<AppML> 教程

<AppML>

快速和简单的 Web 开发

什么是 <AppML>?

<AppML> appML是一个为web应用程序设计的HTML扩展框:

- XML 语言定义了应用的模型
- JavaScript 运行于客户端浏览器上
- PHP 或者 ASP 脚本运行于服务器上

AppML 是应用模式语言（ **A**pplication **M**odeling **L**anguage）。

学习非常简单

- 超级简单的模型
- 超级简单的属性
- 超级简单的应用开发
- 运行于任何平台及任何浏览器
- 安装简单

只有 HTML, JavaScript, 和 XML

<AppML> 只需要在HTML页面中包含 JavaScript，然后再服务端上存储 XML 数据：

HTML 页面:

```
<h1>My First Web Application</h1>

<div id="Place01">
<table id="Template01" class="appmltable">
<tr>
  <th>Customer</th>
  <th>City</th>
 <th>Country</th>
</tr>
<tr id="appml_row">
  <td>#CustomerName#</td>
 <td>#City#</td>
  <td>#Country#</td>
</tr>
</table>
</div>

<script src="appml.js"></script>
<script>
app=new AppML("appml.htmlx","Models/Customers");
app.run("Place01","Template01");
</script>
```

XML :

```
<appml>

<datasource>
<database>
  <connection>Demo</connection>
  <sql>SELECT CustomerName,City,Country FROM Customers</sql>
</database>
</datasource>

</appml>
```

尝试一下 »

如果你之前已经学习了web开发，你会发现 <AppML> 使用非常简单。

如果你之前学习过 PHP, ASP, 或者 ASP.NET的开发, 你会清楚的看到使用 <AppML> 的好处。

现代 Web 架构

<AppML> 是一个结合了最新的技术和现代web开发想法，使用低消耗高速简单的架构：

- 采用 MVC 架构
- 极低的带宽消耗
- 云计算的优化
- 内容完全分离
- 智能，灵活，快速的Web开发
- 高度的可扩展性和可测试性
- 简单的配置和重新配置
- 智能支持用户账号和角色

<Appml> 历史

1999年，Refsnes Data公司开始研发AppML,一种基于XML、用来定义Internet应用程序的语言。2000年9月，一个为了挪威手球联盟而进行的大型项目开始，其目的是想通过仅仅使用AppML将一个巨大的信息系统从旧的DOS环境转换到现代的Internet上。而这个主要的项目刚刚取得了巨大的成功。据开发者估计，和传统的Web开发相比，这次的开发时间被缩短了高达75%。

在2007年九月中旬，AppML内容实现脱机使用，因为它支持ASP和IE。

在2013年十月，AppML宣布，作为一个开放源代码的产品，在PHP、ASP.NET版本中兼容 所有的浏览器。

AppML在1999年最初的设计目标:

- AppML 应用必须运行于是有网络
- AppML 应用应具有平台独立性
- AppML 应用必须使用互联网标准 (HTTP, HTML, CSS, XML)
- AppML 应用必须支持各种应用的需求
- AppML 应用必需是自描述
- AppML 应用程序必须易于开发，维护和修改
- AppML 应用程序必须面向未来

如何使用 <AppML>

本章节将通过以下4个简单的步骤演示如何创建<AppML>的应用程序。

下一章将介绍如何下载<AppML>，并开始在你自己的电脑上开发Web应用程序。

1.创建模型（Model）

使用以下内容创建文件

```
<appml>

<datasource>
<database>
  <connection>Demo</connection>
  <sql>SELECT CustomerName,ContactName,City,Country FROM Customers</sql>
  <orderby>CustomerName</orderby>
</database>
</datasource>

<filters>
  <query>
  <field>CustomerName</field>
  </query>
</filters>

</appml>
```

在子目录中Models（我们建议）将该文件保存为 Customers.xml。

模型解析

<appml> 标签定义了模型。

<datasource>标签定义模型的数据源。

<database>标签定义数据库。

<connection>标签定义数据库的链接。

<sql>标签定义数据查询

<orderby>标签定义默认排序。

<query>标签定义合法的查询过滤器。

2. 创建 WEB 页面

在第一个 <AppML> app中,创建一个 HTML页面:

实例

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Application</h1>

<table>
<tr>
```

```
  <th>Customer</th>
  <th>City</th>
  <th>Country</th>
</tr>
<tr>
  <td>Alfreds Futterkiste</td>
  <td>Berlin</td>
  <td>Germany</td>
</tr>
</table>

</body>
</html>
```

尝试一下 »

3. 添加样式

在你的web页面中添加层叠样式在执行e <AppML> app:

实例

```
<!DOCTYPE html>
<html>

<head>
<link rel="stylesheet" href="appml.css">
</head>

<body>
<h1>My First Web Application</h1>

<table class="appmltable">
<tr>
  <th>Customer</th>
  <th>City</th>
  <th>Country</th>
</tr>
<tr>
  <td>Alfreds Futterkiste</td>
  <td>Berlin</td>
  <td>Germany</td>
</tr>
</table>

</body>
</html>
```

尝试一下 »

4. 添加脚本, 然后执行应用

在你的web页面中添加脚本来运行 <AppML> app:

实例

```
<!DOCTYPE html>
<html>

<head>
<link rel="stylesheet" href="appml.css">
</head>

<body>
<h1>My First Web Application</h1>

<div id="Place01">
```

```
<table id="Template01" class="appmltable">
<tr>
  <th>Customer</th>
  <th>City</th>
  <th>Country</th>
</tr>
<tr id="appml_row">
  <td>#CustomerName#</td>
  <td>#City#</td>
  <td>#Country#</td>
</tr>
</table>
</div>

