第 3 章　Web 应用开发

3.1　Web 应用概论

3.1.1　什么是 Web 应用

浏览器诞生之初仅仅是一个用来显示服务器产生页面的工具，随着几次浏览器大战之后，HTML5 标准和浏览器的性能已经深入人心，浏览器已经变成了一个应用平台。而 Web 的三种语言 HTML、CSS 和 JavaScript 则用来表示应用的内容、修饰应用界面以及处理交互逻辑。智能手机等移动终端和移动互联网的发展，又为应用开发提供了一个更加广阔的空间。技术和生态系统的成熟让基于 Web 的应用成为一种必然的趋势。

Web 应用就是这样一种基于 HTML、CSS 和 JavaScript 语言开发的运行在 Web 运行时之上的应用形式。一个网站其实就可以看做是一个 Web 应用，这个 Web 应用首先下载到客户端，然后通过浏览器运行，而有时候通过离线应用技术，可以让 Web 应用在没有网络的时候离线运行。Web 应用也可以被打包成一个独立的应用，像本地应用一样直接安装运行在系统平台上。后一种情况通常基于可扩展的 Web 运行时，提供更多的本地资源访问接口，让 Web 应用更像本地应用。

Web 运行时与浏览器并不等同，Web 运行时是浏览器最核心的部分，负责上述三种语言的渲染和执行，而浏览器还包括了浏览网页特有的一些功能，例如导航（前进后退和地址栏）、加载进度显示、控制弹出窗口和弹出消息、广告拦截、手势输入等，而这些是 Web 应用不需要的功能。简而言之，浏览器＝Web 运行时＋浏览器界面。

Web 运行时又由两个最核心的部分构成：排版引擎（Layout Engine）和 JavaScript 引擎。排版引擎用于根据 HTML 和 CSS 等显示出正确的页面内容和布局，JS 引擎负责解析和执行 JavaScript 脚本。关于排版引擎和 JS 引擎更加详细的内容请参考：http://en.wikipedia.org/wiki/Layout_engine。

3.1.2　本地应用 vs Web 应用

与 Web 应用相对的就是本地应用，这两种应用从架构上有一些不同。图 3-1a 就是一个 Windows 操作系统上的本地应用架构，本地应用通常用 C/C＋＋等语言编程，由 Windows 支持的编译器和库函数辅助生成本地应用，应用运行在 Win32 的系统接口之

上。其他操作系统，比如 Linux、Android、iOS、Windows Phone 等在本地应用都采用相似的架构。Android、iOS 和 Windows Phone 都提供了一些基于虚拟机技术的编程语言，例如 Java、Objective C、C#，一定程度上解决了应用性能和开发效率之间的平衡，但是不同虚拟机在不同操作系统上没有互相代替的实现，实际上反而不如 C/C++有更好的跨平台性。

本地应用的优点在于应用与操作系统之间没有额外的转换或解释层，也就没有转换解释的时间，依靠编译器对代码进行静态的优化，而开发者还可以采用汇编级别的优化，达到最佳的性能。但本地应用的缺点也很明显，当需要开发一个跨平台的应用时，开发者不仅需要掌握各种不同操作系统和 UI 控件接口，还要针对设备的 CPU、屏幕分辨率、传感器等进行复杂的移植工作，有时甚至不得不为新的设备或操作系统重新开发一个应用，开发和维护成本高，且很难做到一致的用户体验。

Web 应用基本上采用 HTML、CSS 和 JavaScript 语言开发，调用 Web 运行时提供的标准 HTML5 接口，Web 运行时可以运行在各种操作系统和 CPU 架构之上（如 3-1b 所示）。对于开发者来说只要掌握一套技术就可以开发跨平台的应用，在各种操作系统和硬件设备上运行，平台兼容性的问题基本上由 Web 运行时来解决，因此 Web 应用有开发效率高和跨平台的优势。Web 运行时相当于一个虚拟机，相对于本地应用需要额外花费一些时间进行动态的解释，但随着 CPU 性能的提升和运行时的优化，与本地应用之间的性能差距正在急剧缩小。另外，Web 应用还可以采用一种与本地应用混合的形式开发，以满足高性能和特殊平台调用的需求，下一小节 Web 应用的形态中会详细介绍。

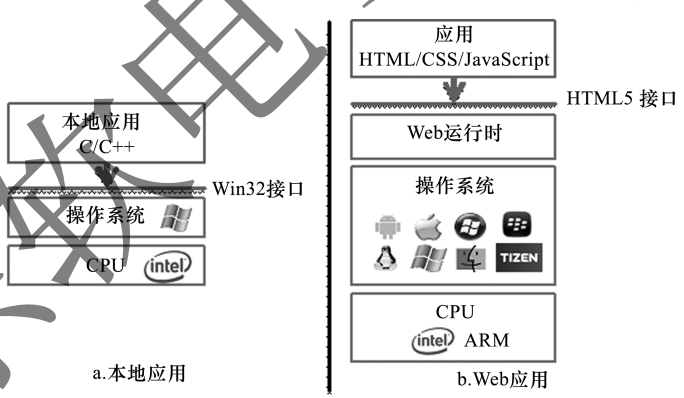

图 3-1　本地应用和 Web 应用架构比较

Web 应用与本地应用的差别除了架构带来的性能等差别之外，还有对本地设备的访问，浏览器一开始并不提供本地设备的访问，这是由过去网站的需求和安全方面的考虑来决定的。但用 Web 开发应用就不得不访问到本地的设备资源，W3C 的设备 API 工作组（Device APIs Working Group）负责讨论设计本地设备访问的标准 API，例如媒体捕获和流（Media Capture and Stream）、Web Intent 等，但浏览器厂商尚未广泛实现这些接口。同时也有浏览器厂商独立提出和实现了自己的设备 API 接口，例如 Mozilla 的 Web API，微软在 Windows 8 平台上提供的 Windows Runtime API，如表 3-1 所示。

