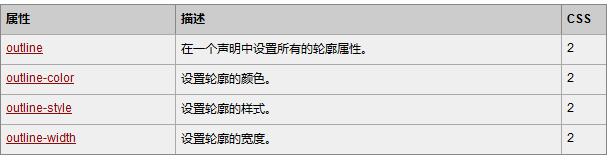
}

只有在规定了 !DOCTYPE 时，Internet Explorer 8 （以及更高版本） 才支持 outline 属性。

outline-style:

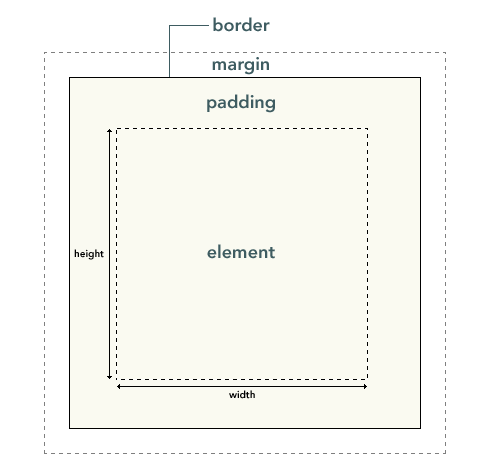


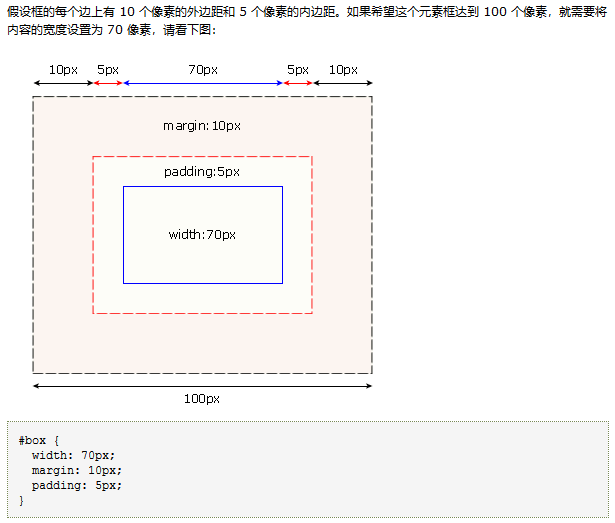
边框属性：



框模型

元素框处理元素内容、[内边距](mk:@MSITStore:D:\\Download\\学习资源\\js文档\\完整版W3CSchool线下教程.chm::/www.w3school.com.cn/css/css_padding.asp.htm" \o "CSS 内边距)、[边框](mk:@MSITStore:D:\\Download\\学习资源\\js文档\\完整版W3CSchool线下教程.chm::/www.w3school.com.cn/css/css_border.asp.htm" \o "CSS 边框) 和 [外边距](mk:@MSITStore:D:\\Download\\学习资源\\js文档\\完整版W3CSchool线下教程.chm::/www.w3school.com.cn/css/css_margin.asp.htm" \o "CSS 外边距) 的方式





padding

padding 属性接受长度值或百分比值，但不允许使用负值

上、右、下、左

h1 {padding: 10px 0.25em 2ex 20%;}

h1 {

padding-top: 10px;

padding-right: 0.25em;

padding-bottom: 2ex;

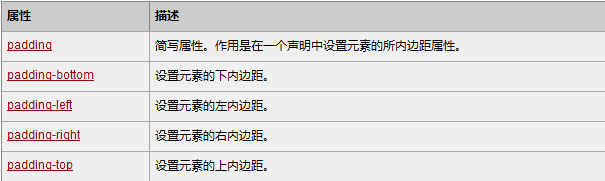
padding-left: 20%;

}

td.test2 {padding: 0.5cm 2.5cm}上和下内边距是 0.5cm，左和右内边距是 2.5cm

百分数值是相对于其父元素的 width 计算的，上下内边距的百分数会相对于父元素宽度设置，而不是相对于高度

内边距属性



Border

边框绘制在“元素的背景之上”

p.aside {border-style: solid dotted dashed double;}

为类名为 aside 的段落定义了四种边框样式：实线上边框、点线右边框、虚线下边框和一个双线左边框。

**希望边框出现，就必须声明一个边框样式：border-style**

把单边属性放在简写属性之后。

**单边样式**

p {border-style: solid solid solid none;}

p {border-style: solid; border-left-style: none;}

**宽度：**

由于 border-style 的默认值是 none，如果没有声明样式，就相当于 border-style: none。因此，如果您**希望边框出现，就必须声明一个边框样式**。

p {border-style: solid; border-width: 15px 5px 15px 5px;}

p {border-style: solid; border-width: 15px 5px;} （值复制）

p {

border-style: solid;

border-top-width: 15px;

border-right-width: 5px;

border-bottom-width: 15px;

border-left-width: 5px;

}

**透明边框**

创建有宽度的不可见边

<html>

<head>

<style type="text/css">

a:link, a:visited {

border-style: solid;

border-width: 5px;

border-color: transparent;

}

a:hover {border-color: gray;}

</style>

</head>

<body>

<a href="#">AAA</a>

<a href="#">BBB</a>

<a href="#">CCC</a>

</body>

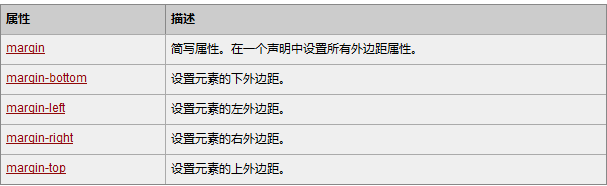
</html>

**边框属性**



Margin

外边距属性



h2 {

margin-top: 20px;

margin-right: 30px;

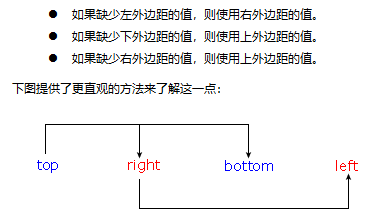
margin-bottom: 30px;

margin-left: 20px;