<script src="appml.js"></script>
<script>
app=new AppML("appml.htmlx","Models/Customers.xml");
app.run("Place01","Template01");
</script>

</body>
</html>
```

尝试一下 »

实例解析

<AppML> 库中含有大量的函数。这些函数可以再你的web页面中调用。

<script src="appml.js"> 加载了 <AppML> 库。

JavaScript 语句: **app=new AppML("appml.htmlx","Models/Customers.xml");** 创建 AppML 应用对象, 然后执行web服务端脚本 "appml.htmlx" 来加载 "Customers.xml"文件的数据。

JavaScript 语句 **app.run("Place01","Template01");** 将数据插入到 id="Place01" 的HTML元素中, 使用 id="Template01" 属性元素作为模板。

属性 **id="appml_row"** 定义了每条数据插入到HTML元素中。

标记中的数据会使用模型的数据替换。

以上所有，你可以想象到更快的原型吗？

它是如何工作的？

当web页面加载时, 你可以再页面中加载 <AppML> 控制器。

使用 <AppML> 控制器, 你可以再页面创建 <AppML> 对象。

当你在页面中运行 <AppML> 对象, 它会请求服务端数据控制器。

<AppML> 对象从服务器接收数据 (使用数据模型)。

<AppML> 对象 (或者你的代码) 在页面中显示数据。

(可选) web用户可以改变数据。

(可选) <AppML> 可以在服务器后台发送数据。

(可选) 服务器控制器可以在服务端存储数据。

典型的 Web 文件和文件夹：

web文件夹: **Demo**

数据文件夹: **Data**

图片文件夹: **Images**

模型文件夹: **Models**

应用: **Demo.htm**

样式: **Demo.css**

<AppML> 配置文件: **appml_config.php** (或者 .htmlx)

<AppML> 样式文件: **appml.css**

<AppML> 浏览器控制器: **appml.js**

<AppML> 服务器控制器: **appml.php** (or .htmlx)

没有限制

可以将 <AppML> 对象放在 HTML 页面。 <AppML> 不影响页面的其他部分。

<AppML> 在方案页面不存在时默认为标准的显示页面。 这是完美的快速原型。

但是 <AppML> 主要功能不是用于页面的显示。 <AppML> 主要是读取 应用程序数据. 它带来的数据可以通过自由的使用 HTML, CSS, 和 JavaScript 来设计它们的显示效果。你可以：

自己编写HTML，让AppML处理数据。

调用模型，并处理所有的显示。

使用AppML的属性和方法,创建其它的组合。

你很快会发现 <AppML>具备了强大的功能，它可以为你的web应用提供数据和数据模型。你可以：

为用户或用户组定义数据安全

连接所有类型数据库, 如 Access, MySQL, SQL, 和 Oracle

连接 XML 文件和 Text 文件

定义数据类型，数据格式，数据限制。

给模型添加任何新元素。

阅读<AppML> 参考手册

□ AppML 教程 AppML 下载 □

□ 如何使用 AppML AppML 架构 □

<AppML> 下载

本章节将介绍如何下载 <AppML>，下载后我们将立即开始在你的电脑上开发 web应用。

下载 <AppML>

<AppML> 不是一个产品。 <AppML> 是一个想法。 它仅仅是在浏览器上的脚本和服务器上的脚本。

任何人都可以下载 <AppML> ，下载后可以修改基本来创建我们自己的专业web应用。

你可以从下载使用于PHP的 <AppML> ： AppmlPHP.zip

本站关于appml的php实例采用链接数据库的方式为mysqli及mysqlnd
更多关于 php mysqli信息请查看：PHP 5 MySQLi 函数
更多关于 php mysqlnd信息请查看：mysqlnd介绍

你也可以下载适用于ASP .NET的 <AppML> ： AppmlASP.zip

ZIP 文件内容:

文件名	描述
appml.php (或者.htmlx)	<AppML> 服务端脚本
appml.css	<AppML>样式文件
appml.js	<AppML> 浏览器脚本
appml_config.php (or .htmlx)	<AppML> 本地配置
Images (文件夹)	<AppLM> 图片样式

如果你有自己的web服务器

如果你已经拥有支持 ASP.NET 或 PHP 的web服务器:

1. 创建一个文件夹并命名为Demo (或者其他)。
2. 从zip中解压文件和文件夹.
3. 拷贝文件很文件夹到你新的web文件夹中。
4. 开始开发你的应用

如果你没有web服务器

如果你没有web服务器，你可以使用 Microsoft's WebMatrix (免费软件) 来开发web应用

使用 WebMatrix, 你不需要web服务器就可以在你的电脑上编辑，测试，及执行web应用。

WebMatrix 自带功能:

- <AppML> 文件编辑器(HTML, CSS, 和 XML)
- 可以运行应用的web服务器 (IIS Express)
- 数据库服务器 (SQL Server Compact)
- 很好的支持服务端语言 (PHP, ASP, 和 ASP.NET)

你可以从以下地址下载WebMatrix：

http://www.microsoft.com/web/webmatrix

当你已经安装 WebMatrix, 就可以参照以上步骤使用Appml。

如果已经有数据库

通过改变 **appml.config**文件的配置信息，可以连接到你自己服务器上的数据库：

PHP MySQL:

```
<database name="demo">
  <host>127.0.0.1</host>
  <name>dbName</name>
  <user>dbUser</user>
  <password>dbPass</password>
```

