WebPascal 5 用户手册

前言

WebPascal 5 是专为Delphi开发者打造的Web开发后端中间层框架，采用Pascal原生语法、纯脚本驱动模式，无需学习陌生语言，即可快速实现Web应用、HTTP接口等开发与部署。本手册将详细介绍框架的安装部署、基础使用、核心功能操作、常见问题排查等内容，帮助开发者快速上手，高效完成开发工作。

适用人群：Delphi开发者（具备基础Pascal语法基础，无需Web开发经验）；技术运维人员（负责框架部署与日常维护）。

注意事项：

1. 框架仅支持Windows系统部署；
2. 脚本文件需统一采用UTF-8无BOM编码；
3. 部署前请确保服务器已开放对应端口（默认8833）。

第一章 框架概述

1.1 框架简介

WebPascal 5 是一款轻量化、开箱即用的Web后端框架，核心定位为“Delphi开发者的Web开发工具”，兼容Delphi原生语法，无需编译，纯脚本驱动，集成多数据库支持、计划任务、安全防护等核心能力，让桌面开发经验无缝迁移至Web开发场景。

1.2 核心功能清单

- 基础能力：Pascal语法全兼容、模块化开发、全场景日志系统；
- Web能力：HTTPS支持、CORS跨域、伪静态、文件上传、HTTP请求处理；
- 数据处理：JSON解析、正则表达式、编码转换、字符串处理；
- 数据库支持：SQLite、MySQL、SQL Server、Oracle统一API操作；
- 运维能力：计划任务调度、自动化资源管理、异常调试；
- 安全能力：JWT身份认证、数据加密、SQL防注入、异常防护；
- 工具集：验证码、二维码、时间处理、雪花算法等一键调用。

1.3 目录结构说明

框架解压后默认目录结构如下，各目录功能固定，请勿随意修改目录名称，避免影响框架运行：

```
WebPascal/
├─── script/     # 脚本目录（核心开发目录，支持多级子目录）
├─── task/       # 计划任务脚本目录（存放定时任务脚本，需配置任务规则）
├─── uploads/    # 上传文件目录（自动按日期分目录存储，无需手动管理）
├─── www/        # 静态资源目录（存放HTML、CSS、JS、图片等文件）
├─── log/        # 日志目录（自动按日期生成日志，30天自动清理）
└─── cache/      # 临时文件目录（自动清理，无需手动操作）
```

第二章 安装与部署

2.1 环境要求

- 系统要求：Windows 7 及以上（32位/64位）、Windows Server 2012 及以上；
- 硬件要求：CPU ≥ 1核，内存 ≥ 1GB，磁盘空间 ≥ 100MB（不含项目文件）；
- 额外依赖：无需安装Delphi IDE，框架自带运行环境；
- 网络要求：若需外部访问，需开放框架配置的端口（默认8833）。

2.2 部署步骤（3步完成）

1) 下载与解压
- 从官方地址（https://webpascal.com）下载WebPascal 5 压缩包；
- 将压缩包解压至服务器或本地任意目录（如：D:\WebPascal）；
- 解压后确认目录完整，重点检查WebPascal.exe（运行程序）、SrvConfig.ini（配置文件）是否存在。

2) 配置修改
- 打开解压目录下的SrvConfig.ini文件（可用记事本、Notepad++等编辑器打开）；
- 核心配置项修改（其他配置保持默认即可）：
  - Port=8833 （框架运行端口，可修改为任意未被占用的端口，如80、8080）；
  - CORS=1 （是否允许跨域，1=允许，0=禁止，默认允许）；
  - GZip=1 （是否允许GZIP压缩，1=允许，0=禁止，默认允许）；
  - UpDir=.\Uploads （文件上传路径，默认即可，可自定义绝对路径）。
- 修改完成后保存配置文件，关闭编辑器。

3) 启动服务
- 双击解压目录下的webpascal.exe，启动框架服务；
- 启动成功后，会弹出控制台窗口，显示“HTTP Server running on :8833, press Enter to stop...”（端口与配置一致）；
- 验证启动：打开浏览器，访问http://localhost:8833（本地）或http://服务器IP:端口（外部），若看到演示页面，说明部署成功；
- 后台运行（可选）：若需长期运行，可通过nssm工具将webpascal.exe注册为Windows服务，避免控制台关闭后服务停止。

