# 需求分析篇

最近二十年来，应用系统的开发经历了几次大的变迁，其中最具有革命性的有两次，一次是从C/S向B/S的变迁，引爆了第一次互联网浪潮；另一次是桌面互联网向移动互联网的变迁，引爆了第二次移动互联网时代。

遗憾的是，在技术进化不可逆转的大潮流下，开发效率并未得到相应的提升，尤其是在B/S的开发过程中，缺少在C/S开发阶段那些所见即所得的开发工具，使得开发工作量和开发成本都成倍增加，这些也给系统的后续运维带来很大的困难。

在通常的前后端分离设计过程中，后端往往退化为纯粹的数据访问接口，为前端提供数据，这种情况下缺点就是后端接口会随业务扩展而不断膨胀，我曾经见过的系统，后端的接口甚至达到了1万多个，这么庞大的后台接口，对于系统的后续升级运维，都是极大的人力负担。

WebDW产品的设计，也是为了简化后台接口结构，减少后台暴露出来的接口数量。

我曾经在PowerBuilder上下过很大功夫，也做过很多项目，后来转做Web系统，深感实在是没有类似PowerBuilder里面DataWindow那么好用的工具，因此产生了把DataWindow移植到其他语言进行实现使用的想法，这就是整个WebDW产品的总体思路来源。我们做这个产品，也是向PowerBuilder这个伟大的产品做一点致敬，表达我们的尊敬之情。

# 技术架构篇

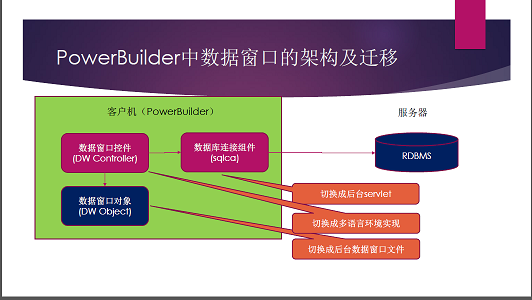
## DataWindow技术简介

从技术角度来说，WebDW是对DataWindow技术的升级实现，把DataWindow这项技术从局限于PowerBuilder语言环境，扩展到了其他开发语言和运行环境。

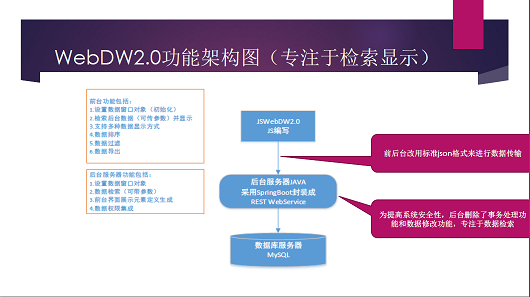
DataWindow是PowerBuilder的一项专用技术，PB利用DataWindow技术来实现数据的可视化展示和可视化操作，而无须手工编写过多的代码。

在PowerBuilder中，DataWindow控件负责完成SQL语句的自动生成，并将后台数据自动编译为特定界面进行数据，界面元素支持多种显示编辑风格，常用的有：标签、文本框、下拉框、单选钮、复选钮等。

## DataWindow技术的迁移

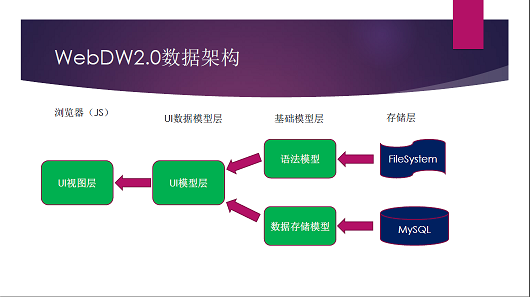


通过分析DataWindow的基础技术，我们把整个DataWindow的技术分解为三部分，然后将这些技术组件在后端用JAVA重新继续实现。



## WebDW技术架构

在WebDW产品架构中，整体产品的架构分解为：UI展示层、UI模型层、数据接口层三部分，其中数据接口层和UI模型层运行在后端服务器上，而前端UI展示层获取到一个标准的UI界面定义，从而在不同编程语言层次上得到一个统一的后台接口和统一的界面描述，前端仅负责渲染展示，而整个界面的定义和描述则在后端完成。