}

p {margin: 20px 30px 30px 20px;}

**值复制**

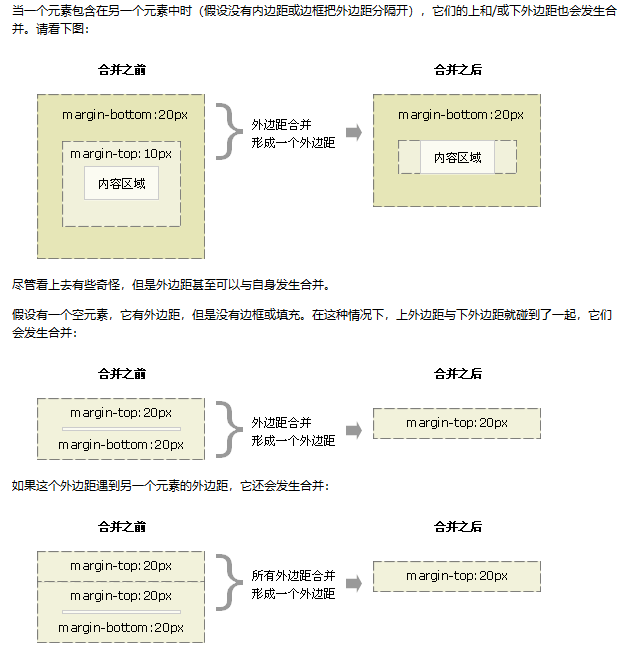


**外边距合并**

当两个垂直外边距相遇时，它们将形成一个外边距

合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者

只有普通文档流中块框的垂直外边距才会发生外边距合并。行内框、浮动框或绝对定位之间的外边距不会合并。



CSS定位

JavaScript

改进设计、验证表单、检测浏览器、创建cookies

向 HTML 页面添加交互行为。

脚本语言（脚本语言是一种轻量级的编程语言）。

由数行可执行计算机代码组成。

通常被直接嵌入 HTML 页面。

是一种解释性语言（就是说，代码执行不进行预编译）。

Bootstrap

1. 响应式表格/图片：根据设备或浏览器窗口的大小自动调整布局和显示效果

<div class=”table/img-responsive”>

1. <span>是HTML中的内联元素，用于在文档中标记一段短文本或行内内容。
   1. 不会独占一行，而是内联显示在文本中的指定位置。
2. Bootstrap&JavaScript&JQuery
   1. Bootstrap：前端框架，提供样式和组件来快速搭建网页
   2. JavaScript：编程语言，编写网页的交互逻辑
   3. JQuery：基于JavaScript的库，简化JavaScript的编程

ASP.NET

一、ASP.NET和ASP.NET MVC

ASP.NET MVC:

1. 版本：System.Web.Mvc——5.2.2.0
2. AssemblyInfo.cs：发布后，项目的ddl文件一个总的描述
3. App\_Data：数据脚本/数据库文件
4. App\_Start：包含了程序初始化需要的一些类
5. Content：保存css文件
6. fonts：保存字体文件
7. Scripts：保存js文件

MVC：

1. Model：存放实体对象
2. View：表现层，和用户直接交互的界面
3. Controllers：存控制器，控制业务逻辑、选择数据传输……
4. Global.asax：全局的配置文件，MVC程序的启动是从这里开始的
5. Web.config：MVC程序的配置文件

ASP.NET开发：WebForm开发

1. aspx文件：涵盖了MVC中三个元素

二、项目发布 IIS发布

1.依赖于Windows—依赖IIS—无法跨平台

2.安装IIS作为服务器（本地）

3.云服务器——远程桌面

服务器管理器：通过服务器管理器来增加角色和功能

准备发布1：

1. 打开IIS
2. 添加一个网站，物理路径指向项目的根目录，设定一个端口号（建议在5000-10000）
3. 直接访问

准备发布2：

1. 右键鼠标，选择发布
2. 设置一些配置

权限问题：

配置网站所在物理路径的权限：

1. 增加两个账号：
   1. IUSR——设置权限为完全控制；
   2. IIS\_IUSRS——设置权限为完全控制；
2. 开发功能开发—数据传值方式
3. ViewBag

base.ViewBag.User1 = "张三";

1. ViewData

base.ViewData["User2"] = "李四";

1. TempData

base.TempData["User3"] = "王五";

1. HttpContext.Session

base.HttpContext.Session["User4"] = "赵六";

1. model

object model = "Richard";

return View(model);

index：

@model string

@Model

不同方式，值会覆盖

注意：

1. ViewBag和ViewData：如果指定的Key是一样的，后者会覆盖前者
2. TempData和HttpContext.Session：传值，跳转到其他方法后，依然可以获取到值。——TempData其实是通过Session来实现
3. model传值不要把数据类型定义成一个String类型，因为有重载存在，返回指定的视图类型。
4. ASP.NET MVC集成日志log4net

log4net：

1. 支持多种方式记录日志
2. 日志组件，第三方开发的程序集

控制器输出

文本记录

记录到数据库中去；

如何集成log4net到ASP.NET MVC中呢？

1. nuget引入程序集log4net（对解决方案进行操作）
2. 准备配置文件，配置文件的属性设置为始终复制
3. 读取配置文件，准备帮助类库

log4net如何记录日志到数据库中去？

1. Global区域

Global.asax：全局文件

一个文本文件，提供全局可用代码。这些代码包括应用程序的事件处理程序、会话事件、方法和静态变量。有时该文件也被称为应用程序文件

MvcApplication：MVC应用程序

Application\_Start：程序的入口，即程序启动后，从这里开始执行；且每次启动这里只执行一次

网站的初始化动作：

/// 各种注册

/// 为了在网站运行的时候，把需要的各种组件都先装配进来

AreaRegistration.RegisterAllAreas();//注册区域

FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);//注册全局Filter

RouteConfig.RegisterRoutes(RouteTable.Routes);//注册路由

BundleConfig.RegisterBundles(BundleTable.Bundles);//注册Bundles——引用JS/CSS需要的组件

区域：开发的时候，功能模块可能会很多（权限管理、用户管理等）

——按照模块的划分，进行模块化的开发——“新建基架项”

每一个模块都可以对应一个MVC基架，每一个板块的职责更加清晰——模块化开发

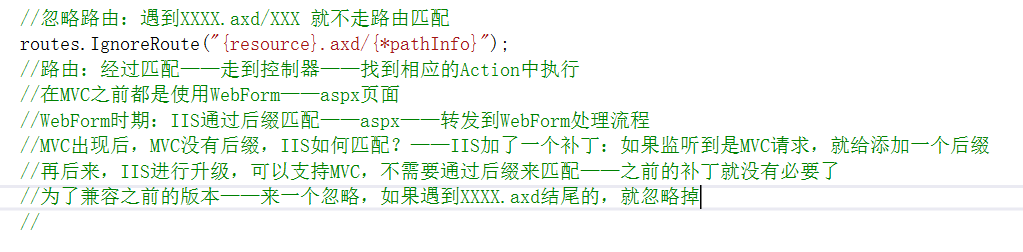
1. 路由——RouteConfig.cs
2. **路由**

ASP.NET MVC Route——转发请求：

1. 客户端发起请求
2. 到达IIS
3. IIS转发到程序集
4. 其中经过一个路由匹配，转发到匹配的控制器，匹配的Action中去处理

RouteConfig：在这里进行路由规则的声明

1. **忽略路由**



1. MVC插件式开发
2. 控制器不一定写在文件夹Controllers下，可以写在其他地方。

为什么？

因为网站在启动的时候，需要做初始化——扫描控制器，然后保存控制器。

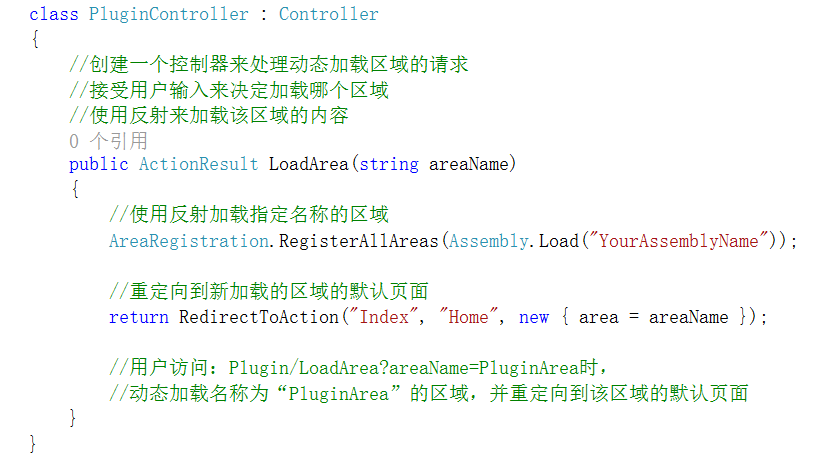
扫描控制器的规则：看是否继承自Controller父类；当请求来了，要真的使用这个控制器实例的时候，是从存储介质中取出来然后进行初始化的。