表 3-1 本地应用和 Web 应用比较

	本地应用	Web 应用
编程语言	C/C++/Objective C/C#/…	HTML+CSS+JavaScript
跨平台	困难	容易
性能优化	语言优化+编译器优化	JS 引擎优化
本地设备	直接访问	W3C 或平台的设备 API

3.1.3 Web 应用的形态

由于标准 API 对设备访问能力的不足和 Web 运行时实现之间的差别，出现了几种不同形态的 Web 应用：基于浏览器的应用、混合应用和平台专属的应用。

1. 基于浏览器的应用

浏览器是传统网站的入口，Web 应用也可以依托于浏览器这个平台上运行。如图 3-2 所示展示了这种应用的基本架构，从表现形式上来看，Web 应用就像一个传统网站一样显示在浏览器内部，浏览器窗口上的一些 UI 元素，比如导航栏、地址栏等，都保留了下来。基于浏览器的 Web 应用与传统网站的不同是浏览器通常会给予它们更多的权限，比如更强的跨域(Cross Domain)访问能力、更多的本地存储空间、对剪切板的访问等(详见 Chrome 扩展的 Manifest 文件说明 http://developer.chrome.com/extensions/manifest.html)。

基于浏览器的 Web 应用只能使用浏览器提供的标准 HTML5 接口，因此它们可以运行于各种浏览器和各种操作系统和硬件平台上，实现跨平台的效果，而且可以通过 HTML5 的 Offline Application Cache 实现离线状态下运行应用。但缺点就是对本地设备的访问受限于 W3C 的标准和浏览器的实现，无法 100%发挥设备的能力，性能方面又完全依赖于 Web 运行时的实现，因此无法达到最大程度的优化。

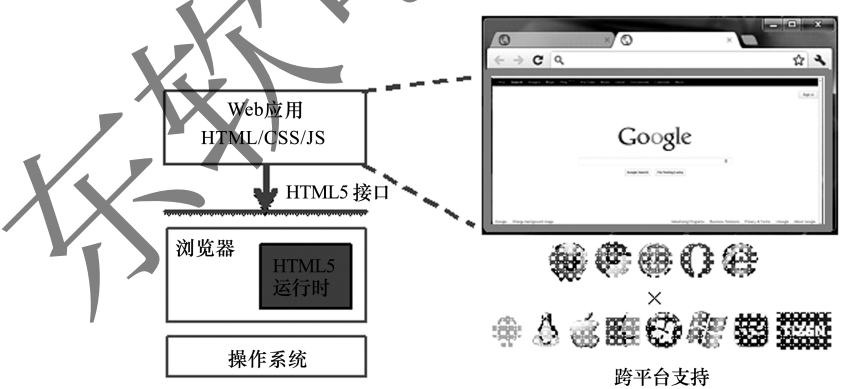

图 3-2 基于浏览器的应用

2. 混合应用(Hybrid App)

混合应用(Hybrid App)是基于浏览器的应用和本地应用之间的一种过渡形式，混合应用在开发同时采用 Web 语言和本地语言：Web 部分通过 HTML5 标准接口实现界面和基本的交互逻辑，称为 Webview，但系统提供的 Web 运行时并不包含很多设备访问的接口，因此需要通过本地语言来实现这部分功能；本地语言一方面调用本地操作系统的接

口来实现非标准的本地设备访问，一方面通过 HTML5 运行时的扩展能力为 Web 部分提供相应的接口，让其可以间接地访问到更多的本地设备，来弥补 W3C 标准在设备访问方面的不足。除了提供必要的设备访问能力之外，本地代码部分还常用来实现一些高性能的或者需要加密的算法。Adobe 的 PhoneGap(http://phonegap.com/)就是一个混合应用的框架，它通过为不同平台上提供一套相同的设备 API 的访问接口，给出一个间接完整的跨平台应用的解决方案。

混合应用最大的优势是可以把 Web 应用打包成一个普通的应用包，直接安装到目标平台上，而不依赖于浏览器运行。分别利用了两种语言的优势达到一种跨平台性和性能间的平衡，用户体验上混合应用可以达到与本地应用毫无差别。但由于采用了本地代码开发，需要手工实现不同平台的移植，这也几乎是它唯一的缺点，如图 3-3 所示。

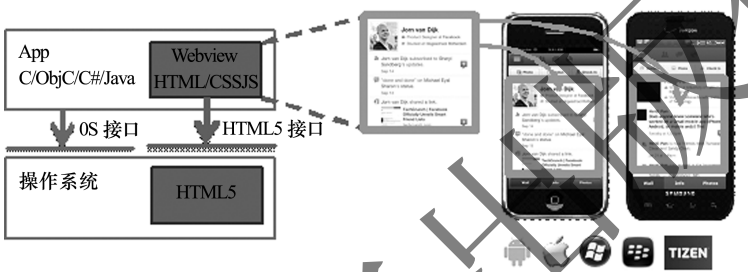

图 3-3　混合应用

3. 平台专属应用

平台专属的 Web 应用跟基于浏览器的应用类似也是完全 Web 语言来实现，不同的是平台原生提供了访问本地设备的 Web 接口，但这些接口暂时没有成为 W3C 的标准。例如微软的 Windows Runtime(简称 WinRT)、Palm 的 WebOS 和 Mozilla 的 Firefox OS，就是这样的平台。