考虑到多种客户端的同时支持，WebDW的整体技术架构如下图所示。



# 产品介绍篇

## Webdw服务器产品(共一种)

### 概述

WebDW服务器是WebDW产品的核心产品，所有前端各项产品都需要通过调用后端服务器的接口来实际获取数据，然后才能进行界面渲染。

WebDW服务器是在后台服务容器里面运行的数据窗口控件，它可以通过加载数据窗口对象来进行功能的扩展。加载不同的数据窗口对象就可以将访问接口对应到不同的数据表，对应于不同的业务逻辑功能。

WebDW服务器是整体技术架构上的业务逻辑层，处于界面渲染层和数据存储层之间进行交互转换。

### Web服务器访问API

详见相关API文档说明。

### Ｗeb服务器技术选型

开发语言选用JAVA语言开发，开发框架选用SpringBoot作为基础框架。

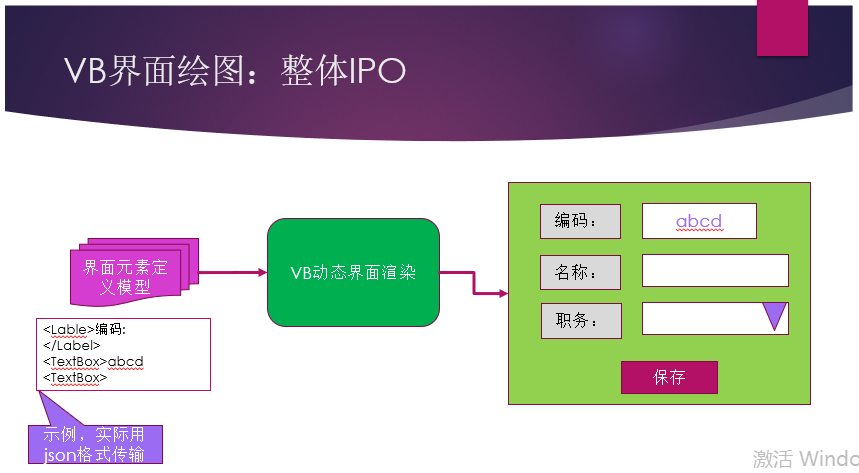
## Webdw客户端产品(共12种)

### 概述

在整体技术架构中，客户端负责的是界面的渲染，从服务器端调用以后，会返回一个标准的界面描述字符串，然后各个客户端分别按照自己的编程语言规范来进行界面渲染展示。

整体上的客户端要完成三个大的功能：

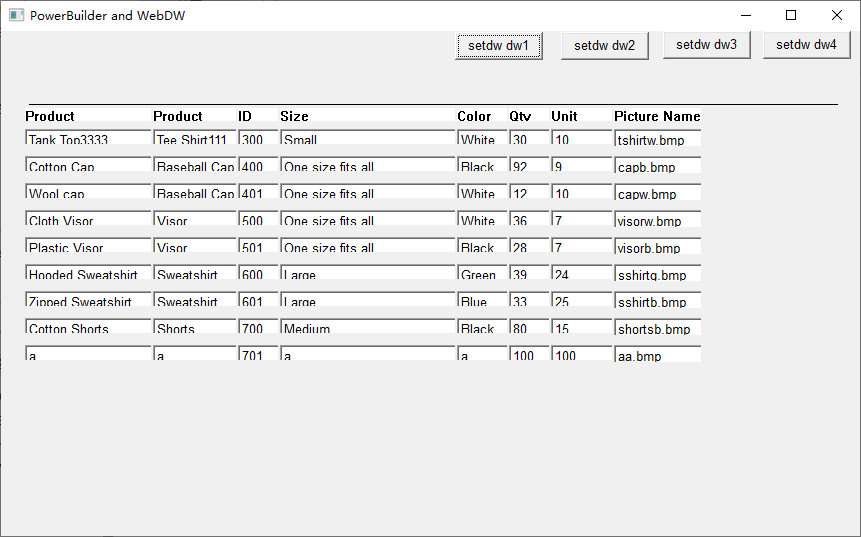
* 后台服务器API调用，完成Ajax调用，返回界面描述字符串，格式为json。
* 客户端JSON格式解析，将JSON格式的字符串解析为标准的界面定义元素。
* 界面渲染，将界面定义元素按照不同类型的分类，分别渲染成标签、文本框、下拉框、单选钮、复选钮。