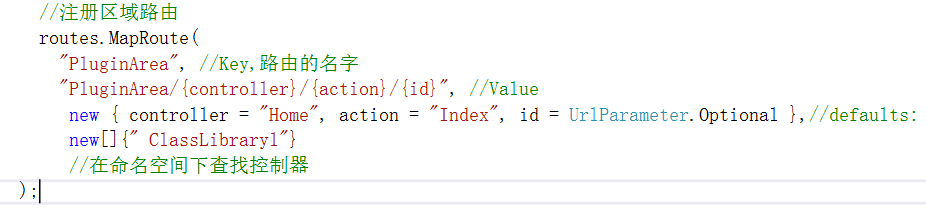
可以把控制器的部分（逻辑计算的部分）独立开，放到一个类库中：好处——插件式开发：

1. 把一个大的业务逻辑，做拆分；每一个类库分别处理部分内容；每一块内容当作一个插件来完成。
2. 大项目——划分为N个模块——每个模块独立出来做一个插件——插件式开发
3. 部署的时候，只要把所有的插件加载进来，就可以做到多个插件组合成一个大平台。
4. 热插拔——网站没有停止；不影响系统工作的基础上，接入或移出正在工作的系统

如果有新功能发布，不需要停止服务器，可以把新的插件发布上去：

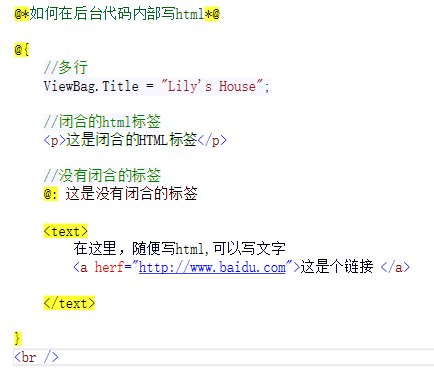
1. 准备好插件
2. 触发一个增量初始化的动作——注册路由规则、扫描控制器等





1. Razor语法
2. 在视图中可以写入各种Html标签
3. .cshtml文件并不是一个Html页，而是一个cs文件（类，后台代码文件）——可以识别HTML各种标签
4. .cshtml既可以写html，也可以写后台代码——需要增加一个@{}







易错点

前后格式要相同：

decimal? aa = dt.Rows[i]["提产目标"].ToDecimalOrNull();

2024.3.4

1. ajax:
   1. type：请求类型
   2. url：请求链接地址
   3. contentType：发送给服务器的格式（属于请求头中的字段，服务器端解析该字段来获知请求中的消息主体的编码方式，再对主体进行解析）
   4. dataType：收到数据的格式
   5. data：发送给服务端的数据
2. &nbsp:HTML中的非断空格实体
3. beetl标签：将重复使用的HTML代码用一行代码标签替代，易于维护

<#table id=”……”/>

1. HTTP请求=状态行+请求头+消息主体

2024.3.8

1. 枚举：为变量定义一个预设的值集合，目的是限制变量只能从预定义的值集合中取值。
   1. 适用于具有固定集合值的变量（例如星期、月份、操作状态）
2. GetWeekOfYear：返回指定日期是一年中的第几周的整数
   1. GetWeekOfYear(DateTime time, CalendarWeekRule.FirstDay, DayOfWeek.Sunday);
3. CalendarWeekRule：一个枚举，定义了计算一年中第一周必须满足的条件。
   1. FirstDay：第一周是包含一年中第一天的那一周
   2. FirstFullWeek：第一周是该年中第一个完整的周，每一天都要在新的一年
   3. FirstFourDayWeek：第一周是包含一年中第一天，并且该周至少有四天在新的一年中的那一周

2024.3.10

Request.QueryString["method"]

1. Request指客户端（浏览器）向服务器发送的请求。

Request通常包括：

1. 请求方法：客户端希望服务器执行的操作（GET、POST）
2. URL：包含协议（HTTP/HTTPS）、主机名、路径等。
3. 请求头：附加的请求信息
4. 请求体：仅在使用POST、PUT等方法时存在，比如表单的字段值等
5. QueryString：用来处理URL查询字符串的方法。URL查询字符串指URL中问号后面的部分，用来传参数给服务器或网页。

QueryString方法将查询字符串解析为键值对的形式。

使用场景：向URL中添加/从URL中获取参数。

Page\_Load(object sender, EventArgs e)

1. EventArgs e：.NET框架中事件处理程序的参数（事件处理程序需要接收一个参数来提供与事件相关的信息）
2. EventArgs：.NET框架中的一个基类，用于作为事件参数的基本类。包含一些通用的数据，例如事件发生的时间、是否取消事件等。
3. object sender： sender表示事件的发送者（触发事件的对象）

2024.3.18

NFine框架和ASP.NET

NFine框架是一个基于ASP.NET MVC的开源框架，拓展了ASP.NET MVC 框架的功能。

2024.4.16

临时表：

1. 需要可以随时创建，用完随时释放
2. 临时存储大量数据
3. 没有正式的表结构
4. 查询语句里面只能出现一次

视图:

虚拟表

一系列查询语句

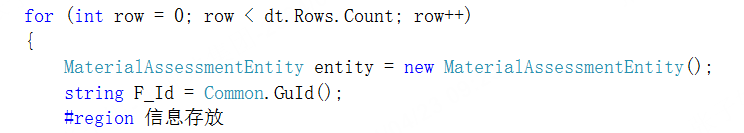
报错

建立临时表：unknown error 1060:列名是否有重复的

“DBHelper: {}”：sql语句一定要加封号！！！

注意“limit”放置的位置

导入注意row的起始值是0还是1



div内元素垂直居中：

style="transform:translateY(50%)"

# 数据库逻辑

分组“group by”可以筛选掉重复的，但也会隐藏数据，后续如果要进行匹配，注意不要掉了：后续按A，B匹配，可以group by A，B

有二级总成的，一级总成时，要保证前面数据都是全的，group by的时候要注意，只需要在最后获得最终结果的地方，group by 到相应层级就行

group by可以理解为总成到哪个字段层面

想做二级总成（所有数据），到最后一步再进行判断是否显示一级总成

if(A == “B”)

如果A和B看起来相等，却不执行条件体，检查是否有空格！！！

select 最后一个字段后面不要有逗号！

sum(ifnull(QTY\_WMS ,0))