2.3 部署常见问题

- 问题1：启动后控制台闪退 → 解决方案：检查端口是否被占用，修改SrvConfig.ini中的Port为未占用端口；
- 问题2：浏览器无法访问 → 解决方案：检查防火墙是否开放对应端口，确认服务已正常启动；
- 问题3：启动后提示“错误信息” → 解决方案：根据提示进行处理。

第三章 基础开发入门

3.1 开发准备

- 开发工具：无需安装Delphi IDE，可使用记事本、Notepad++、VS Code等任意文本编辑器；
- 脚本规范：所有Web脚本需放在script目录下，后缀为.api（如hello.api），编码为UTF-8，也支持其他扩展名或不带文件扩展名；
- 访问规则：脚本访问地址格式为 http://域名/ask/脚本名.api（如：http://localhost:8833/ask/hello.api）。

3.2 第一个Web脚本（Hello World）

1. 打开script目录，新建文本文件，命名为hello.api（注意，后缀为api，而不是txt）；
2. 在文件中写入以下Pascal代码（语法与Delphi完全一致）：

```
// Hello World 示例
writeln('Hello WebPascal 5!');
writeln('Delphi开发者的Web开发专属利器！');
```

3. 保存文件（确保编码为UTF-8）；
4. 访问测试：打开浏览器，输入http://localhost:8833/ask/hello.api，页面将显示脚本输出的内容，说明脚本运行成功。

3.3 基础语法说明

WebPascal 5 完全兼容Delphi原生语法，以下为开发中常用的基础语法，与Delphi桌面开发一致：

- 变量声明：支持var关键字，如var i: Integer; var s: string;
- 循环语句：for、while、repeat-until，如for i := 1 to 10 do begin ... end;
- 条件语句：if-else、case，如if i > 5 then writeln('大于5') else writeln('小于等于5');
- 常用函数：保留Delphi高频函数，如writeln（输出内容到页面）、FormatDateTime（时间格式化）、IntToStr（数值转字符串）、Trim（去空格）等；
- 注释：支持//单行注释、{...}、(* *)多行注释，与Delphi一致。

3.4 模块化开发（include脚本引入）

当脚本较多时，可通过include()函数引入其他脚本，实现代码复用，避免重复编写：

1. 在script目录下新建common.api，写入公共函数：

```
// 公共函数：计算两个数的和
function Add(a, b: Integer): Integer;
begin
  Result := a + b;
end;
```

2. 在另一个脚本（如test.api）中引入common.api并调用函数：

```
// 引入公共脚本
```

```
include('common.api');
// 调用公共函数
var sum := Add(3, 5);
writeln('3 + 5 = ' + IntToStr(sum));
```

3. 访问http://localhost:8833/ask/test，即可看到计算结果，说明引入成功。

注意：include()函数的路径为相对路径，默认以script目录为根目录，若脚本在子目录，需加上子目录名称（如include('/user/common.api')）。

第四章 核心功能操作指南

4.1 数据处理操作

4.1.1 JSON处理

WebPascal 5 内置JSON处理函数，支持JSON字符串与对象的相互转换、遍历、键值操作，无需额外引入组件：

```
// 1. JSON字符串转对象
var jsonStr := '{"name":"张三","age":18,"gender":"男"}';
var obj := json.parse(jsonStr);
// 2. 获取JSON对象的值（支持默认值操作符??，避免空值报错）
var name := obj.name;          // 直接获取，若键不存在返回null
var email := obj.email ?? '未设置'; // 键不存在时返回默认值
// 3. 修改JSON对象的值
obj.age := int(20);
// 4. JSON对象转字符串
var newJsonStr := json.stringify(obj);
writeln(newJsonStr);
writeln(unicodetoansi(newJsonStr));
// 5. 遍历JSON对象
for var i := 0 to obj.length() - 1 do
begin
  writeln(obj.ElementName(i)+' : '+obj[i]+' | '+obj[i].TypeName());
end;
```