平台专属的应用可以像混合应用一样作为独立的应用安装，而不依赖于浏览器，达到本地应用相同的用户体验。但由于采用了平台自己的设备访问接口，造成应用只能运行在特定的平台上，影响应用的跨平台性，如图 3-4 所示。

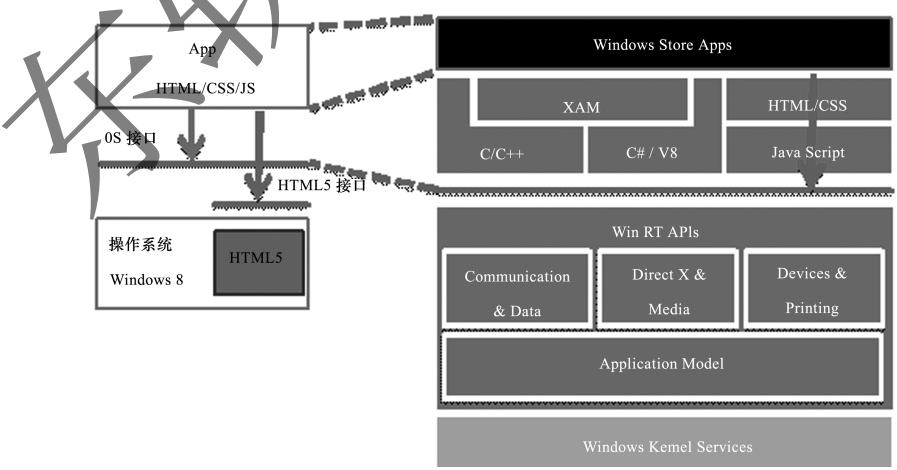

图 3-4　平台专属应用

3.1.4 Web 应用发展趋势

Web 应用有两个发展方向，一是服务端生成计算，二是客户端服务集成，下面分别介绍。

1. 服务端生成计算

Web 应用的主要开发语言是 HTML＋CSS＋JavaScript，都是纯文本格式，跟传统网站的开发语言相同，也就意味着 Web 应用也可以从服务端动态生成，在用户需要的时候从客户端获取和更新，相对来说本地应用通常以二进制的格式发布，应用的更新通过应用市场或官方网站进行更新（如图 3-5 所示）。

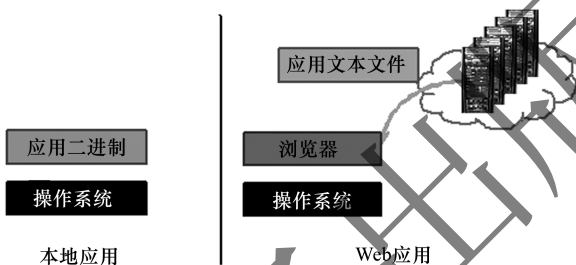

图 3-5 按需更新应用

同时，如图 3-6 所示，Web 应用的代码文件可以采取两种发布方式，一是通过静态文件的方式发布，这种方式像本地应用一样将全部代码文件打包发布；二是根据用户、当前环境、设备，以及服务器上的数据、人际关系等，动态生成 Web 应用的代码，类似传统网站的更新方式，而不需要通过应用市场的审查和发布。

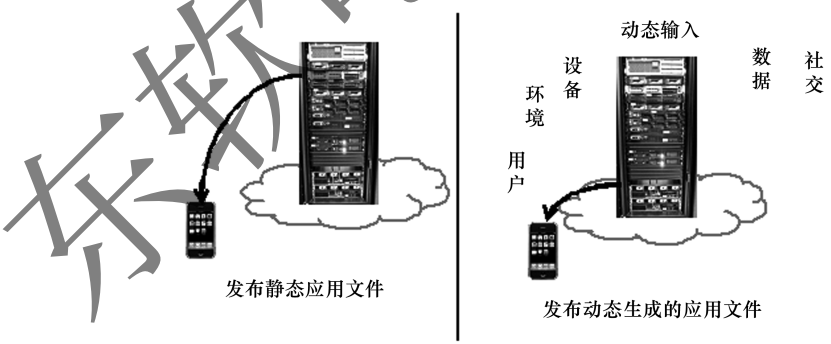

图 3-6 动态生成应用

2. 客户端服务集成

Web 应用发展的另一个方向就是客户端的服务集成（如图 3-7 所示），很多第三方服务都提供 RESTFUL 的接口进行调用，例如社交网站的 OAuth 登陆服务，云存储服务，复杂的计算服务，对于 Web 应用来说，基于 HTTP 协议的 RESTFUL 接口是一种非常自然的调用方式，直接使用 AJAX 可以调用。