**临时表只能在同一句SQL语句中出现一次**

**order by 根据ASCII码来排序**

**Union**

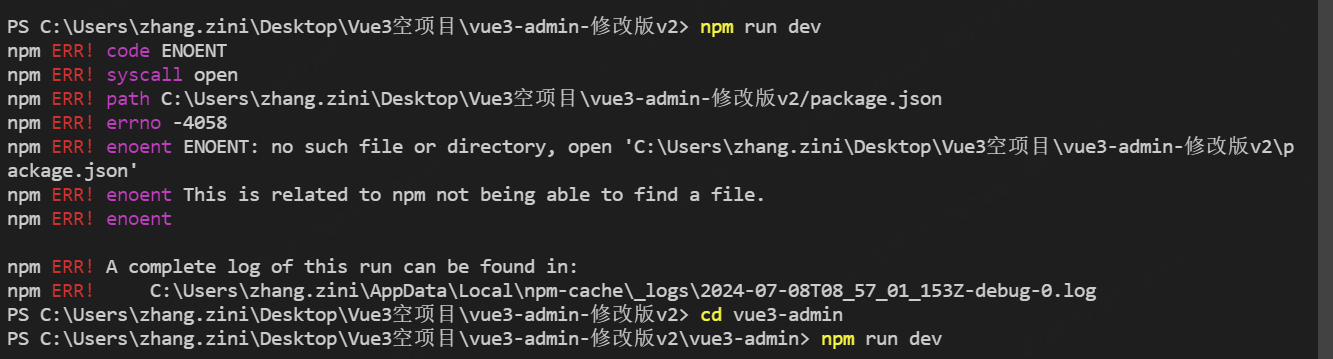
**sql预处理**

**sql注入**

**MySQL CONCAT**

# VSCODE

npm run dev要在package.json所在根目录执行



**界面跳转：**

window.location.href = url

**多选下拉框：**

重大项目----新建----涉及工厂

初始化—设置隐藏选项—事件监听—事件处理函数—处理已选择的值—更新展示选项

配置图表选项：

ECharts—配置选项option

title

tooltip（提示框）

xAxis（X轴）

show：是否显示坐标轴

type：坐标轴类型（value：数值轴，category：类目轴，time：时间轴，log：对数轴）

axisLine：坐标轴轴线刻度

axisTick：坐标轴刻度的样式

splitLine：网格线的样式

yAxis：Y轴

数据系列：series

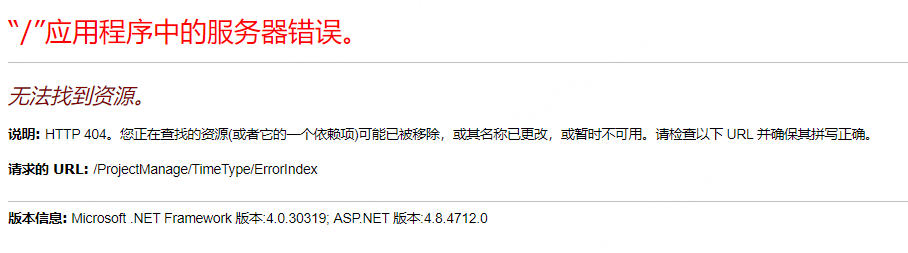
解决方案文件看不到？

显示所有文件

代码错误没解决，重新生成解决方案可能出现“.dll”文件错误——因为错误没解决，所以编译不成功

**TimeType/ErrorIndex**

因为在Controller里面限制了权限



**aDatasort undefined:**



列号是否对得上

**SQL错误码1064：语法错误**

**union创建临时表**

**create** **temporary** **table** table1 **as**

**select** '关键指标' **as** AttentionItem,F\_KeyMeasure **as** Progress,F\_KeyMeasureRisk **as** Risk **from** cp\_weeklyreport\_basicinfo

**union**

**select** '仿真验证',F\_SimulationVertification,F\_SimulationVertificationRisk **from** cp\_weeklyreport\_basicinfo

**union**

**select** '试制供货',F\_PrototypeSupply,F\_PrototypeSupplyRisk **from** cp\_weeklyreport\_basicinfo

**union**

**select** '试验情况' ,F\_TestStatus,F\_TestStatusRisk **from** cp\_weeklyreport\_basicinfo

**union**

**select** '开模情况',F\_MoldStatus,F\_MoldStatusRisk **from** cp\_weeklyreport\_basicinfo

**union**

**select** '软件受控',F\_SoftwareControlled,F\_SoftwareControlledRisk **from** cp\_weeklyreport\_basicinfo

**union**

**select** '产线情况',F\_ProductionLineStatus,F\_ProductionLineStatus **from** cp\_weeklyreport\_basicinfo

**union**

**select** '质量问题',F\_QualityIssues,F\_QualityIssuesRisk **from** cp\_weeklyreport\_basicinfo

;

多个导入的时候，多个Irepository和repository，多个service——Entity不是同一个的情况下

input：



value是button的显示值

传值用data-value

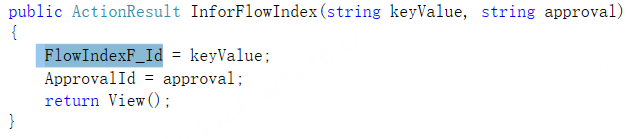
用于所有类型的AJAX提交表单

：主要用于表单提交。

???



???