### C/S开发(共8种)

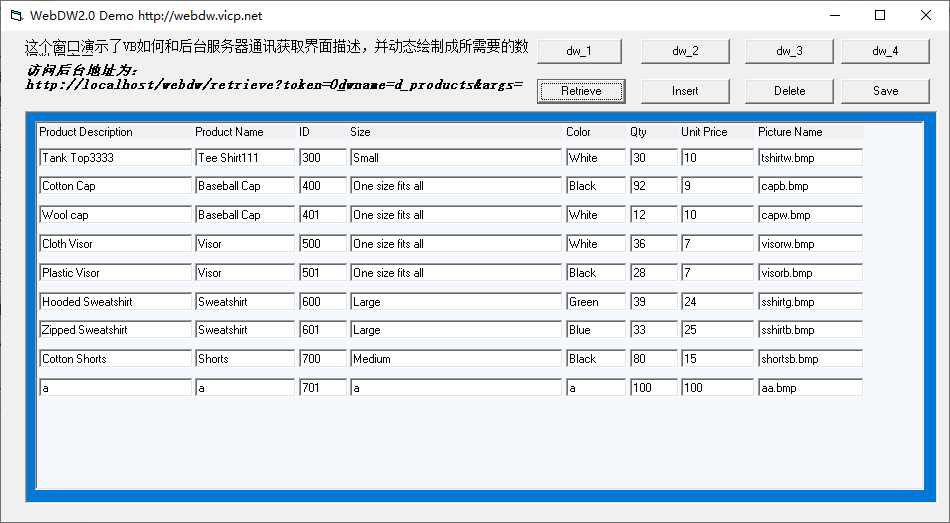
#### PowerBuilder

界面示例如下：



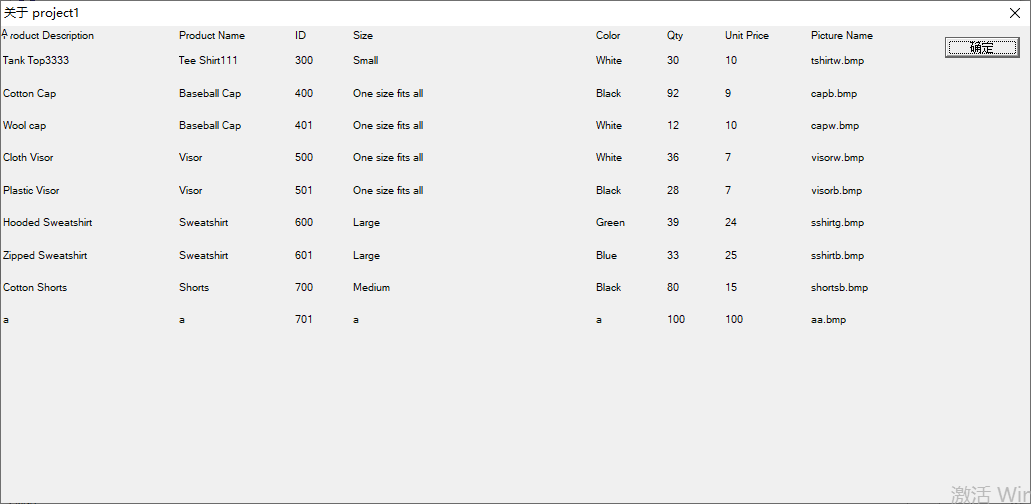
#### VisualBasic

界面示例如下：



#### VisualC++

VC界面示例如下：



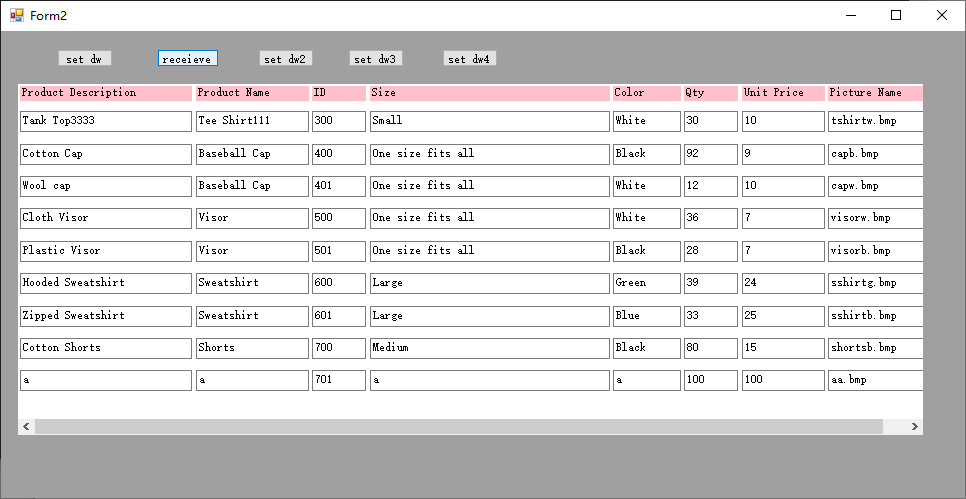
#### VB.net

VB.net界面示例如下：



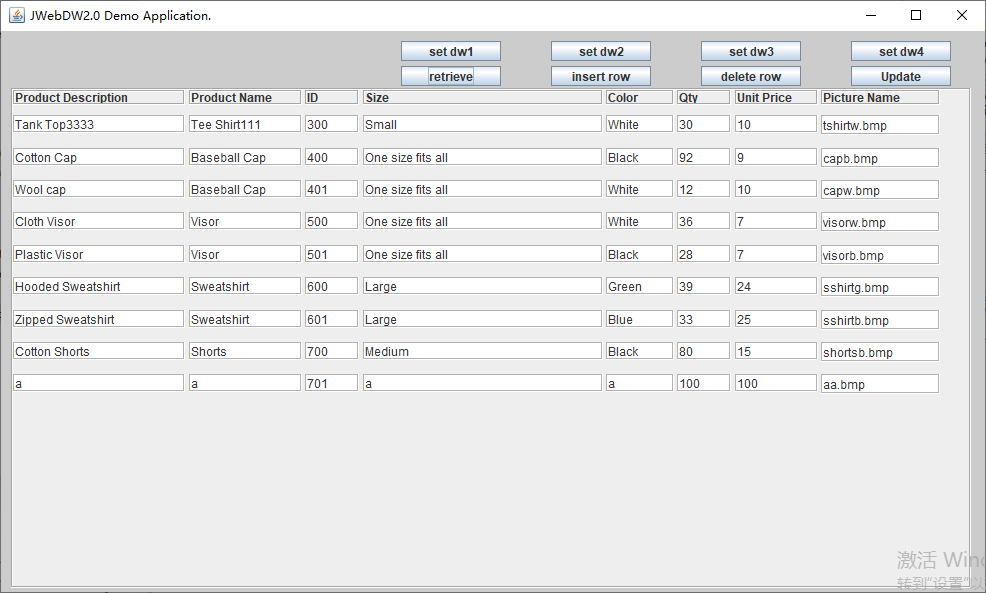
#### C#

C#客户端示例如下：



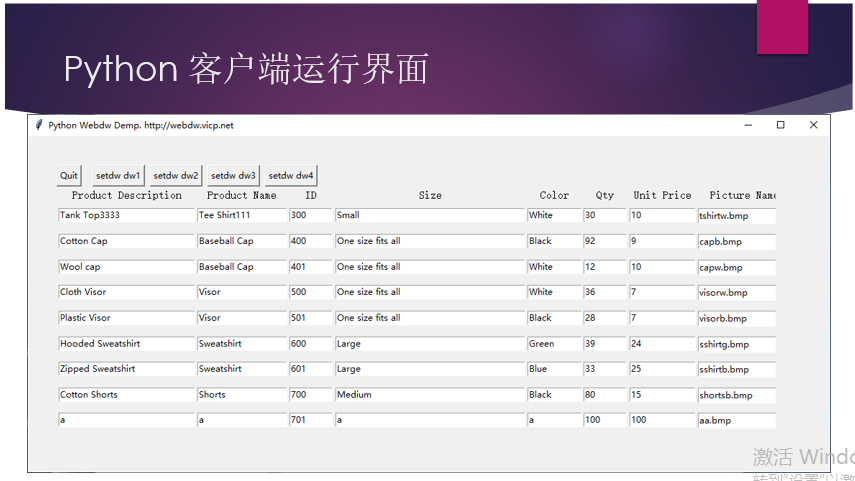
#### JAVA

JAVA客户端示例如下：



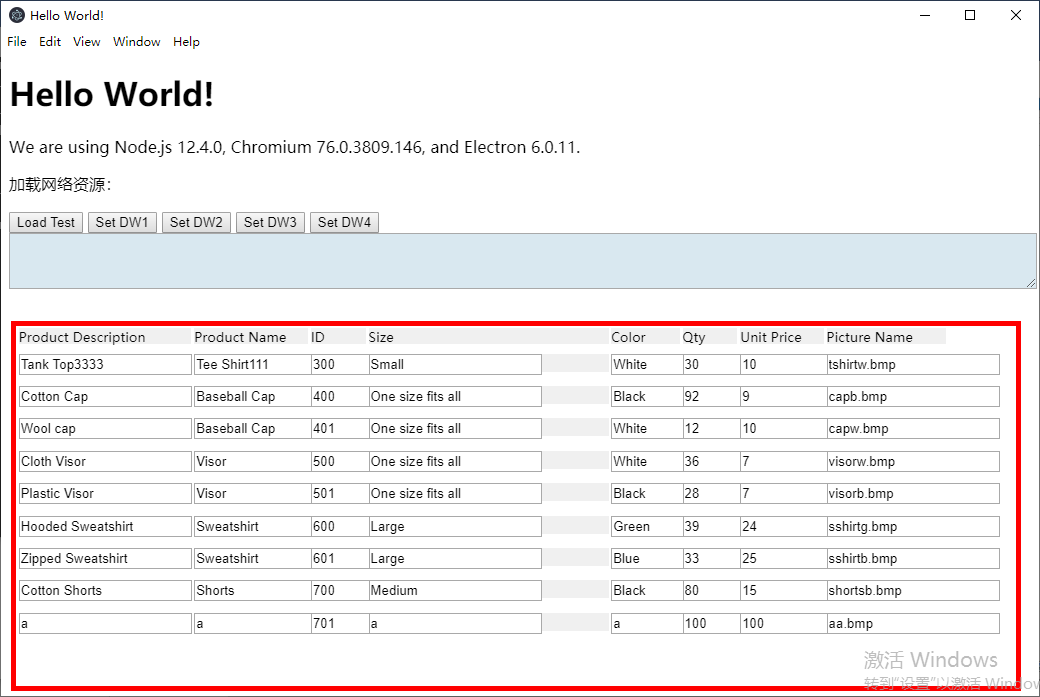
#### Python

Python客户端界面示例如下：