4.1.2 编码转换

支持URLencode、Base64、HEX转换，以及Unicode解码，适配各类数据传输场景：

```
// 1. URL编码（支持emoji）
var urlStr := 'https://webpascal.com?name=张三😊';
var urlEnStr := urlencode.encode(urlStr);
writeln('URL编码后：' + urlEnStr);
// 2. URL解码
var urlDeStr := urlencode.decode(urlEnStr);
writeln('URL解码后：' + urlDeStr);
// 3. Base64编码
var base64EnStr := base64.encode('WebPascal 5');
writeln('Base64编码后：' + base64EnStr);
// 4. Base64解码
```

```
var base64DeStr := base64.decode(base64EnStr);
writeln('Base64解码后: ' + base64DeStr);
// 5. HEX编码
var hexEnStr := StrToHex('abc测试123');
writeln('Hex编码后: ' + hexEnStr);
// 6. HEX解码
var hexDeStr := HexToStr(hexEnStr);
writeln('Hex解码后: ' + hexDeStr);
```

4.1.3 正则表达式

实现匹配、替换、分组提取等功能，满足字符串解析、校验需求：

```
// 1. 匹配是否符合规则（如手机号校验）
var phone := '13800138000';
var isPhone := regex.ismatch(phone, '^1[3-9]\d{9}$');
if isPhone then
  writeln('手机号格式正确')
else
  writeln('手机号格式错误');
// 2. 提取匹配内容（如提取URL中的域名）
var url := 'https://webpascal.com/docs';
var domain := regex.match(url, '(?<=https://).*?(?=/)');
writeln('提取的域名: ' + domain);
// 3. 字符串替换（替换所有数字为*）
var str := 'abc123def456';
var newStr := regex.replace(str, '\d+', '*');
writeln('替换后: ' + newStr);
```

4.2 数据库操作指南

WebPascal 5 原生适配SQLite、MySQL、SQL Server、Oracle等主流数据库，一套统一API即可操作所有数据库，无需适配不同语法。

4.2.1 数据库连接

通过db.conn()函数一行代码完成数据库连接，参数为（数据库类型，连接字符串）：

```
(* 连接 MSAccess 数据库的连接配置 *)
db.conn('mydb','DriverID=MSAcc;Database=.\app.mdb;');

(* 连接 SQLite 数据库的连接配置 *)
db.conn('mydb','DriverID=SQLite;Database=.\sqlite.db;');

(* 连接 MSSQL2000 数据库的连接配置 odbc模式 *)
db.conn('mydb','DriverID=MSSQL2000;Server=127.0.0.1;Database=test;User_name=sa;Pas
sword=;');
db.conn('mydb','DriverID=MSSQL2000;Database=test;Server=127.0.0.1;OSAuthent=Yes;')
;

(* 连接 MSSQL2000以上版本 数据库的连接配置 *)
```

```
db.conn('mydb','DriverID=MSSQL;Server=127.0.0.1,1433;Database=test;User_name=sa;Password=;');
db.conn('mydb','DriverID=MSSQL;Server=.;Database=test;OSAuthent=Yes;');

(* 连接 mysql 数据库的连接字符串 *)
db.conn('mydb','DriverID=MySQL;Database=mysql;Server=127.0.0.1;Port=3306;User_name=root;');

(* 连接 ORACLE 数据库的连接字符串 *)
db.conn('mydb','DriverID=Ora;Database=OraSrv:1521/orcl;User_name=ADDemo;Password=a;CharacterSet=UTF8;');

(* 连接 PostgreSQL 数据库的连接字符串 *)
db.conn('mydb','DriverID=PG;Server=127.0.0.1;Database=test;User_name=postgres;Password=123456;');

(* 连接 Firebird 数据库的连接字符串 *)
db.conn('mydb','DriverID=FB;Database=.\test.fdb;User_name=SYSDBA;Password=masterkey;');

(* 连接 IBM DB2 数据库的连接字符串 *)
db.conn('mydb','DriverID=DB2;Server=127.0.0.1;Database=test;User_name=db2admin;Password=;');

(* 连接 MongoDB 数据库的连接字符串 *)
db.conn('mydb','DriverID=Mongo;Server=127.0.0.1;Port=27017;Database=test;');

(* 连接 Sybase SQL Anywhere 数据库的连接字符串 *)
db.conn('mydb','DriverID=ASA;Server=127.0.0.1;Database=test;User_name=sa;Password=;');

(* 连接 InterBase 数据库的连接字符串 *)
db.conn('mydb','DriverID=IB;Server=127.0.0.1;Database=.\test.gdb;User_name=SYSDBA;Password=masterkey;');

(* 连接 Informix 数据库的连接字符串 *)
db.conn('mydb','DriverID=INFX;Server=127.0.0.1;Database=test;User_name=informix;Password=;');

(* 连接 ODBC 通用数据源的连接字符串 *)
db.conn('mydb','DriverID=ODBC;DSN=MyDSN;User_name=sa;Password=;');

// 连接池相关配置, 附加在上方各连接字符串末尾
// 完整示例:
db.conn('mydb','DriverID=MSSQL;Server=127.0.0.1,1433;Database=test;User_name=sa;Password=;Pooled=True;POOL_CleanupTimeout=30000;POOL_ExpireTimeout=90000;POOL_MaximumItems=200;')
// Pooled=True - 开启连接池
// POOL_CleanupTimeout - 清理超时(毫秒)
```

```
// POOL_ExpireTimeout - 连接过期时间(毫秒)
// POOL_MaximumItems - 池最大连接数
```