传参：

前后端参数名字要一样，不然接收不到参数



记得加双引号！！



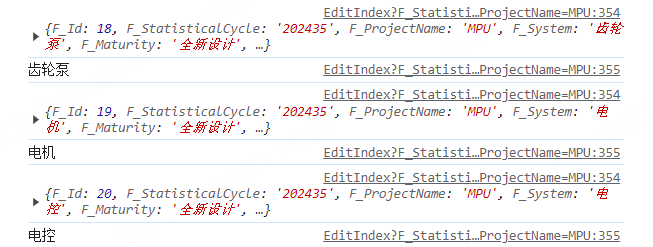
full和data





data指每一列的数据仅限于**“data”指明了列名**，否则data和full一样

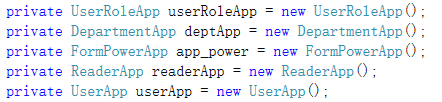




更改单元格内容再显示更新后的内容

1. Failreason：
   1. 定位，更新数据库内容，重新加载表
2. 更改基础信息表——有id定位
   1. 定位到哪条数据的哪一字段，更新数据库，返回更新后的数据，重新赋值

**一些基础文件**

****

**一个表单使用两个选人弹窗，注意tbodyReader的id不能重复！**

**设置一个全局变量记录type，更新不同的refreshGridListReader**

**textarea标签的值放在两标签之间！！**

****

**name属性的使用：**

****

****

****

**线上汇报**

****

**画图，接口，展示**

****

**数据库的操作**

****

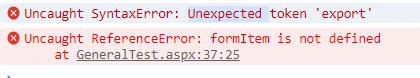
****

**连接数据库**

**链接到新标签页**

****

****

****

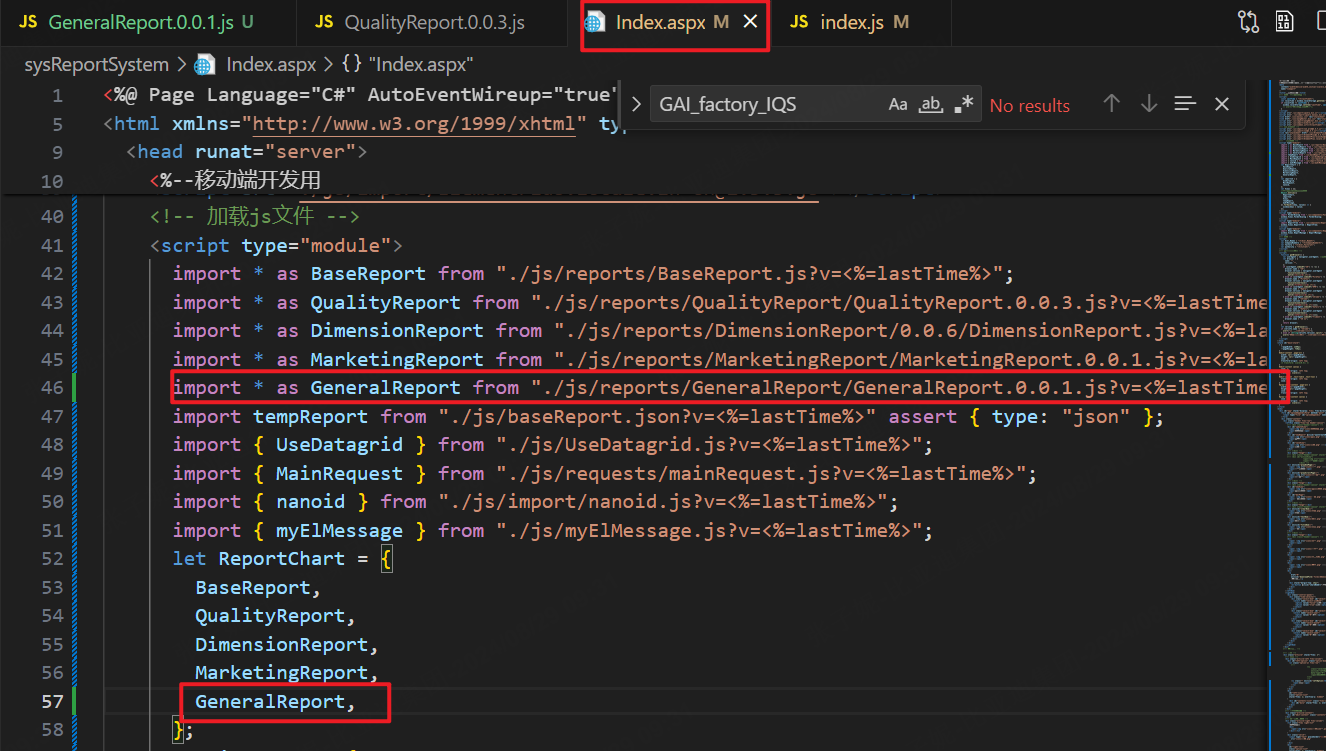