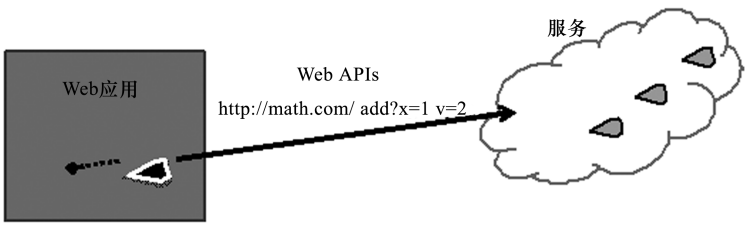

图 3-7　客户端服务集成

3.2　前端框架

“工欲善其事，必先利其器”。利用目前已有的 HTML5 前端开发框架来快速开发应用无疑是一个非常有吸引力的选择。选择一个合适的 HTML5 开发框架可以有效保障手工编码带来的代码稳定性问题，同时可以加快开发进度。

本节将主要介绍目前流行的三大 HTML5 前端开发框架：jQuery Mobile，Bootstrap 和 backbone。

3.2.1　jQuery Mobile

1. 简介

jQuery Mobile 是由(MT)Media Temple 联合多家移动设备厂商以及软件企业共同发起的针对触屏智能手机与平板电脑的 Web 应用的前端开发框架。

jQuery Mobile 构建于大名鼎鼎的 jQuery 以及 jQuery UI 类库之上，为前端开发人员提供了一个兼容所有主流移动设备平台的统一的 UI 接口系统。拥有出色的弹性，轻量化以及渐进增强特性与可访问性。

jQuery Mobile 框架遵循“write less，do more”思想来设计，通过该框架，用户可以开发跨平台、跨设备的统一的 HTML5 应用，无需针对每个设备和操作系统分别开发应用代码。目前 jQuery Mobile 支持的的系统和设备有：iOS，Android，BlackBerry，Tizen，Bada，Windows Phone，WebOS，Symbian。

2. 下载和使用

jQuery Mobile 强调语义标注，非常易于使用。只要会使用基本的 HTML，就可以基于 jQuery Mobile 快速构建 Web 应用。目前 jQuery Mobile 最新的稳定版本为 V1.2.0，可以通过 jQuery Mobile 官方网站 http://jquerymobile.com/ 下载源码。

jQuery Mobile 使用非常简单，打开任何你所喜好的文本编辑器，新建一个文件并命名为：test.html，在该文件开始处添加 jQuery Mobile 框架代码：

```
<! DOCTYPE html>
<html>
<head>
    <title>My Page</title>
    <meta name = "viewport" content = "width = device-width, initial-scale = 1">
    <link rel = "stylesheet" href = "http://code.jQuery.com/mobile/1.2.0/jQuery.mobile-
1.2.0.min.css" />
    <script src = "http://code.jQuery.com/jQuery-1.8.2.min.js"></script>
    <script src = "http://code.jQuery.com/mobile/1.2.0/jQuery.mobile-1.2.0.min.js">
```

```
</script>
    </head>
    <body>
    Add your content here.

    </body>
    </html>
```

保存文件，并在浏览器中打开，即可浏览。

下面我们来实现一个 Hello World 例子，<body> 标签中的内容用下面代码替换。

```
<body>

<div data-role = "page">

    <div data-role = "header">
    <h1>hello</h1>
    </div><! - - /header - - >

    <div data-role = "content">
    <p>Hello world</p>
    </div><! - - /content - - >

</div><! - - /page - - >

</body>
```

在这段代码中，我们首先使用一个 data-role 来描绘该页面，同时使用 hello 作为我们的 header data-role。在 content 标签里面，我们填入了 Hello world 。这些特定的 data-role 定义的 HTML5 属性，在 JQuery Mobile 中用于快速构建增强风格的用户界面。

保存代码并在浏览器中打开，效果如下图 3-8 所示：

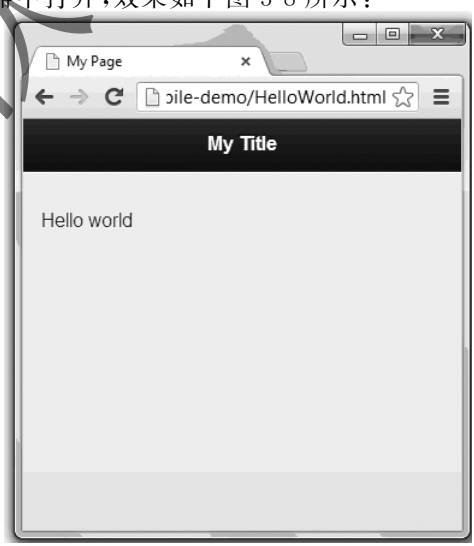

图 3-8 Hello world 运行效果

读者可以调整浏览器边框的大小来查看 jQuery Mobile 构建的 header，Content 等自动添加了哪些样式。My Title 可以根据屏幕大小自动居中，不管屏幕分辨率是多少，而内容部分 Hello World 则靠左对齐。