注意：连接前需确保数据库服务已启动，连接字符串中的参数（如用户名、密码、数据库名）需与实际环境一致。

4.2.2 核心操作函数

所有数据库统一使用两个核心函数：db.query（查询操作）、db.exec（增删改操作），返回JSON格式结果，便于处理。

```
// 配置数据库连接
include('/conn.cfg');

// 1. 查询操作（db.query(连接名, SQL语句, 参数1, 参数2, ... )）
// 注意：使用:p1~:p18作为参数占位符，避免SQL注入
var res := json.parse(db.query('mydb','SELECT * FROM tEmployee WHERE FID=:p1 and
FClose=:p2', 3, 0));

// 处理查询结果
if res.code = 200 then  // 200表示操作成功
begin
  writeln('查询到 ' + IntToStr(res.data.length()) + ' 条数据');
  // 遍历结果集
  for var i := 0 to res.data.length() - 1 do
  begin
    var user := res.data[i];
    writeln('用户名：' + user.FID+'|'+ user.FName + '，级别：' +
IntToStr(user.FLevel));
  end;
end
else
begin
  writeln('查询失败：' + res.msg); // res.msg为错误信息
end;

// 2. 增删改操作（db.exec(连接名, SQL语句, 参数1, 参数2, ... )）
// 新增数据
var addRes := json.parse(db.exec('mydb','INSERT INTO tEmployee(FName,FLevel)
VALUES(:p1,:p2)', '李四', 1));
if addRes.code = 200 then
  writeln('新增成功，影响行数：' + IntToStr(addRes.rowcount))
else
  writeln('新增失败：' + addRes.msg);

// 修改数据
var updateRes := db.exec('mydb','UPDATE tEmployee SET FLevel=:p2 WHERE FID=:p1',
3, 2);

// 删除数据
```

```
var deleteRes := db.exec('mydb','DELETE FROM tEmployee WHERE FID=:p1', 4);
```

4.2.3 数据库操作常见问题

- 问题1：连接失败 → 解决方案：检查数据库服务是否启动，连接字符串参数是否正确；
- 问题2：查询无结果 → 解决方案：检查SQL语句是否正确，参数占位符与参数数量是否匹配；
- 问题3：提示“SQL注入风险” → 解决方案：必须使用:p1~:p18占位符，禁止直接拼接SQL字符串。

4.3 计划任务调度

内置轻量级计划任务调度器，无需额外部署第三方服务，支持5种调度模式，脚本可直接在task目录下配置。

4.3.1 调度模式说明

- 模式0：按秒循环执行（格式：0,秒数），如0,10 → 每10秒执行一次；
- 模式1：固定时间执行（格式：1,年,月,日,时,分,秒），如1,2026,5,20,10,0,0 → 2026年5月20日10点执行；
- 模式2：每年执行（格式：2,月,日,时,分,秒），如2,5,20,10,0,0 → 每年5月20日10点执行；
- 模式3：每月执行（格式：3,日,时,分,秒），如3,20,10,0,0 → 每月20日10点执行；
- 模式4：每日执行（格式：4,时,分,秒），如4,0,0,0 → 每日凌晨0点执行。