**formItem报错：要注释**

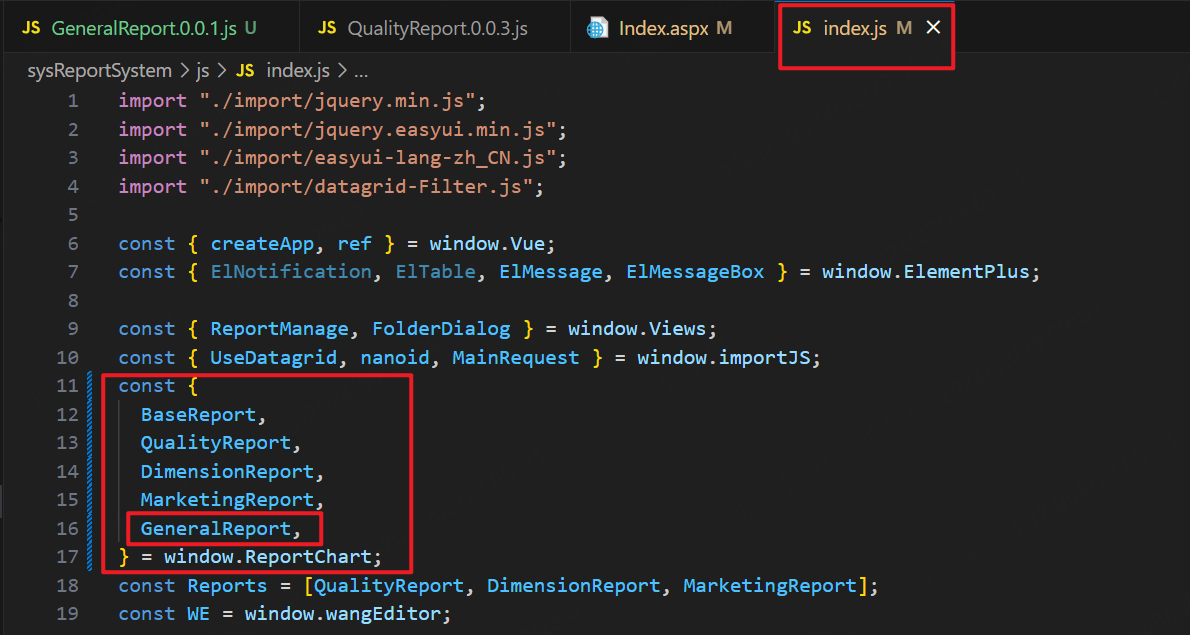
****

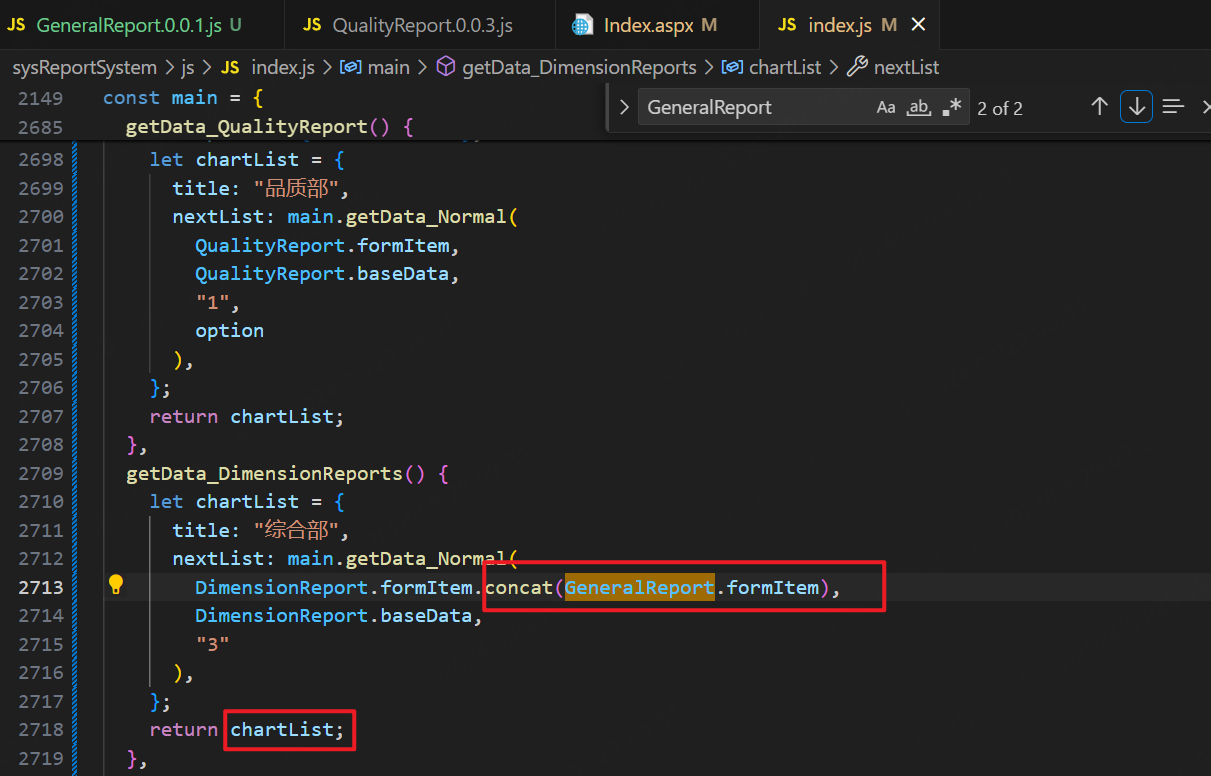
**识别回车符：**

****

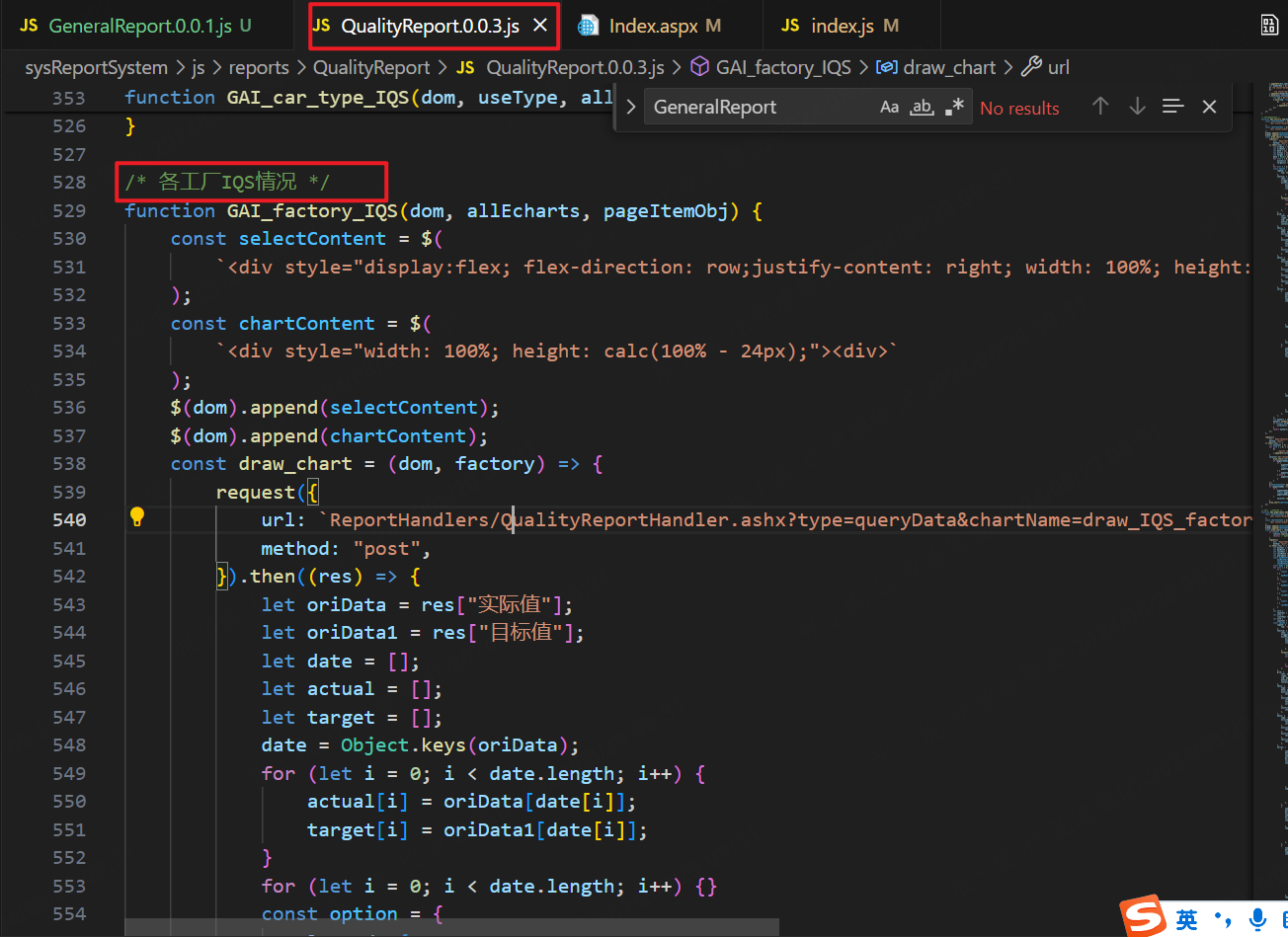
**到线上汇报系统中显示：**

****

****

****

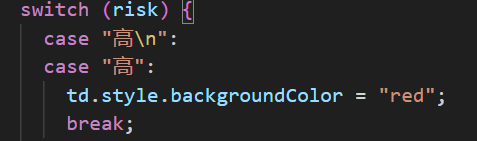
**筛选器参照：**

****

**华为云：**

**获取**

**switch共用代码块：**

****

**input-button 和button的区别：**

****

**button标签，默认type=”submit”，会提交表单**

**input-button 普通按钮，不会提交**

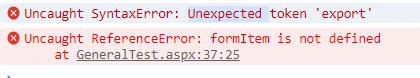
****

**id和点击函数不可同名，否则报“not a function”错误**

**链接到新标签页**

****

****

****

**formItem报错：要注释**

****

**识别回车符：**

****

**FindEntity:**