下面我们再实现以下稍微复杂一点的界面，将正文部分添加进我们移动开发中常见的列表视图，并在结果添加两个并排的按钮。将<body> 标签中的内容替换，完整代码如下：

```
<! DOCTYPE html>
<html>
<head>
    <title>My Page</title>
    <meta name = "viewport" content = "width = device-width, initial-scale = 1">
    <link rel = "stylesheet" href = "http://code.jQuery.com/mobile/1.2.0/jQuery.mobile -
1.2.0.min.css" />
    <script src = "http://code.jQuery.com/jQuery - 1.8.2.min.js"></script>
    <script src = "http://code.jQuery.com/mobile/1.2.0/jQuery.mobile - 1.2.0.min.js">
</script>
</head>
<body>

<div data-role = "page">

<ul data-role = "listview" data-inset = "true" >
    <li data-role = "list-divider" role = "heading">Working Week</li>
    <li><a href = "#">Monday</a></li>
    <li><a href = "#">Tuesday</a></li>
    <li><a href = "#">Wednesday</a></li>
    <li><a href = "#">Thursday</a></li>
    <li><a href = "#">Friday</a></li>
</ul>

</div><! - - /page - - >

</body>
</html>
```

我们在浏览器中打开，效果如图 3-9 所示：

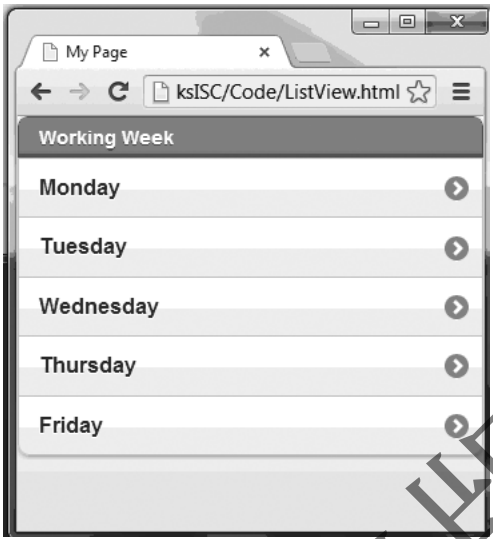

图 3-9 ListView 默认效果

下面我们尝试修改显示的主题。在 list-divider 中添加 data-divider-theme＝"g"，data-theme＝"a" ，更新效果如图 3-10 所示：

```
＜ul data-role = "listview" data-divider-theme = "a"
data-theme = "e" data-inset = "true" data-inset = "true" ＞
```

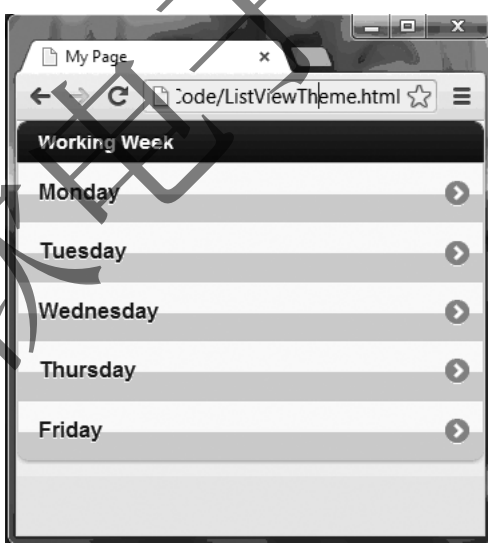

图 3-10 ListView 修改主题后效果

3.2.2 Bootstrap

1. 简介

Bootstrap 是 Twitter 推出的一个开源的前端框架。Bootstrap 由 Twitter 的设计师 Mark Otto 和 Jacob Thornton 合作开发，由动态语言 Less 写成。它是一套“易用、优雅、灵活、可扩展”的前端工具集，提供了优雅的 HTML/CSS 规范。

Bootstrap 一经推出后颇受欢迎，一直是 GitHub 上的热门开源项目，包括 MSNBC（微软全国广播公司）的 Breaking News 都使用了该项目。Bootstrap 兼容于所有主流浏览器，包括各种移动设备。

2. 下载和使用

Bootstrap 建立了一个响应式的 12 网格布局系统，它引入了 fixed 和 fluid-with 两种布局方式，可以快速构建 Web 应用。目前 Bootstrap 最新版本为 3.0.0，可以在 Bootstrap 的官方网站 http://twitter.github.com/bootstrap/上下载源码。

Bootstrap 使用非常简单，打开任何文本编辑器，新建一个文件并命名为：test.html，在该文件开始处添加 Bootstrap 框架代码：

```
<! DOCTYPE html>
<html lang = "en">
<head>
<script src = "jQuery.js"></script>
<script src = "bootstrap.js"></script>
<link href = "bootstrap.css" rel = "stylesheet">
</head>
```

从上面的代码可以看到：Bootstrap 基于 JQuery 构建，所以需先引入 JQuery。Bootstrap 框架代码包括了 bootstrap.js 和 bootstrap.css 两个文件。

然后，我们可以将如下代码替换 body 元素的内容：

```
<body>
    <div class = "container-fluid">
        <div class = "row-fluid">
            <div class = "span3">
                <div class = "well sidebar-nav">
                    <ul>
                        <li><a href = "#">link</a></li>
                        <li><a href = "#">link</a></li>
                        <li><a href = "#">link</a></li>
                        <li><a href = "#">link</a></li>
                        <li><a href = "#">link</a></li>
                        <li><a href = "#">link</a></li>
                        <li><a href = "#">link</a></li>
                    </ul>
                </div>
            </div>
            <div class = "span9">
                <div class = "hero-unit">
                    <h1>Hello world! </h1>
                </div>
            </div>
```

```
        </div>
    </div>
</body>
```

上面的代码中，我们采用流式布局构建，将页面分成两部分：左边是 7 个链接，占 3 个宽度；右边是一个 Hello world 的标题，占 9 个宽度。效果如图 3-11 所示：