#### Electron

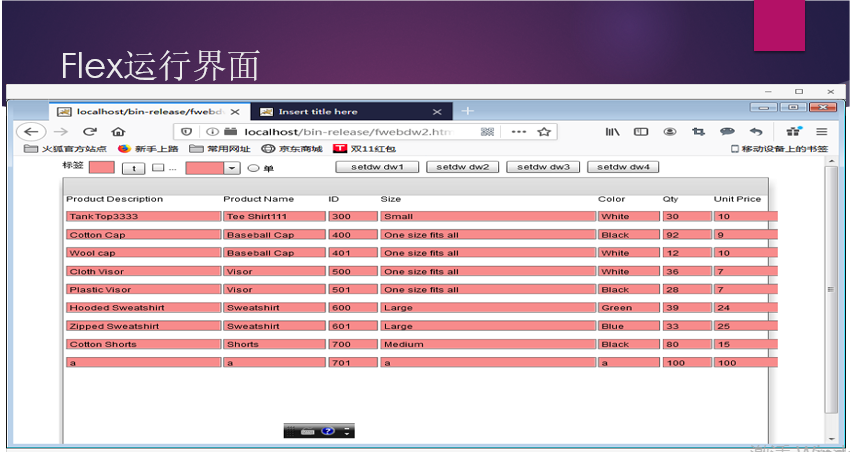
Electon客户端界面示例如下：



### B/S开发(共3种)

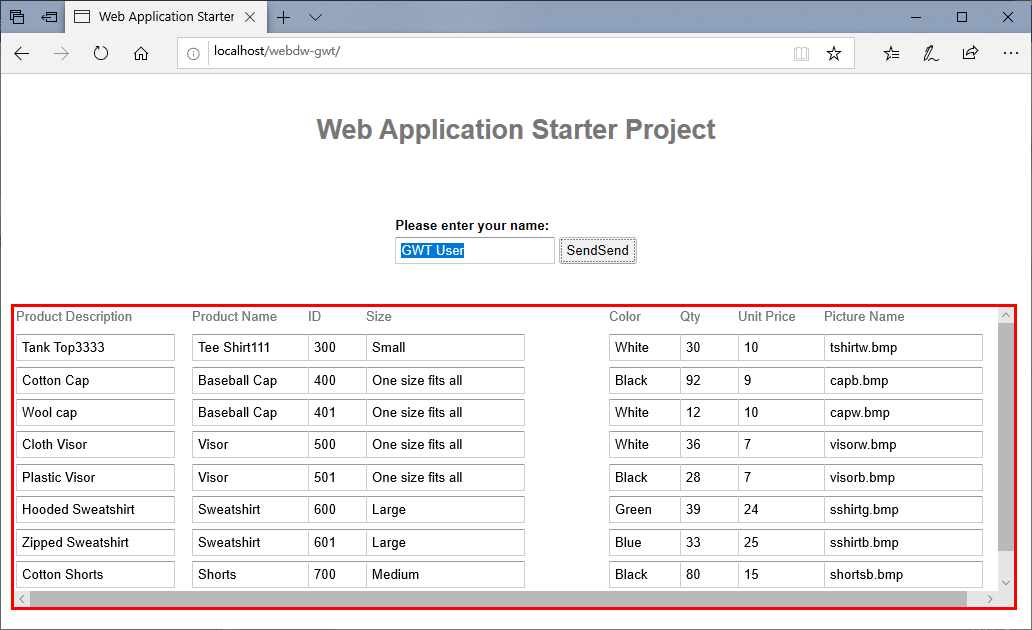
#### Flex

Flex客户端演示界面如下：



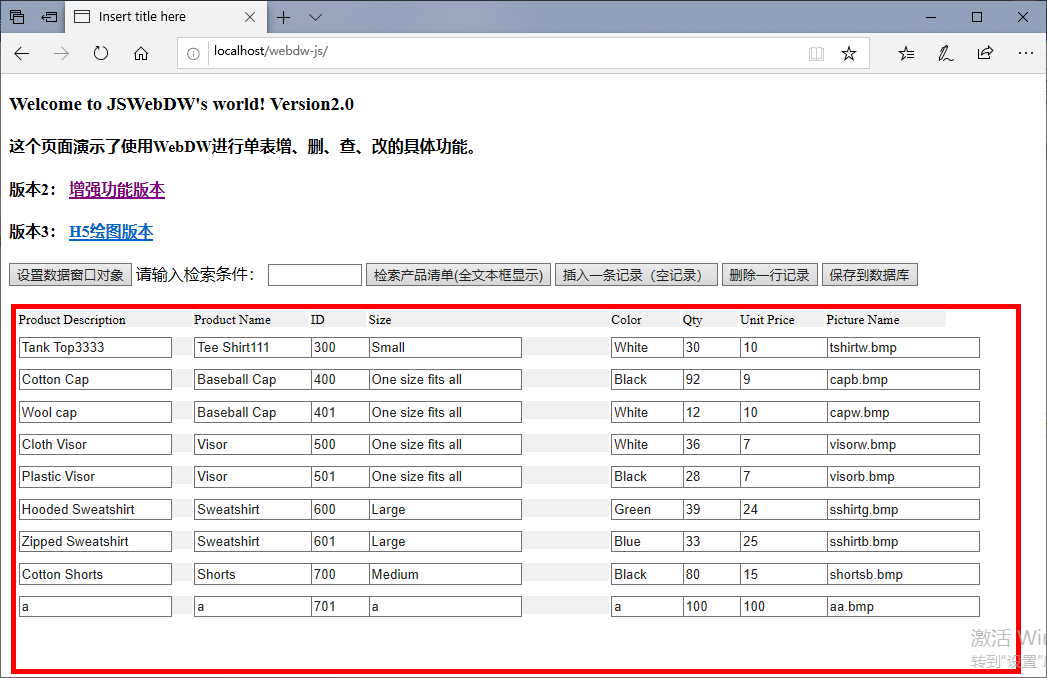
#### GWT

GWT的演示界面如下：



#### JavaScript

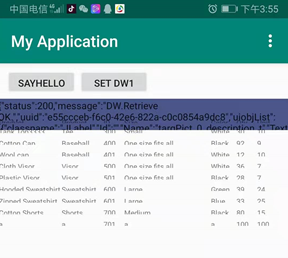
JavaScript版本客户端界面示例如下：



### Mobile开发(共1种)

#### Android

Android版本的客户端界面示例如下：



# 使用方法篇

## 服务器产品使用方法

### 数据库初始化

服务器采用JDBC方式连接数据库，默认选择使用MySQL数据库。数据库初始化步骤如下：

* 安装MySQL数据库
* 创建空白数据库webdw2.0