```
</database>
```

ASP.NET SQL Server

```
<database name="demo">
<connection>
Provider=SQLOLEDB;data source=sName;Database=dbName;user id=dbUser;password=dbPass
</connection>
</database>
```

ASP.NET Access (完整路径)

```
<database name="demo">
<connection>
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:DemoDataDemo.mdb
</connection>
</database>
```

ASP.NET Access (虚拟路径;)

```
<database name="demo">
<connection>
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=#webroot#DataDemo.mdb
</connection>
</database>
```

如果你没有数据库

如果你没有数据库:

你可以使用 WebMatrix 创建一个数据库。

或者从以下链接下载 Access 数据库: Northwind.zip.
或者从以下链接下载 空的 Access 数据库: Database.zip

更多下载

你可以从本周下载更多应用实例教程:

PHP 版本

ASP.NET 版本

你可以下载完整的数据应用测试实例:

PHP 版本

ASP.NET 版本

□ 如何使用 AppML　　AppML 架构 □

<AppML> 架构

MVC 架构

<AppML> 采用了 MVC 架构。

MVC 全名是 Model View Controller，是模型（Model）－视图（View）－控制器（Controller）的缩写，一种软件设计典范。

模型(Model) 描述你的应用。

视图(View) 显示你的数据。

制器(Controller) 控制你的应用。

Wikipedia: Model, View, Controller

模型(MODEL) - 仅仅是一个简单的XML文件

模型描述了您的应用程序，并且可在不同的硬件和软件平台（PC、iPhone、Tablets 等）重复使用。它不关心用户界面（UI）或表现形式。

模型采用xml编写，存储于web服务器中。

```
<appml>

<datasource>
  <database>
    <connection>Northwind</connection>
    <sql>SELECT CustomerName,ContactName,City,Country FROM Customers</sql>
  </database>
</datasource>

<filters>
<query>
  <field label="Customer">CustomerName</field>
  <field>City</field>
  <field>Country</field>
</query>
<order>
  <field label="Customer">CustomerName</field>
  <field>City</field>
  <field>Country</field>
</order>
</filters>

</appml>
```

以上实例定义了数据源来自 Northwind 数据库。

该模型允许使用预定义的 SQL 获取数据。它还允许通过 Customer、City 和 Country 查询数据和排序。

视图（VIEW） - 仅仅是一个普通的 HTML 文件

视图即是 UI（User Interface：用户界面）。它通常是一个显示和输入数据（可选）HTML 页面：

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="appml.css" />
</head>
<body>
<h1>My First Web Application</h1>

<div id="Place01"></div>

<script src="appml.js"></script>

<script>

customers=new AppML("appml.html","Customers.xml");
```

```
customers.run("Place01");
</script>
</body>
</html>
```

以上 HTML 页面使用通过执行脚本语言创建一个 AppML 对象，并将数据显示在 id="Place01" 的 div 中。

采用了 "appml.js" 脚本文件。

CONTROLLER(控制器) - 仅仅是一个浏览器很服务端的脚本

服务端脚本通过以下方式控制应用：

从浏览器中接收请求数据

将模型和数据返回给浏览器

从浏览器中接收更新数据

在服务器上更新数据

数据通信过程，请进行数据安全验证。

浏览器脚本通过以下方式控制应用：

当页面加载时，你可以加载<AppML> 控制器到页面上。

使用控制器，你可以在页面创建 <AppML> 对象。

当执行 <AppML> 对象时, 它会向服务器请求数据。

<AppML> 对象从服务器接受数据（使用数据模型）。

<AppML> 对象（或者你的代码）在页面中显示你的数据。

（可选）web用户修改数据。

（可选）<AppML> 可以向服务器发送修改请求。

典型的 Web 文件和文件夹：

web文件夹：**Demo**
数据文件夹：**Data**
图片文件夹：**Images**
模型文件夹： **Models**
应用：**Demo.htm**
样式：**Demo.css**
<AppML> 配置文件：**appml_config.php** (或者 .htmlx)
<AppML> 样式文件：**appml.css**
<AppML> 浏览器控制器：**appml.js**
<AppML> 服务器控制器：**appml.php** (或者 .htmlx)

快速、灵活的应用开发

快速应用开发（Rapid Application Development、RAD）不仅是一种需求抽取方法，它还是是软件开发为一体的方法。快速应用开发目的是快速发布系统方案，而技术上的优美相对发布的速度来说是次要的。