4.3.2 计划任务配置步骤

1. 在task目录下新建脚本，后缀为.api（如backup.api）；
2. 在脚本头部添加<task>配置块，指定调度模式，然后编写任务逻辑：

```
<task>
{
      "title": "按秒执行任务",
      "mode": 0,
      "time": "10"
}
</task>
var s := 'WebPascal 5';
// 输出到日志，这里 writeln 会被忽略
logger.info('hello ' + s + ' 按秒执行任务');
```

3. 保存脚本（UTF-8编码），框架会自动识别task目录下的脚本，按配置的模式执行；
4. 调试说明：去掉<task>配置块，将脚本移动到script目录，可直接通过URL访问调试，调试完成后再移回task目录并添加配置块。

4.4 内置工具集使用

框架内置多种开发高频工具，一行代码即可调用，无需重复开发：

```
// 1. 验证码生成（返回可嵌入页面的图片资源）
// 第一个参数为验证码内容，第二个为品质（1-100）
writeln('<img src="' + captcha('1234', 80)+ '">');
// 2. 二维码生成
// 内容、品质、放大倍数
writeln('<img src="' + QRCode('https://webpascal.com', 80, 2) + '">');
// 3. 时间处理
var nowTime := FormatDateTime('YYYY-MM-DD HH:MM:SS', Now); // 格式化当前时间
var timeStamp := LocalDateTimeToUnixTime(now); // 本地时间转Unix时间戳
var localTime := UnixTimeToLocalDateTime(timeStamp); // Unix时间戳转本地时间
// 4. 雪花算法生成唯一ID
var uniqueId := Snowflake();
writeln('唯一ID: ' + uniqueId);
// 5. 容量格式化（自动区分GB/GiB）
var sizeStr := FmtSize(1024*1024*1024); // 1GB，输出"1.00 GB"
writeln('容量: ' + sizeStr);
```

4.5 文件上传操作

框架内置标准化/upload接口，支持大文件分块上传，自动按日期分目录存储，无需额外开发上传逻辑。

4.5.1 前端上传示例（HTML）

```
<!DOCTYPE html>
<html>
<body>
<form action="http://localhost:8833/upload" method="post" enctype="multipart/form-data">
  选择文件: <input type="file" name="file">
  <input type="submit" value="上传">
</form>
</body>
</html>
```

注意：上传路径默认在uploads目录，自动按日期分目录。

4.6 HTTP服务端属性

WebPascal 5 提供了访问 HTTP 请求信息的属性，通过 request 对象可以获取请求的各种参数。

```
// 基本属性
request.URL: 当前请求的URL路径，例值: /ask/http_request.api
request.RemoteIP: 客户端远程IP地址，例值: 192.168.0.1
request.Method: 请求方法，例值: GET
TStrings对象: request.Headers、request.Query、request.Params
request.Headers: 请求头信息
```

```
request.Query: URL查询参数
request.Params: 表单参数或请求体参数
获取方式（例如获取post参数id）：
request.Params['id']、request.Params[0]、request.Params.id
遍历方式: for i := 0 to request.Headers.length - 1 do writeln(...)
```

注意：属性名大写开头。

4.7 全局变量

WebPascal 5 提供了全局变量管理功能，用于在脚本间共享数据。所有全局变量保存在服务器内存中，重启服务后清空。

```
// 写入全局变量
WriteGlobalVar(name, value): 写入全局变量，支持字符串、数字、JSON字符串类型
读取全局变量
ReadGlobalVar(name): 读取全局变量，不存在返回null
ReadGlobalVarDef(name, default): 读取全局变量，不存在时返回默认值
// 删除与清理
DeleteGlobalVar(name): 删除指定的全局变量
CleanupGlobalVars: 清理所有全局变量
// 查询变量列表
GlobalVarsNames(pattern): 获取所有变量名列表，支持*通配符过滤
```