* 执行src\dbsql\目录下的webdw2.0.sql语句，创建WebDW后台所需要的数据表，以及相关的演示数据表。

### 服务器安装

服务器安装步骤如下：

* 安装JDK,要求JDK8以上版本，低版本JDK没有经过测试
* 安装Apache Tomcat，要求8以上版本，低版本Tomcat未经过测试
* 修改Tomcat配置文件，监听到80端口（默认为8080端口）
* 将下载得到的webdw.war文件复制到Tomcat安装目录下webapps目录下
* 启动Tomcat，会自行进行war包解压

### 服务器数据库连接配置修改

修改war包内部的application.properties文件，以适合对应的数据库连接。

### 服务器启动

执行Tomcat安装目录下bin目录下的startup.bat命令，检查日志和控制台输出确认服务器启动成功。

## 客户端产品使用方法

下载对应的客户端版本来进行具体使用。如果是可执行文件，可以直接点击运行；如果是脚本，则需要调用编译器来执行；如果是网页形式，则需要发布到Tomcat上再通过网页来访问。

不同版本的客户端开发时，需要参考标准的客户端DEMO程序源代码来进行开发，可以直接把这些代码集成进行项目里，也可以根据项目实际情况，对客户端代码进行封装以后再进行调用。

# 产品下载

服务器端产品下载（war包格式）

|  |  |  |  |
| --- | --- | --- | --- |
| 应用平台 | 开发语言 | 下载文件包 | 说明 |
| 应用服务器 | JAVA |  |  |
|  |  |  |  |

客户端产品下载(RAR格式)

|  |  |  |  |
| --- | --- | --- | --- |
| 应用平台 | 开发语言 | 可执行文件包 | 源代码项目包 |
| C/S | PowerBuilder |  |  |
| VisualBasic |  |  |
| VisualC++ |  |  |
| VB.net |  |  |
| C# |  |  |
| JAVA |  |  |
| Python |  |  |
| Electron |  |  |
| B/S | Flex |  |  |
| GWT |  |  |
| JavaScript |  |  |
| Mobile | Android |  |  |

# 产品报价

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 产品/服务名称 | 基准报价 | 备注 |
| 1 | 服务器端产品 | RMB 5万 | 含自选一个赠送的客户端产品，含一年内服务器免费升级费用 |
| 2 | 客户端产品 | RMB 1万 | 任何一个语言的客户端，报价为RMB 1万元，含一年内服务升级 |
| 3 | 服务器产品升级 | RMB 1万/年 | 选用标准产品服务的，1年内免费服务器产品升级 |
| 4 | 定制功能开发 | RMB 1000/人天 | 按照具体功能评估工作量，基于现有产品提供定制扩展开发，报价为每人天1000元。（驻场费用另算） |
| 5 | 项目定制开发 | RMB 3万/人月 | 按照项目功能范围评估工作量计算费用，基于现有产品提供定制开发，报价为每人月3万元。（含驻场费用） |