<AppML> 提供超快速的原型设计，比传统的软件开发方法高100倍的速度。

应用程序原型可以直接从应用程序模型运行，无需任何编码。

Wikipedia: Rapid Application Development

敏捷软件开发是基于用户和开发者相互协作的基础上一步一步的基发展而来的方法。

<AppML> 应用从原型到完整的应用可以通过递增的方式一步步来编写实现。

Wikipedia: Agile Software Development

声明式编程

软件开发往往无法按照预期的时间和预算完成。软件编码错误也是经常出现。 这是因为计算机代码是很难开发、测试、维护。

编码已经**过时了**。你应该更多描述做什么，而不是如何实现它。

使用 <AppML> 你需要在模型中**声明**你的应用 。

使用 <AppML> 可以**少写或者不用编写代码**。

Wikipedia: Declarative Programming

代码先行（**Code First**）

Web应用程序开发可以使用以下两种不同的方式：

1. 代码先行（Code First）：使用预编程，预先测试的代码，只增加新的应用程序说明。

2. 契约优先（Contract First）：从头开始使用完整的应用程序的说明要求编写应用程序。

<AppML> 采用最合理的概念：**代码先行（Code First）**。

面向服务的体系结构（**service-oriented architecture，SOA**）

Web Service 是一个数据接口，通过URL指定，就像一个web页面。 但它有别于web页面，它只是一种传达信息的方式。

一个典型的 **Web Service** 为页面提供了数据。

使用 <AppML>，HTML 显示为用户界面，<AppML> 提供数据。

Original Web Services 设计使用了 XML 标志如 SOAP、WSDL 和 UDDI。

Modern Web Services 比如 <AppML> 应用更加简单。

> 更容易理解 - 可以被我们阅读
>
> 轻量级 - 没有不必要的代码或标记
>
> 易于实施 - 没有所需的开发工具

面向服务的体系结构（service-oriented architecture，SOA）是一个组件模型，它将应用程序的不同功能单元（称为服务）通过这些服务之间定义良好的接口和契约联系起来。接口是采用中立的方式进行定义的，它应该独立于实现服务的硬件平台、操作系统和编程语言。这使得构建在各种这样的系统中的服务可以以一种统一和通用的方式进行交互。

Web Services的优势

> Web services 只需要少量的代码
>
> Web services 被设计来处理一组有限的任务
>
> Web services 使用基于HTTP的通信协议
>
> Web services 独立于操作系统
>
> Web services 独立于编程语言
>
> Web services 可以连接不同的应用程序，系统和设备
>
> Web Services 可以很容易地发布信息
>
> Web Services 有利于快速应用程序开发

例如一个Web services，可以设计一个小程序，提供其他最新的股票的交易价格的应用程序。

Web services使用HTTP协议与其他系统进行通信，Web服务是独立于操作系统和编程语言。

调用Web services的应用程序将始终使用HTTP协议发送请求。调用应用程序将永远不会关心其他计算机运行的操作系统或编程语言。

Web services可以为更多的企业创造新的可能性，因为它提供了一种简单的分发大量信息的方式。

比如：航班时刻表和机票预定系统。

云计算**(Cloud Computing)**

云计算（Cloud Computing）是SOA的扩展：应用即服务（Application-as-a service），存储即服务（Storage-as-a-service），数据即服务（Data-as-a-service）。

对于大多数人，云计算是在web上存储数据：

> 邮寄很日历

文档和电子表格

书籍、笔记、待办事项列表

音乐、图片和电影

数据库和应用程序

原因很明显：

有来自世界各地的访问数据

与他人分享我的数据

硬件升级或者崩溃

<AppML> 可以很轻松地将数据库和应用程序放在云中。

AppML 下载 AppML 参考手册

AppML 架构 AppML 案例简介

<AppML> 参考手册

<AppML> 数据模型：

```
<appml security="security">

<datasource>
Datasource definition goes here
</datasource>

<filters>
Filter definitions goes here (if any)
</filters>

<update>
Update definitions goes here (if any)
</update>

<anything>
Anything you want to add to the model
</anything>

</appml>
```

<AppML>安全

<AppML> 安全通过 <AppML> 标签中的安全属性设置。

```
<appml security="artists">
```

以上应用开头包含了安全定义属性，只允许 artists 用户登陆。

在这种情况下，用户登录的用户名必须为 "artists"组的成员。

<datasource> 元素

<AppML>应用的的 <datasource> 元素定义了4个不同数据类型的:

子元素（只有一个可以适用）

元素	描述
<database>	定义数据类型
<xmlfile>	定义 XML 源文件
<csvfile>	定义一个逗号分隔的文本文件

<database> 元素

<database> 元素定义了数据库

子元素

元素	描述
<connection>	链接数据库名
<execute>	数据检索前执行的SQL语句（可选）
<sql>	用于检索数据的SQL语句
<maintable>	应用程序的主表（可选）
<keyfield>	主表的键字段（可选）

存储在SQL数据库中的数据

这是面向数据的应用程序最常用的解决方案。

```
<datasource>
<database>
<connection>CDDataBase</connection>
<sql>SELECT Artist, Title, Country FROM CD_Catalog</sql>
</database>
</datasource>
```

上面的模型可以从"CDDataBase"数据库的"CD_Catalog"表中选择三个数据选项 (Artist, Title, Country) 。

结果返回的行数是未知的。

存储在 XML 文件中的数据

<AppML> 可以从XML文件中读取数据:

实例

```
<appml>

<datasource>
<xmlfile src="cd_catalog.xml">

<record>CD</record>

<item>
<name>Title</name>
<nodename>TITLE</nodename>
</item>

<item>
<name>Artist</name>
<nodename>ARTIST</nodename>
```

```
</item>
<item>

<name>Country</name>
<nodename>COUNTRY</nodename>
</item>

</xmlfile>
</datasource>

</appml>
```

尝试一下 »

此方法能够将数据存储在服务器上的XML文件。

数据存储在 文本（Text）文件中

<AppML> 可以从文本文件中读取数据：

实例

```
<appml>

<datasource>
<csvfile src="cd_catalog.txt">

<item>
<name>Title</name>
<index>1</index>
</item>

<item>
<name>Artist</name>
<index>2</index>
</item>

<item>
<name>Price</name>
<index>5</index>
</item>

</csvfile>
</datasource>

</appml>
```

尝试一下 »

此方法可以在服务器上的将数据存储在文本文件中。

如果需要你可以创建数据库

<AppML> 如果有需要你可以创建一个数据库：

```
<database>
<connection>CDDataBase</connection>

<execute>
CREATE TABLE CD_catalog (
CD_Id INT IDENTITY,
Title NVARCHAR(255),
Artist NVARCHAR(255),
Country NVARCHAR(255),
Company NVARCHAR(255),
Price NUMBER,Published INT)
</execute>

</database>
```