全局变量在服务重启后丢失，适用于临时数据存储、会话管理、缓存等场景。

4.8 文件管理

WebPascal 5 提供完整的文件和目录操作API，支持文件属性查询、哈希校验、复制删除、目录遍历等操作。

哈希校验文件

Hash.File(文件路径, 算法): 计算文件哈希值

支持的哈希算法:

sm3、sha224、sha256、sha384、sha512、sha3_224、sha3_256、sha3_384、sha3_512、sha1、md5

参数需要加h开头，如hMD5、hSHA1、hSHA256

```
// 文件属性查询
File.Exists(路径): 判断文件是否存在
File.Path(路径): 获取文件所在目录路径
File.FileName(路径): 获取文件名
File.CreateTime(路径): 获取文件创建时间
File.ModifyTime(路径): 获取文件修改时间
File.Size(路径): 获取文件大小，返回Int64
File.MIME(路径): 获取文件MIME类型
File.MIME2Ext(MIME): MIME类型转扩展名
// 文件复制与删除
File.Copy(源路径, 目标目录): 复制文件（只能复制到上传目录下）
File.Delete(路径): 删除文件（只能删除上传目录下的文件）
File.Delete只能删除文件上传目录下的文件，不能删除上传目录外的文件。
```

```
// 目录操作
Dir.Exists(路径)：判断目录是否存在
Dir.DirList(路径)：列出子目录
Dir.FileList(路径)：列出文件
Dir.CreateTime(路径)：获取目录创建时间
Dir.ModifyTime(路径)：获取目录修改时间
Dir.Size(路径)：获取目录大小，返回Int64
Dir.mkDir(路径)：创建多级目录
Dir.Delete(路径)：删除目录（只能删除上传目录下的目录，会清空目录下文件）
Dir.Delete只能删除文件上传目录下的目录，同时会清空目录下的文件。
```

4.9 服务统计探针

WebPascal 5.5 新增了以下服务级探针接口，服务于nginx负载均衡和健康检查。

/health 存活探针

访问/health接口，返回{"status": "ok"}，可用于nginx等软件的存活检测

/stats 服务统计探针

访问/stats接口，返回服务运行信息：工作时间、流量、连接数、请求和响应时间、CPU使用率、内存使用量

4.10 HTTP客户端详解

WebPascal 5 内置HTTP客户端，支持常用请求方法，示例如下：

```
// 文件属性查询
GET请求
http.get(url, header, params)：发起GET请求
POST表单
http.post(url, header, params)：POST表单数据
POST JSON
http.json(url, header, params)：POST JSON数据
PUT请求
http.put(url, header, params)：PUT请求
DELETE请求
http.delete(url, header, params)：DELETE请求
表单上传（支持文件）
http.formdata(url, header, params)：表单上传，支持文件
文件转存（下载到服务器）
http.getfile(url)：将远程文件转存到服务器的uploads目录，返回/up/路径
http.getfile将远程文件转存到服务器的uploads目录，返回/up/...路径。
带编码和超时的请求
http.get(url, header, params, reqEnc, resEnc, timeout)：支持指定请求编码、响应编码和超时时间编码参数使用UTF-8无BOM。
```

第五章 安全配置与防护

5.1 JWT身份认证

支持JWT令牌生成与验证，用于接口身份校验，防止未授权访问：

```
// 1. 生成JWT令牌（自定义密钥、有效期、扩展信息）
var token := jwt.encode('mySecretKey',{"userId":1, "userName":"张三"},7);
// 参数说明：密钥、扩展信息（JSON字符串）、有效期（天）
writeln('生成的令牌：' + token);
// 2. 验证JWT令牌
var verifyRes := jwt.verify('mySecretKey', token);
if verifyRes then
begin
  writeln('令牌验证成功');
  var jwtData:=json.parse(base64.decode(regex.match(b,'(?<=\.).*?(?=\.)')));
  writeln('<br>'+jwtData);
  writeln('<br>创建：'+formatdatetime('yyyy-MM-dd
hh:nn:ss.zzz',UnixTimeToLocalDateTime(jwtData.iat)));
  writeln('<br>过期：'+formatdatetime('yyyy-MM-dd
hh:nn:ss.zzz',UnixTimeToLocalDateTime(jwtData.exp)));
  if UnixTime>jwtData.exp then
  begin
    writeln('<br><font color=red>已超时！</font>');
    exit;
  end;
end
else
  writeln('令牌验证失败！');
```