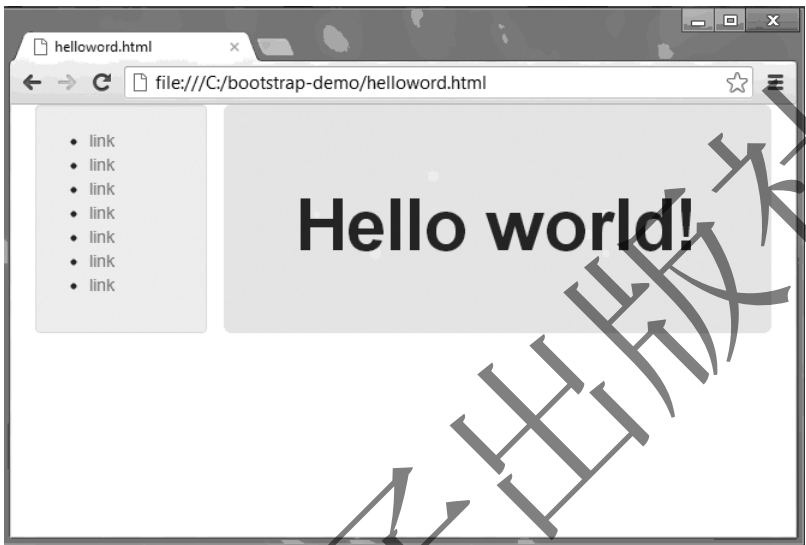

图 3-11 流式布局效果图

对上面的 Hello world 应用稍作改变，我们可以实现一个简单的 HTML5 介绍网站。将 test.html 页面的 body 元素中的内容替换成如下代码：

```
<body>
    <div class = "container-fluid">
        <div class = "row-fluid">
            <div class = "span3">
                <div class = "well sidebar-nav">
                    <ul class = "nav nav-list">
                        <li class = "nav-header">Sections</li>
                        <li><a href = "#">HTML5 Introduction</a></li>
                        <li><a href = "#">HTML5 Course</a></li>
                        <li><a href = "#">HTML5 Drag</a></li>
                        <li><a href = "#">HTML5 Geolocation</a></li>
                        <li><a href = "#">HTML5 Audio</a></li>
                        <li><a href = "#">HTML5 Vedio</a></li>
                        <li><a href = "#">HTML5 Canvas</a></li>
                    </ul>
                </div>
            </div>
            <div class = "span9">
```

```
                <div class="hero-unit">
                    <h1>Welcome to HTML5! </h1>
                </div>
                <div class="row-fluid">
<div class="span4">
<h2>HTML5 Introduction</h2>
<p>HTML5 Introduction…</p>
<p><a class="btn" href="#">View details &raquo;</a></p>
</div>
<div class="span4">
<h2>HTML5 Course</h2>
<p>HTML5 Course…</p>
<p><a class="btn" href="#">View details &raquo;</a></p>
</div>
<div class="span4">
<h2>HTML5 Drag</h2>
<p>HTML5 Drag…</p>
<p><a class="btn" href="#">View details &raquo;</a></p>
</div>
</div>
<div class="row-fluid">
<div class="span4">
<h2>HTML5 Audio</h2>
<p>HTML5 Audio…</p>
<p><a class="btn" href="#">View details &raquo;</a></p>
</div>
<div class="span4">
<h2>HTML5 Vedio</h2>
<p>HTML5 Vedio…</p>
<p><a class="btn" href="#">View details &raquo;</a></p>
</div>
<div class="span4">
<h2>HTML5 Canvas</h2>
<p>HTML5 Canvas…</p>
<p><a class="btn" href="#">View details &raquo;</a></p>
</div>
</div>
            </div>
        </div>
    </div>
</body>
```