完善快速原型模型！

点我分享笔记

反馈/建议

<AppML> 案例研究 - 简介

此案例研究演示了如何构建一个完整的 <AppML> 互联网应用程序，具有针对数据库中的若干表进行信息列举、编辑和搜索的功能。

我们将创建什么

我们将使用已有的数据库，名为 Demo。

对于数据库中的每个表，我们将：

- 创建原型模型
- 创建原型 HTML 页面
- 创建应用程序模型
- 创建 HTML 模板
- 创建完整的应用程序

Demo 数据库

对于案例研究，我们使用 MS Office 预装的 Northwind 数据库的一个子集。

这个数据库是众所周知的，不论是使用 PHP 还是经典 ASP 抑或是 ASP.NET，都易于测试。

Demo 数据库包含了若干带有数据的表，如下所示：

Customers

Customer	Address	City	Country
Around the Horn	120 Hanover Sq.	London	UK
Berglunds snabbkjøp	Berguvsvägen 8	Luleå	Sweden
Blauer See Delikatessen	Forsterstr. 57	Mannheim	Germany
Blondel père et fils	24, place Kléber	Strasbourg	France
Bólido Comidas preparadas	C/ Araquil, 67	Madrid	Spain
Bottom-Dollar Markets	23 Tsawassen Blvd.	Tsawassen	Canada

Suppliers

Supplier	Address	City	Country

Exotic Liquid	49 Gilbert St.	London	UK
New Orleans Cajun Delights	P.O. Box 78934	New Orleans	USA
Grandma Kelly's Homestead	707 Oxford Rd.	Ann Arbor	USA
Tokyo Traders	9-8 SekimaiMusashino-shi	Tokyo	Japan
Cooperativa de Quesos 'Las Cabras'	Calle del Rosal 4	Oviedo	Spain
Mayumi's	92 Setsuko Chuo-ku	Osaka	Japan
Pavlova, Ltd.	74 Rose St. Moonie Ponds	Melbourne	Australia
Specialty Biscuits, Ltd.	29 King's Way	Manchester	UK
PB Kn ckebr d AB	Kaloadagatan 13	G teborg	Sweden
Refrescos Americanas LTDA	Av. das Americanas 12.890	S o Paulo	Brazil

Products

Produc	QuantityPerUnit	UnitPrice	UnitsInStock
Chai	10 boxes x 20 bags	18	39
Chang	24 - 12 oz bottles	19	17
Aniseed Syrup	12 - 550 ml bottles	10	13
Chai	10 boxes x 20 bags	18	39
Chef Anton's Gumbo Mix	36 boxes	21	0
Grandma's Boysenberry Spread	12 - 8 oz jars	25	120
Uncle Bob's Organic Dried Pears	12 - 1 lb pkgs.	30	15
Northwoods Cranberry Sauce	12 - 12 oz jars	40	6
Mishi Kobe Niku	18 - 500 g pkgs.	97	29
Ikura	12 - 200 ml jars	31	31

□ AppML 参考手册 AppML 案例原型 □

<AppML> 案例研究 - 原型

此案例研究演示了如何构建一个完整的 <AppML> 互联网应用程序，具有针对数据库中的若干表进行信息列举、编辑和搜索的功能。

原型

在本章中，我们将为数据库中的每个表建立一个原型模型。

原型是非常便于使用的开发应用程序的起点。

原型模型

首先，为原型创建一个文件夹。该文件夹命名为 Prototypes。

然后，为数据库中的每个表创建一个原型模型。

使用 SELECT * from 每个表，并保存模型为 XML 文件：

模型：Proto_Customers.xml

```
<appml>
<datasource>
<database>
<connection>Demo</connection>
<sql>SELECT * FROM Customers</sql>
</database>
</datasource>
</appml>
```

模型：Proto_Suppliers.xml

```
<appml>
<datasource>
<database>
<connection>Demo</connection>
<sql>SELECT * FROM Suppliers</sql>
</database>
</datasource>
</appml>
```

模型：Proto_Products.xml

```
<appml>
<datasource>
<database>
<connection>Demo</connection>
<sql>SELECT * FROM Products</sql>
</database>
</datasource>
</appml>
```

原型视图

创建一个原型视图，把它保存为 Demo_Prototype.html，并尝试一下：

视图：Demo_Prototype.htm

```
<h1>Customers</h1>
<div id="List01"></div>

<script src="appml.js"></script>
<script>
```

```
customers=new AppML("appml.php","Prototypes/Customers");
customers.run("List01");
</script>
```

尝试一下 »

现在把所有的合并在一起

最后，通过少量 JavaScript 编码，为所有原型模型创建一个简单的原型页面：

Demo_Prototype_Views.htm

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="appml.css" />
</head>

<body>
<h1>Demo Applications</h1>

<button onclick='myOpen("Customers")'>Customers</button>
<button onclick='myOpen("Products")'>Products</button>
<button onclick='myOpen("Suppliers")'>Suppliers</button>
<button onclick='myOpen("Shippers")'>Shippers</button>
<button onclick='myOpen("Categories")'>Categories</button>
<button onclick='myOpen("Employees")'>Employees</button>
<button onclick='myOpen("Orders")'>Orders</button>
<button onclick='myOpen("OrderDetails")'>OrderDetails</button>
<br><br>

<div id="Place01"></div>

<script src="appml.js"></script>
<script>
function myOpen(pname)
{
var app_obj
app_obj=new AppML("appml.php","Prototypes/" + pname);
app_obj.run("Place01");
}
</script>