****

**只能写LINQ语句，不能写SQL**

**不需要加双引号**

****

**try-catch和throw exception**

****

**外层要有try-catch，代码才不会中断**

**limit：**

**前面要有空格**

****

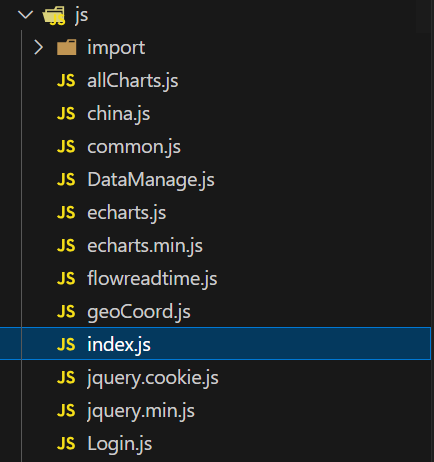
****

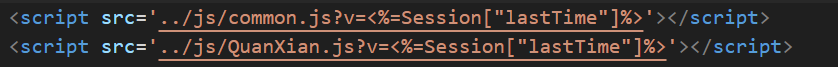
**可以进一步查看详细信息**

**Entity里面的字段和数据库表一定要一一对应**

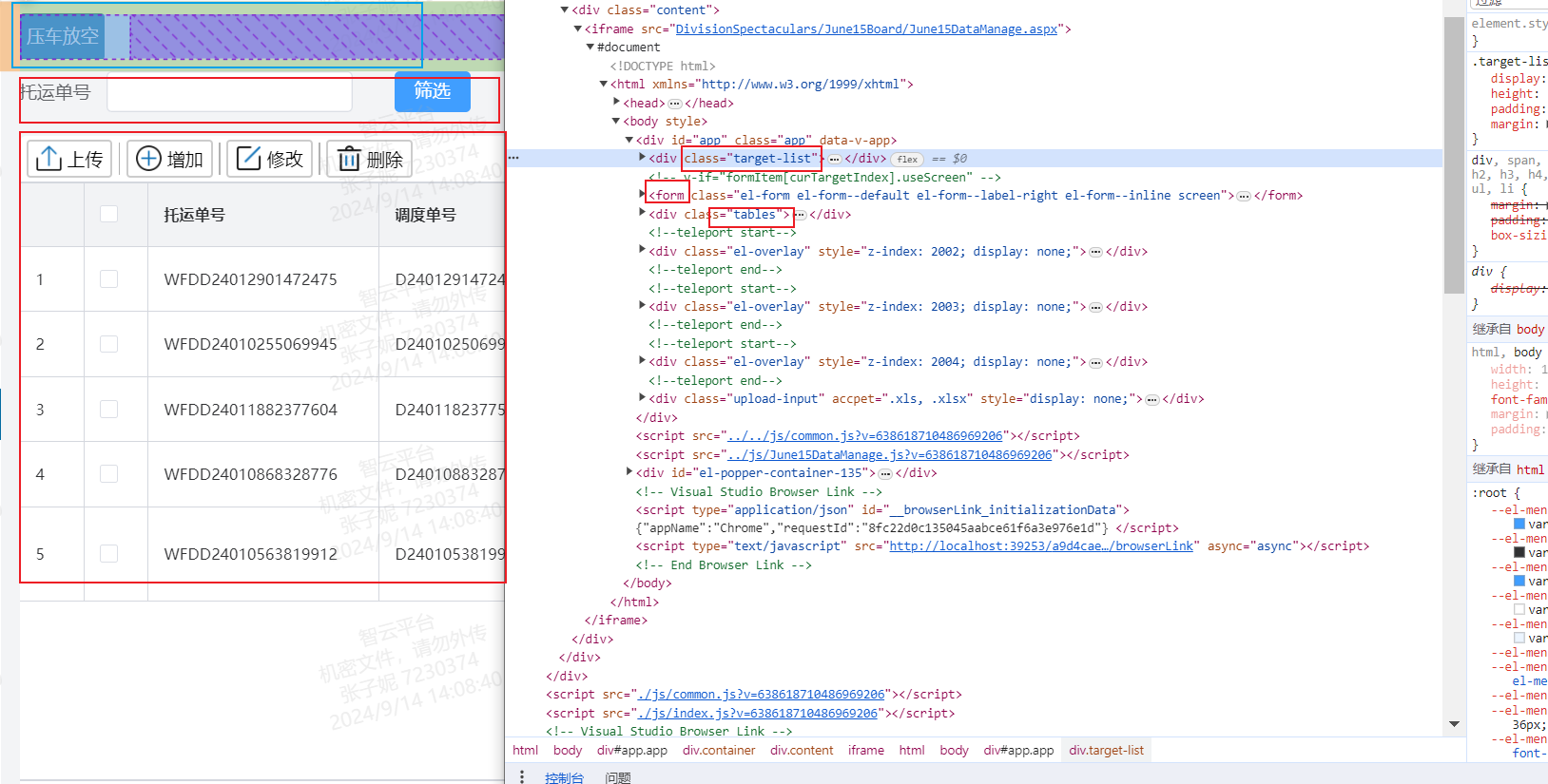
**但是可以建立一个展示List,就不需要一一对应了**

**？？**

****

****

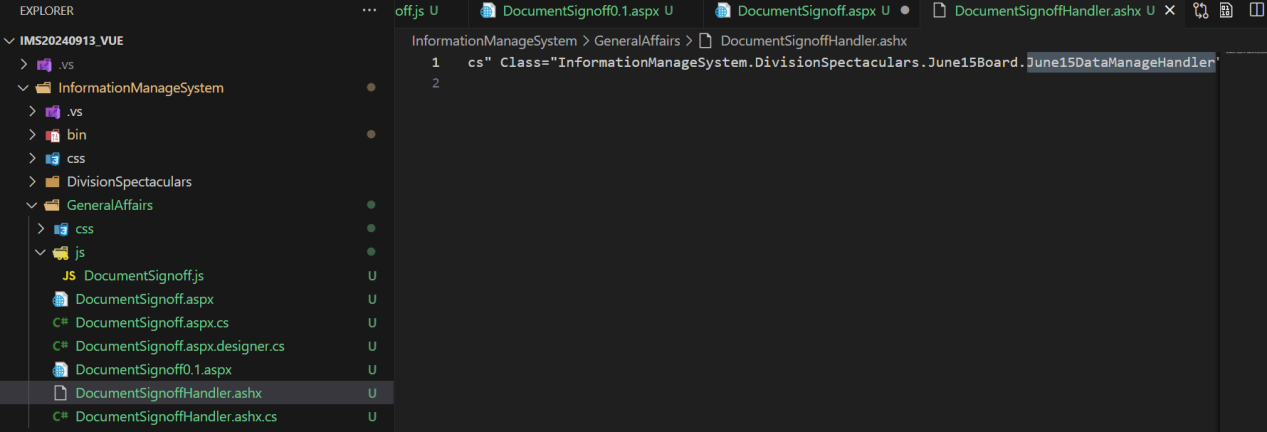
****

****

****

****

**路由跳错文件：**

****

**没有改**

**sql语句判断，一个等号**

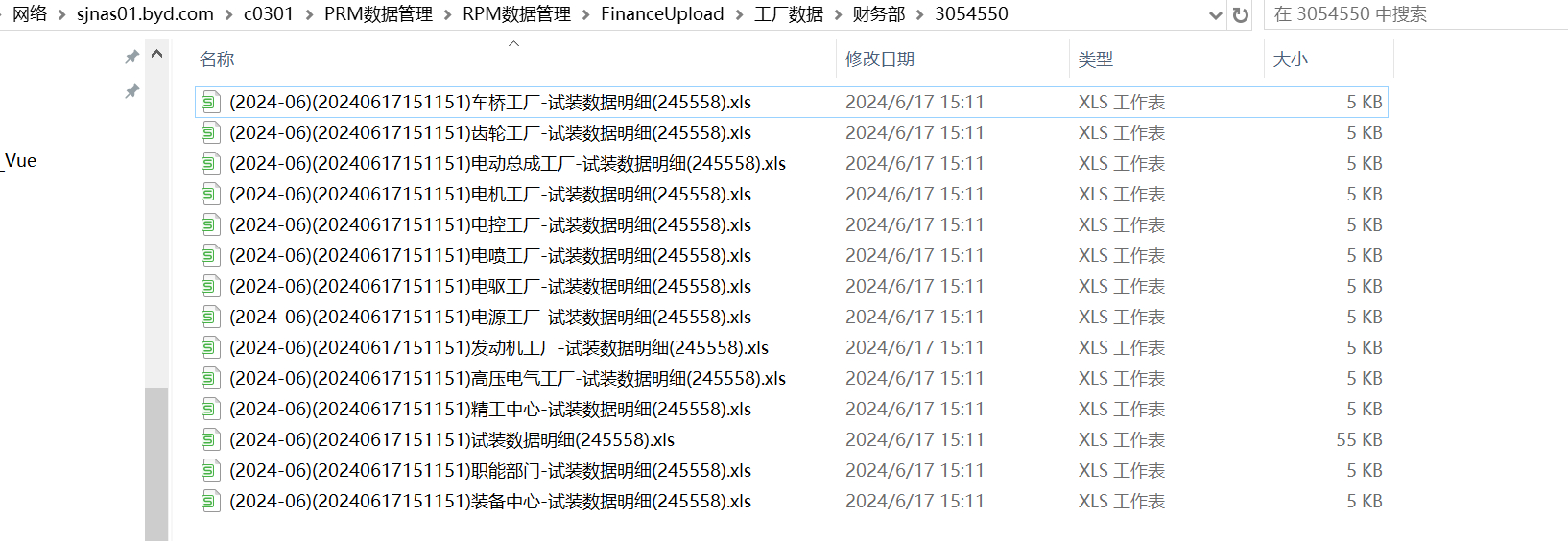
****

无法选中，加了scope，文件名下的数据不显示

span标签赋值，要使用.text，不能用.val



嵌套html语句，要用.html



前后端传参，特殊符号会被解释为空格，解决方法：