效果如下图 3-12 所示：

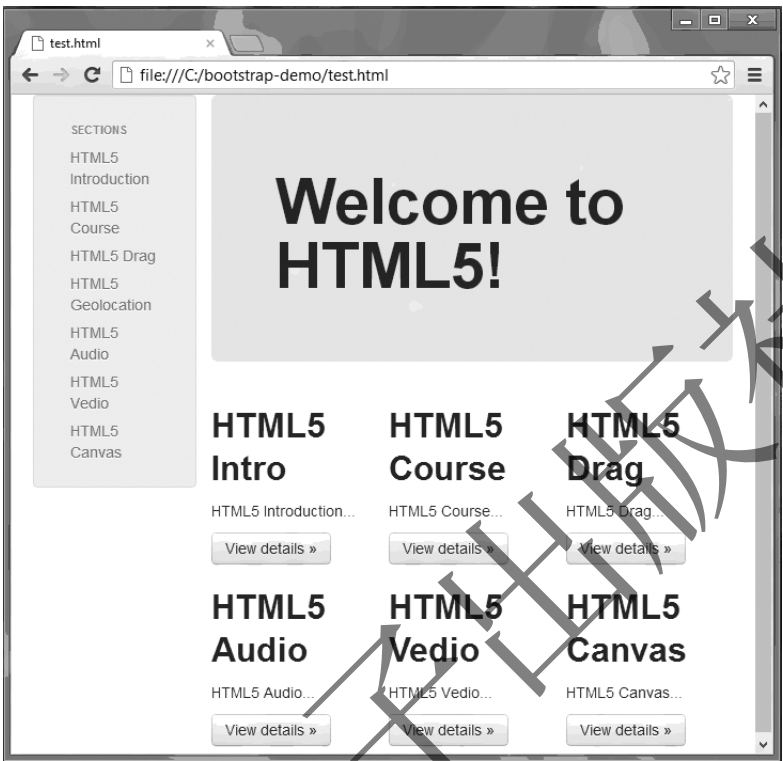

图 3-12 html5 页面

3.2.3 Backbone

1. 简介

Backbone 是一种重量级的 JavaScript MVC 应用框架。Backbone 的设计遵循了 MVC 模式的思想，它提供了三种类型：Model，View，Collection。其中，View 实现了 MVC 模式中 view 和 controller 的功能，而 Model 和 Collection 的结合使用，承担了 MVC 模式中 model 的职责。

Backbone 构建于 JQuery 和 underscore 这两个类库上。通过该框架，用户可以在开发的 HTML5 应用时，无需编写额外的代码，就可以使得 Model 改变时，View 所代表的 HTML 元素也会随之自动更新。

2. 下载和使用

Backbone 的最新稳定版可以在 github 上下载：https://github.com/document-cloud/backbone。Backbone 构建于 JQuery 和 underscore 这两个类库上，所以在使用 Backbone 之前，我们需要先加载 JQuery 和 underscore。

首先将 JQuery，underscore，Backbone 三个 js 文件记载到当前目录下。打开任何你所喜好的文本编辑器，新建一个文件并命名为 test.html，在该文件开始处添加 JQuery、underscore 和 Backbone 代码：

```
<! DOCTYPE html>
```

```
<html>
<head>
<script src = "jQuery - 1.7.1.js"></script>
<script src = "underscore.js"></script>
<script src = "backbone.js"><script>
<link rel = "stylesheet" href = "todos.css"/>
</head>
</html>
```

保存文件,并在浏览器中打开,即可浏览。

下面我们来实现一个备忘录的例子,<body>标签中的内容用以下内容替换:

```
<body>
<div id = "todoapp">
<h1>Todos</h1>
<input id = "new-todo" type = "text" place = "What needs to be done?">
<ul id = "todo-list"></ul>
</div>
<script type = "text/template" id = "item-template">
<div class = "view">
<input class = "toggle" type = "checkbox" />
<label><% - title %></label>
</div>
</script>
</body>
```

在这段代码中,我们首先定义了页面标题“Todos”,一个提示信息是“What needs to be done”的文本框“new-todo”和一个暂时没有内容的无序列表“todo-list”。然后,加载一段脚本 “item-template”,作为元素的模板,供之后的 JavaScript 代码使用。效果如图 3-13:

图 3-13　备忘录例子

此时,这张页面仅仅是一张静态 HTML 页面,不具备任何的交互能力。为了让它响

应用户的操作，接下来我们将为它添加相应的 JavaScript 代码。

打开本文编辑器，新建一个文件命名为“todos.js”，并在其中添加如下代码：

```
$(function(){
    var Todo = Backbone.Model.extend({
        defaults:function(){
            return {
                title:"empty todo",
            };
        },
    });

    var TodoList = Backbone.Collection.extend({
        model: Todo
    });

    var Todos = new TodoList;


    var TodoView = Backbone.View.extend({

        tagName:  "li",

        template: _.template( $('#item-template').html()),

        events: {
            "keypress .edit"  : "updateOnEnter",
        },

        initialize: function() {
    this.model.on('change', this.render, this);
            this.model.on('destroy', this.remove, this);
        },

        render: function() {
    this.$el.html(this.template(this.model.toJSON()));
            return this;
},

      updateOnEnter: function(e) {
    if (e.keyCode == =13) this.close();
        },
    });
```

```
var AppView = Backbone.View.extend({

    el: $('#todoapp'),

    events: {
        "keypress #new-todo":  "createOnEnter",
    },

    initialize: function() {
this.input = this.$("#new-todo");
        Todos.on('add', this.addOne, this);
    },

    addOne: function(todo) {
var view = new TodoView({model: todo});
        this.$("#todo-list").append(view.render().el);
    },

    createOnEnter: function(e) {
if (e.keyCode != 13) return;
        if (!this.input.val()) return;

        try{
            Todos.create({title: this.input.val()});
        }
        catch(e){
            //do nothing
        }
        this.input.val('');

    },
});
var App = new AppView;

});
```

在上面的代码中，我们新建了四个类，其中 Todo 继承了 Model 类，TodoList 继承了 Collection 类；而 TodoView 代表了 View 的职责，它代表了 HTML 列表中的<li>元素，而 AppView 代表整个 Web 应用。

AppView 会监听页面上文本框“new-todo”，一旦用户按下回车键【Enter】，AppView 就会调用 TodoList，将文本框中用户输入的内容作为“title”属性，在 TodoList 中添加一

个新的 Todo 对象成员；一旦添加成功，就会触发 TodoList 的 add 事件。向页面中的<ul>元素添加新的<li>成员，并重新渲染页面。

下面我们将“todos.js”引入 test.html，完整的 HTML 代码如下：

```
<! DOCTYPE html>
<html lang = "en">
<head>
    <title>This is a test</title>
<script src = "jQuery.js"></script>
<script src = "underscore.js"></script>
<script src = "backbone.js"></script>
<script src = "todos.js"></script>
<link rel = "stylesheet" href = "todos.css"/>
</head>
<body>
<div id = "todoapp">
<h1>Todos</h1>
<input id = "new-todo" type = "text" placeholder = "What needs to be done?">
<ul id = "todo-list"></ul>
</div>
<script type = "text/template" id = "item-template">
<div class = "view">
<input class = "toggle" type = "checkbox" />
<label><% - title %></label>
</div>
</script>
</body>
</html>
```