建议：令牌有效期根据实际需求设置（如登录接口设置7天）。

5.2 数据加密与签名

集成多种加密算法，用于敏感数据加密、签名验证：

```
// 1. 哈希加密（MD5、SHA256等，用于密码加密）
var md5Str := hash.hash('123456',hMD5);
var sha256Str := hash.hash('123456',hSHA256);
writeln('MD5加密：' + md5Str);
writeln('SHA256加密：' + sha256Str);
// 2. 对称加密（AES，用于敏感数据传输）
var key := '1234567890abcdef'; // 密钥（AES-128需16位，AES-256需32位）
var plainText := '敏感数据123';
// 加密
var encryptStr := aes.encode(plainText, key, 'CBC'); // 加密模式CBC/ECB
// 解密
var decryptStr := aes.decode(encryptStr, key, 'CBC');
writeln('加密后：' + encryptStr);
writeln('解密后：' + decryptStr);
```

哈希算法：

sm3、sha224、sha256、sha384、sah512、sha3_224、sha3_256、sha3_384、sha3_512、sha1、md5，参数需要加h开头，比如hSHA256；

对称加密：AES、DES、3DES、SM4，支持 ECB/CBC 多种加密模式。

5.3 防注入与异常防护

- SQL防注入：所有数据库操作必须使用:p1~:p18参数占位符，框架自动区分参数类型，从源头避免注入风险；
- 异常防护：脚本执行出错时，框架会返回标准化JSON错误信息，包含行号、列号，便于调试，同时避免敏感信息泄露：

```
{
    "code": 500,
    "msg": "Script error: Unexpected \"=\" [line 42, column 4]"
}
```

- 编码安全：所有脚本文件必须采用UTF-8编码，接口数据传输默认UTF-8，避免编码乱码与漏洞。

第六章 日志与调试

6.1 日志系统使用

框架内置全场景日志系统，自动按日期生成日志文件，存放在log目录，支持5个日志级别，无需手动管理：

```
// 不同级别日志输出（按优先级从低到高：debug < info < warn < error < fatal）
logger.debug('调试信息：用户访问了test接口'); // 调试用
logger.info('普通信息：用户登录成功，ID=1'); // 常规操作记录
logger.warn('警告信息：文件大小超出预期'); // 非错误但需关注的信息
logger.error('错误信息：数据库查询失败'); // 操作失败，需排查
logger.fatal('致命信息：框架启动失败'); // 严重错误，影响服务运行
```

日志管理：日志文件自动按日期命名（如20260316.log），30天前的日志会自动清理，无需手动删除（可在SrvConfig.ini中修改MaxDays来设置天数）。

6.2 调试方法

- 脚本调试：将脚本放在script目录，通过URL访问，查看页面输出或日志文件，定位错误；
- 错误定位：根据控制台或日志中的错误信息（行号、列号），找到脚本中的问题代码；
- 调试技巧：使用logger.debug()输出变量值，查看数据是否符合预期；将复杂逻辑拆分，逐步调试。

第七章 版本与升级

7.1 版本说明

WebPascal 5 提供两个版本，适配不同使用场景，版本功能完全一致，仅授权与服务不同：

- 免费版：永久免费，无功能限制，适用于个人开发、学习测试、小型项目；支持免费版本更新，提供官方教程、演示案例；

- 商业版：一次性购买源码，终身使用，适用于企业开发、商业项目；包含Delphi完整源码，支持二次开发；提供永久技术支持与版本升级。

7.2 版本升级步骤

1. 从官方地址下载最新版本的压缩包；
2. 停止当前框架服务（关闭控制台或停止Windows服务）；
3. 备份原有script、task、uploads目录和数据库及配置文件，避免数据丢失；
4. 解压新版本压缩包，覆盖原有WebPascal.exe即可；
5. 启动服务，验证升级是否成功（访问原有接口，确认功能正常）。