判断，或语法：



# Git相关

提交到服务器，如果期间还有别的用户提交了代码，

1. 先重新拉取成最新代码
2. 重新生成解决方案
3. 提交合并
4. 虚拟机内，更新代码：git pull

更新本地代码：

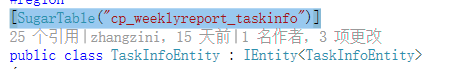
SourceTree：先获取再拉取

保存提交代码：

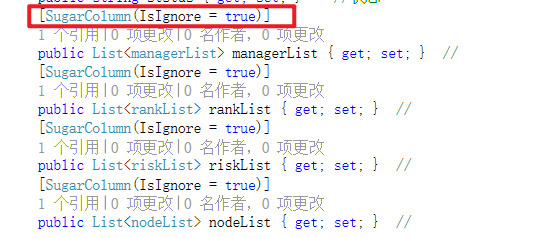
SourceTree：暂存所有——提交——推送

暂存所有——提交——拉取——解决冲突（重新生成解决方案）——推送

sql sugar映射：



将实体映射到数据库表（实体名和表名不一致时）

-

实体有某些字段而数据库表中没有时，使用这个可以不将这些字段存库

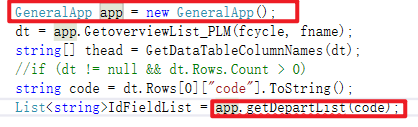


加了static的方法不可以用实例对象来访问

正确:

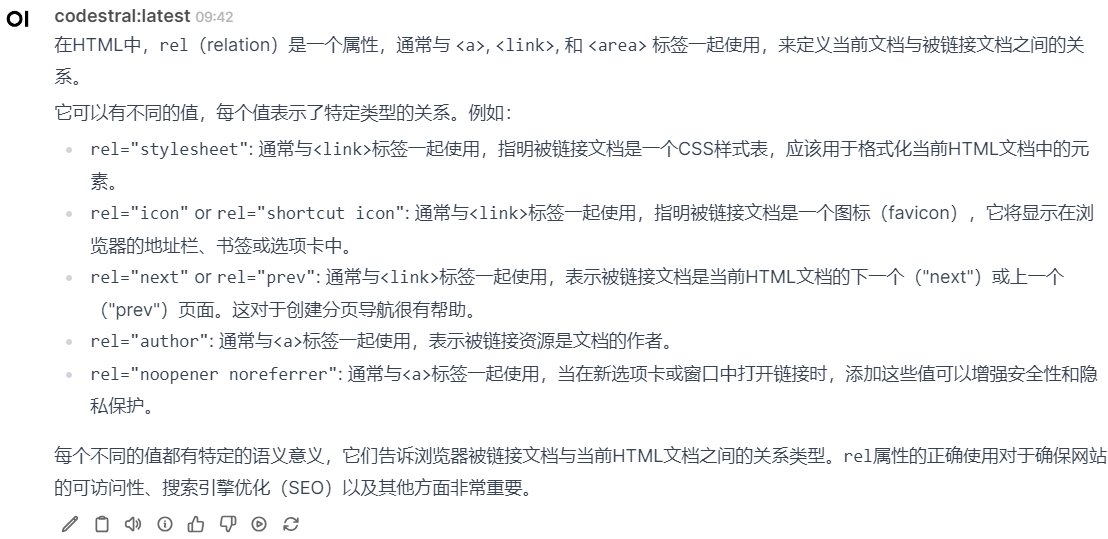


错误：





rel





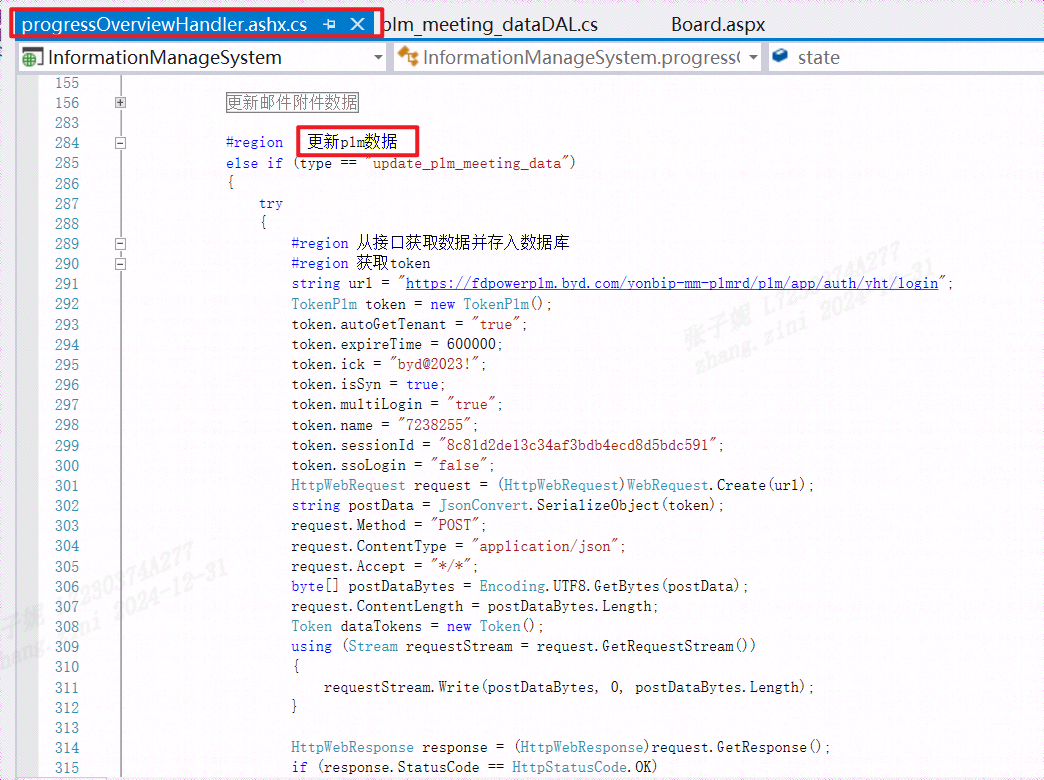
app要实例化：

new app 或者加上static



接口获取相关：

先登录获取token——IMS



后端改了一定要重新编译再提交