</body>
</html>
```

显示结果 »

AppML 案例简介　　AppML 案例模型

点我分享笔记

反馈/建议

<AppML> 案例研究 - 应用程序模型

此案例研究演示了如何构建一个完整的 <AppML> 互联网应用程序，具有针对数据库中的若干表进行信息列举、编辑和搜索的功能。

应用程序模型

在本章中，我们将为数据库中的 Customers 表建立一个完整的应用程序模型。

<AppML> 过滤器

如需允许过滤 <AppML> 数据，只需简单地向模型添加一个 <filters> 元素：

实例：

```
<filters>
<query>
<field label="Customer">CustomerName</field>
<field>City</field>
<field>Country</field>
</query>
<order>
<field label="Customer">CustomerName</field>
<field>City</field>
<field>Country</field>
</order>
</filters>
```

如需全面了解，请参阅 <AppML> 参考手册。

<AppML> 更新

如需允许更新 <AppML> 数据，只需简单地向模型添加一个 <update> 元素：

实例：

```
<update>
<item><name>LastName</name></item>
<item><name>FirstName</name></item>
<item><name>BirthDate</name></item>
<item><name>Photo</name></item>
<item><name>Notes</name></item>
</update>
```

且向 <database> 元素添加一个 <maintable> 和 <keyfield> 元素：

实例：

```
<maintable>Customers</maintable>
<keyfield>CustomerID</keyfield>
```

如需全面了解，请参阅 <AppML> 参考手册。

<AppML> 安全

您可以通过向 <AppML> 标签添加一个 security 属性来很容易地为 <AppML> 模型添加安全。

实例：

```
<appml security="admin">
```

在上面的实例中，只有用户登录成为用户组 "admin" 的会员才能访问模型。

如需为 <update> 元素设置安全，只需简单地向 <update> 元素添加一个 security 属性：

实例：

```
<update security="admin">
<item><name>LastName</name></item>
<item><name>FirstName</name></item>
<item><name>BirthDate</name></item>
<item><name>Photo</name></item>
<item><name>Notes</name></item>
</update>
```

完整的 Customers 模型

在本章中，我们将为数据库中的每个表设立一个应用程序模型。

创建一个名为 Models 的新文件夹。在 Models 文件夹中，为每个应用程序创建一个模型。

模型：Customers.xml

```
<appml security="">

<datasource>
<database>
<connection>Demo</connection>
<maintable>Customers</maintable>
<keyfield>CustomerID</keyfield>
<sql>SELECT * FROM Customers</sql>
<orderby>CustomerName,City,Country</orderby>
</database>
</datasource>

<filters>
<query>
<field label="Customer">CustomerName</field>
<field>City</field>
<field>Country</field>
</query>
<order>
<field label="Customer">CustomerName</field>
<field>City</field>
<field>Country</field>
</order>
</filters>

<update security="admin">
<item><name>CustomerName</name></item>
<item><name>ContactName</name></item>
<item><name>Address</name></item>
<item><name>PostalCode</name></item>
<item><name>City</name></item>
<item><name>Country</name></item>
</update>

</appml>
```

模型视图

创建一个模型视图，把它保存为 Demo_Model.html，并尝试一下：

视图：Demo_Model.htm

```
<h1>Customers</h1>
<div id="List01"></div>

<script src="appml.js"></script>
<script>
customers=new AppML("appml.htmlx","Models/Customers");
customers.run("List01");
</script>
```

尝试一下 »

现在把所有的合并在一起

然后，通过少量 JavaScript 编码，为所有模型创建一个测试页面：

Demo_Model_Views.htm

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="appml.css" />
</head>

<body>
<h1>Demo Applications</h1>

<button onclick='myOpen("Customers")'>Customers</button>
<button onclick='myOpen("Products")'>Products</button>
<button onclick='myOpen("Suppliers")'>Suppliers</button>
<button onclick='myOpen("Shippers")'>Shippers</button>
<button onclick='myOpen("Categories")'>Categories</button>
<button onclick='myOpen("Employees")'>Employees</button>
<button onclick='myOpen("Orders")'>Orders</button>
<button onclick='myOpen("OrderDetails")'>OrderDetails</button>
<br><br>

<div id="Place01"></div>

<script src="appml.js"></script>
<script>
function myOpen(pname)
{
var app_obj
app_obj=new AppML("appml.php","Models/" + pname);
app_obj.run("Place01");
}
</script>

</body>
</html>
```

显示结果 »

AppML 案例原型　　AppML 案例模板

点我分享笔记

反馈/建议

<AppML> 案例研究 - HTML 模板

此案例研究演示了如何构建一个完整的 <AppML> 互联网应用程序，具有针对数据库中的若干表进行信息列举、编辑和搜索的功能。

添加 HTML 模板

在本章中，我们将演示如何向 HTML 页面添加 HTML 模板。

列出客户

HTML - View

```
<h1>Customers</h1>

<div id="List01"></div><br>

<table id="Template01" class="appmltable" style="display:none">
<tr>
<th>Customer</th>
<th>City</th>
<th>Country</th>
</tr>
<tr id="appml_row">
<td>#CustomerName#</td>
<td>#City#</td>
<td>#Country#</td>
</tr>
</table>

<script src="appml.js"></script>
<script>
var customers
customers=new AppML("appml.php","Models/Customers");
customers.run("List01","Template01");
</script>
```

尝试一下 »