第八章 常见问题与解决方案

8.1 问题排查

以下汇总了开发中常见的问题及对应解决方案：

问题现象	可能原因	解决方案
脚本无法访问，提示“404 Not Found”	脚本路径错误、后缀不正确（也支持无后缀）、编码不是UTF-8	确认脚本在script目录，后缀与访问地址相同，编码为UTF-8无BOM
无返回值，提示“200 OK”，StatusCode=200	脚本未输出任何内容、脚本非语法格式错误	做内容输出，或用try..except..end;捕获异常
数据库连接失败，提示“无法连接服务器”	数据库服务未启动、连接字符串参数错误	启动数据库服务，检查连接字符串中的主机、端口、用户名、密码是否正确
计划任务不执行	脚本不在task目录、<task>配置块格式错误、服务未启动	将脚本移至task目录，检查配置块格式，确保框架服务正常运行
文件上传失败	上传路径无权限、文件过大、前端enctype属性错误	给uploads目录添加写入权限，检查文件大小，前端form标签添加enctype="multipart/form-data"
脚本执行报错，提示“未定义标识符”	语法错误、函数名拼写错误、未引入相关脚本	检查脚本语法，确认函数名正确，必要时引入相关公共脚本

第九章 附录

9.1 常用API参考

9.1.1 基础API

- writeln(内容)：输出内容到页面，支持HTML、JSON、文本；
- include(脚本路径)：引入其他脚本文件；
- logger.级别(内容)：输出日志，级别为debug/info/warn/error/fatal。

9.1.2 数据库API

- db.conn(连接名，连接配置字符串)：连接数据库；
- db.query(连接名，SQL语句，参数...)：查询操作，返回JSON结果，有data；
- db.exec(连接名，SQL语句，参数...)：增删改操作，返回JSON结果，无data，有rowcount。

9.1.3 工具类API

- captcha(内容，品质)：生成验证码；
- QRCode(内容，品质，放大倍数)：生成二维码；
- FmtDecimal(字节数)：存储、传输容量格式化，GB=1000M；
- FmtSize(字节数)：内存、文件容量格式化，GB=1024M；
- Snowflake(数据中心，机位)：生成分布式唯一ID，参数分别为0..31值。

9.1.4 HTTP客户端API

- http.get(url, header, params, reqEnc, resEnc, timeout)：GET请求
- http.post(url, header, params, reqEnc, resEnc, timeout)：POST表单
- http.json(url, header, params, reqEnc, resEnc, timeout)：POST JSON
- http.put(url, header, params, reqEnc, resEnc, timeout)：PUT请求
- http.delete(url, header, params, reqEnc, resEnc, timeout)：DELETE请求
- http.formdata(url, header, params)：表单上传（支持文件）
- http.getfile(url)：将远程文件转存到服务器的uploads目录，返回/up/路径

9.1.5 全局变量API

- WriteGlobalVar(name, value)：写入全局变量
- ReadGlobalVar(name)：读取全局变量
- ReadGlobalVarDef(name, def)：读取全局变量，不存在返回默认值
- DeleteGlobalVar(name)：删除全局变量
- GlobalVarsNames(pattern)：获取变量名列表（支持*通配符）
- CleanupGlobalVars：清理所有全局变量

9.1.6 文件管理API

- Hash.File(文件路径，算法)：计算文件哈希值（如hMD5、hSHA1、hSHA256）
- File.Exists(v)：判断文件是否存在

- File.Path(v)：获取文件所在目录路径
- File.FileName(v)：获取文件名
- File.CreateTime(v)：获取文件创建时间
- File.ModifyTime(v)：获取文件修改时间
- File.Size(v)：获取文件大小（Int64）
- File.MIME(v)：获取文件MIME类型
- File.MIME2Ext(v)：MIME类型转扩展名
- File.Copy(源路径, 目标目录)：复制文件
- File.Delete(v)：删除文件
- Dir.Exists(v)：判断目录是否存在
- Dir.DirList(v)：列出子目录
- Dir.FileList(v)：列出文件
- Dir.CreateTime(v)：获取目录创建时间
- Dir.ModifyTime(v)：获取目录修改时间
- Dir.Size(v)：获取目录大小（Int64）
- Dir.mkDir(v)：创建多级目录
- Dir.Delete(v)：删除目录

9.1.7 其他API

- GetPort()：获取当前服务端口

9.2 官方资源

- 官方地址：https://webpascal.com
- 咨询邮箱：593106@qq.com
- 配套资源：开发文档、演示案例、前端模板（可从官方地址下载）。

Copyright © 2026 WebPascal.com 版权所有 | 专为Delphi开发者打造