列出客户和客户表单

通过巧妙地使用模板，可以很容易添加 <AppML> 列表对象和 <AppML> 表单之间的链接：

HTML - View

```
<h1>Customers</h1>
<div id="Form01"></div><br>
<div id="List01"></div><br>

<table id="Template01" class="appmltable" style="width:100%;display:none">
<tr>
<th></th>
<th>Customer</th>
<th>City</th>
<th>Country</th>
</tr>
<tr id="appml_row">
<td style="cursor:pointer" onclick="openForm('#CustomerID#')">
<img src="images/appmlFolder.png"></td>
<td>#CustomerName#</td>
<td>#City#</td>
<td>#Country#</td>
</tr>
</table>

<script src="appml.js"></script>
```

```
<script>
var customers,customerForm;
customers=new AppML("appml.php","Models/Customers");
customers.run("List01","Template01");

function openForm(id)
{
customerForm=new AppML("appml.php","Models/Customers");
customerForm.displayType="form";
customerForm.run("Form01","",id);
}
</script>
```

尝试一下 »

列出客户和客户订单

通过巧妙地使用模板，可以很容易添加 <AppML> 列表对象和所链接的列表之间的链接：

HTML - View

```
<h1>Customers</h1>
<div id="List01"></div><br>
<div id="Orders01"></div><br>

<table id="Template01" class="appmltable" style="width:100%;display:none">
<tr>
<th>Customer</th>
<th>City</th>
<th>Country</th>
<th></th>
</tr>
<tr id="appml_row">
<td>#CustomerName#</td>
<td>#City#</td>
<td>#Country#</td>
<td><a href='' onclick='openOrders("#CustomerID#");return false;'>Orders</a></td>
</tr>
</table>

<table id="Template02" class="appmltable" style="width:100%;display:none">
<tr>
<th>Customer</th>
<th>Date</th>
<th>Salesperson</th>
<th>Shipper</th>
</tr>
<tr id="appml_row">
<td>#CustomerName#</td>
<td>#OrderDate#</td>
<td>#Salesperson#</td>
<td>#ShipperName#</td>
</tr>
</table>

<script src="appml.js"></script>
<script>
var customers,orders;
customers=new AppML("appml.php","Models/Customers");
customers.run("List01","Template01");
function openOrders(id)
{
orders=new AppML("appml.php","Models/Orders");
orders.setQuery("orders.customerid",id);
orders.commands=false;
orders.run("Orders01","Template02");
}
</script>
```

尝试一下 »

现在把所有的合并在一起

最后，通过少量代码复制，我们就可以完成项目。

客户列表、表单和订单

```
<h1>Customers</h1>

<div id="List01">
<table id="appml_list" class="appmllist">
<tr>
<th>Customer</th>
<th>City</th>
<th>Country</th>
<th></th>
</tr>
<tr id="appml_row">
<td>#CustomerName#</td>
<td>#City#</td>
<td>#Country#</td>
<td><a href='' onclick='openOrders("#CustomerID#");return false;'>Orders</a></td>
</tr>
</table>
</div>

<div id="List02"></div>

<script src="appml.js"></script>
<script>
var Customers,Orders
Customers=new AppML("appml.php","Models/Customers");
Customers.run("List01");

function openOrders(id)
{
var Orders=new AppML("appml.php","Models/Orders");
Orders.setQuery("orders.customerid",id);
Orders.commands=false;
Orders.run("List02");
}
</script>
```

尝试一下 »

在接下来的章节中，您可以看到更多带有完整源代码的应用程序。

AppML 案例模型　　AppML 案例 Employees

AppML 案例模板 AppML 案例 Customers

<AppML> Employees - 完整的应用程序

源代码

HTML 源代码

```
<html>
<body>

<h1>Employees</h1>

<div id="Form01"></div>
<div id="List01"></div>
<br>

<table id="Template01" class="appmltable" style="width:100%;display:none">
<tr>
<th></th>
<th>Last Name</th>
<th>First Name</th>
<th>Date of Birth</th>
</tr>
<tr id="appml_row">
<td style="cursor:pointer" onclick="employeesForm.run('Form01','Template02','#EmployeeID#')">
<img src="Images/appmlPlus.png"></td>
<td>#LastName#</td>
<td>#FirstName#</td>
<td>#BirthDate#</td>
</tr>
</table>

<div id="Template02" class="appmlform" style="width:100%;display:none">
<label>Last Name:</label><input id="LastName">
<label>First Name:</label><input id="FirstName">
<label>Date of Birth:</label><input id="BirthDate">
<label>Photo:</label><input id="Photo">
<label>Notes:</label><input id="Notes">
</div>

<script src="appml.js"></script>
<script>
var employees,employeesForm
employees=new AppML("appml.php","Models/Employees");
employees.run("List01","Template01");
employeesForm=new AppML("appml.php","Models/Employees");
employeesForm.displayType="form";
</script>

</body>
</html>
```

AppML 案例模板 AppML 案例 Customers

<AppML> Customers - 完整的应用程序

源代码

HTML 源代码

```
<html>
<body>

<h1>Customers</h1>

<div id="Form01"></div>
<div id="List01"></div><br>
<div id="Orders01"></div>

<table id="Template01" class="appmltable" style="width:100%;display:none">
<tr>
<th></th>
<th>Customer</th>
<th>City</th>
<th>Country</th>
<th>Orders</th>
</tr>
<tr id="appml_row">
<td style="cursor:pointer" onclick='customerForm.run("Form01","","#CustomerID#")'>
<img src="Images/appmlPlus.png"></td>
<td>#CustomerName#</td>
<td>#City#</td>
<td>#Country#</td>
<td><a href='' onclick='openOrders(#CustomerID#);return false;'>Orders</a></td>
</tr>
</table>

<table id="Template02" class="appmltable" style="width:100%;display:none">
<tr>
<th>Customer</th>
<th>Date</th>
<th>Salesperson</th>
<th>Shipper</th>
</tr>
<tr id="appml_row">
<td>#CustomerName#</td>
<td>#OrderDate#</td>
<td>#Salesperson#</td>
<td>#ShipperName#</td>
</tr>
</table>

<script src="appml.js"></script>

<script>
var customers,customerForm,orders;
customers=new AppML("appml.php","Models/Customers");
customers.run("List01","Template01");
customerForm=new AppML("appml.php","Models/Customers");
customerForm.displayType="form";
```

```
function openOrders(id)
{
orders=new AppML("appml.php","Models/Orders");
orders.commands=false;
orders.setQuery("orders.customerid",id);
orders.run("Orders01","Template02");
}
</script>

</body>
</html>
```

AppML 案例 Employees

AppML 案例 Products

点我分享笔记

反馈/建议

AppML 案例 Customers

AppML 未来的应用程序

<AppML> Products - 完整的应用程序

源代码

HTML 源代码

```
<html>
<body>

<h1>Products</h1>

<div id="Form01"></div>
<div id="List01"></div><br>

<table id="Template01" class="appmltable" style="width:100%;display:none">
<tr>
<th></th>
<th>Product Name</th>
<th>Category</th>
<th>Supplier</th>
</tr>
<tr id="appml_row">
<td style="cursor:pointer" onclick="productForm.run('Form01','Template02','#ProductID#')">
<img src="Images/appmlPlus.png"></td>
<td>#ProductName#</td>
<td>#CategoryName#</td>
<td>#SupplierName#</td>
</tr>
</table>

<div id="Template02" class="appmlform" style="width:100%;display:none">
<label>ProductName:</label><input id="ProductName">
```

```
<label>Supplier:</label>
<select id="SupplierID" data-appmlapplication="Models/Dropdown_Suppliers">
</select>
<label>Category:</label>
<select id="CategoryID" data-appmlapplication="Models/Dropdown_Categories">
</select>
<label>Unit:</label><input id="Unit">
<label>Price:</label><input id="Price">
</div>

<script src="appml.js"></script>
<script>
var products,productForm
products=new AppML("appml.php","Models/Products");
products.run("List01","Template01");
productForm=new AppML("appml.php","Models/Products");
productForm.displayType="form";
</script>

</body>
</html>
```

AppML 案例 Customers

AppML 未来的应用程序

点我分享笔记

反馈/建议

未来的应用程序

下文介绍**菜鸟教程版本的**有关 未来 Web 应用程序。

可执行文件将消亡，JavaScript 将存活

编译的可执行文件（如 C 或 Java 语言的编译）不能在不同的硬件上运行。

可执行文件（EXE 文件、ActiveX 和 COM 对象、DLL 文件）是防止运行在互联网上的应用程序发展的组件。

未来的应用程序将无法使用，或依靠，安装在客户端计算机上的组件。

我们的建议：

仅使用 HTML、CSS 和 JavaScript 来编写您未来的应用程序。

确保您未来的应用程序可以运行在任何的 Web 浏览器中。

Web 应用程序将成为互联网服务

过去的应用程序都是大型的、为特定目的创建的应用程序。它们中的大多数很快将会消亡，因为它们不能随着需求的变更提供服务。

应用程序应该是灵活的、通用的、能在不被破坏的情况下很好地适应需求的变更。

应用程序应该能够扩展，从支持每天数以百万计的请求。

应用程序应该能够在不被破坏的情况下，在多个服务器之间进行传播和移动。

应用程序应该能够与其它的应用程序共同使用。

应用程序不应该包含大量的代码。应用程序应该被分解成更小的服务，以便易于创建和易于维护。

应用程序是可返回数据到提交的互联网请求的一系列互联网服务。

应用程序应该在不保持到服务器的永久连接的情况下通过标准的互联网协议请求服务。

我们的建议：

使用基于 SOA（Service Oriented Architecture 面向服务架构）的互联网来编写您未来的应用程序。

让您的应用程序服务更通用，更灵活，且能服务于不同类型的请求。

未来的应用程序将很容易创建和编辑

客户端和服务器端将在一个易于理解的方式进行数据交换。

如果能够避免，应用程序将不被编码。

应用程序将通过编辑模型而不是编辑代码来创建和修改。

应用程序描述将被人类可读。

应用程序描述将具有自我描述性。

应用程序将由用户编写，而不是由程序员编写。

我们的建议：

使用人类可读的文本文件来描述服务，并通过执行这些描述来提供服务。

使用文本文件（如 XML 文件）来描述应用程序。

使用文本文件（如 XML 文件）来进行数据交换。

使用 HTML、CSS 和 JavaScript 来执行应用程序。

三个小的 Web 开发人员...

很久以前，有三个小的 Web 开发人员，开发一个新的 Web 站点。

1. 第一个 Web 开发人员使用 <AppML>。

2. 第二个 Web 开发人员使用他最喜欢的服务器端编程语言。

3. 第三是使用专业的企业 Web 开发框架。

第一个 Web 开发人员在两天内完成演示。经过与用户的合作，一周内完成了初步原型。经过两周的测试，一个智能的、快速的和易于使用的网站准备发布。

6 个月后，第二个 Web 开发人员准备好了他的网站。但是 WWW 已经改变了需求，所以对他的网站并不满意。这个 Web 开发人员无法对他的项目做出重大的改变，因为它包含了太多的代码。于是，他开始第 2 版的开发。

第三个 Web 开发人员从来没有成功地完成他的工作。那个专业的 Web 开发框架，使用起来很困难，很难理解，几乎是不可能的测试。

看一看第一个开发人员是如何做到的。