打开浏览器，在文本框中输入任意内容，并按下回车键。效果如图 3-14 所示：

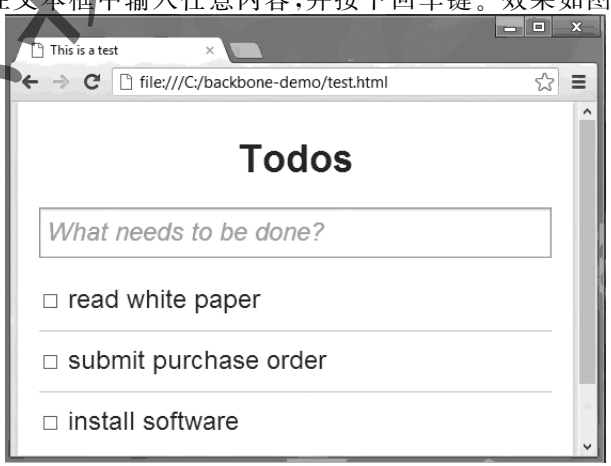

图 3-14 效果图

3.3　使用 RIB 进行 App 快速开发

3.3.1　RIB 简介

RIB(Rapid Interface Builder) 是一个基于浏览器的 HTML5 应用开发工具,具有图形化设计界面,由 Intel 支持。

使用 RIB 可以快速进行 Web 应用的 UI 设计,支持 JQuery Mobile 和 Tizen widget。

3.3.2　本地开发

RIB 可以把开发环境下载安装到本地,但是只能用 chrome,并且设置 chrome 属性,对于 Windows,右键点击开始菜单的 chrome 图标,在参数对话框修改参数,添加如下命令后缀:

```
"C:\Program Files\Google\Chrome\Application\chrome.exe"  - - allow-file-access-from-files - - enable-file-cookies
```

对于 Linux,用如下命令打开 chrome:

```
$ google-chrome ——enable-file-cookies ——allow-file-access-from-files
```

再打开下载到本地的 rib 目录的 index.html,便有了本地的开发环境。

3.3.3　举例

接着我们开发一个具有三个页面的 Web 应用。

第一页:用户登录。第二页:用户注册。第三页:登录成功。如图 3-15。

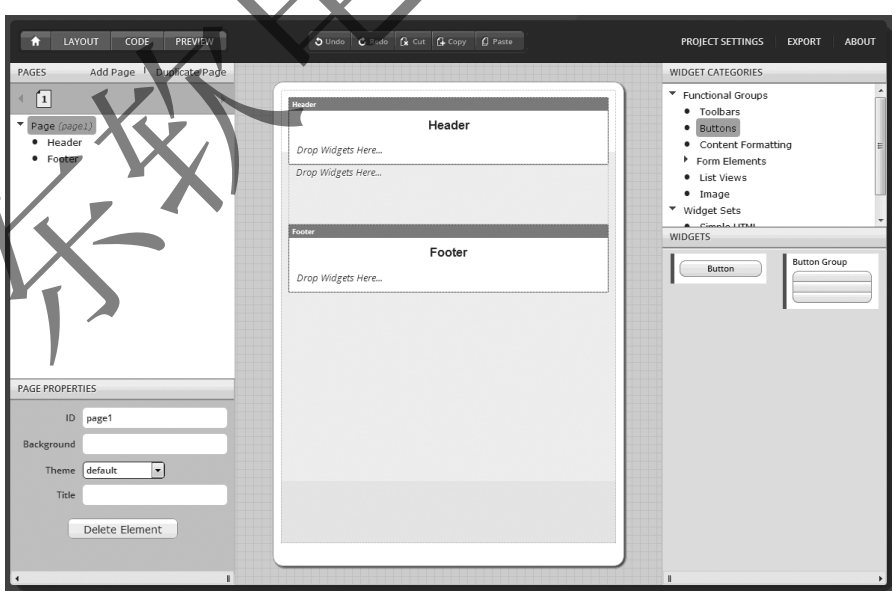

图 3-15　示例图

在编辑过程中，可以使用 Undo、Redo、Cut、Copy、Paste。

1. 制作第一页

(1)打开网页 https://01.org/rib/online，缺省就有第一页存在。

(2)选择 header，左侧的属性栏修改 Text 内容为 Demo - Login。

(3)选择 footer，左侧的属性栏修改 Text 内容为 Web 应用 by Intel。

(4)然后从右侧拖拽下列元素到中间 header 和 footer 之间的区域。

(5)插入 Form Elements/Text Input，设置 Label 为 name，其缺省 ID 是 text1。

(6)插入 Form Elements/Text Input，设置 Label 为 pass，其缺省 ID 是 text2。

(7)插入 Button，设置 Text 为 Login，修改 ID 是 login

(8)插入 Form Elements/Text，设置 Text 为 or。

(9)插入 Button，设置 Text 为 Register，修改 ID 是 register。

如图 3-16。

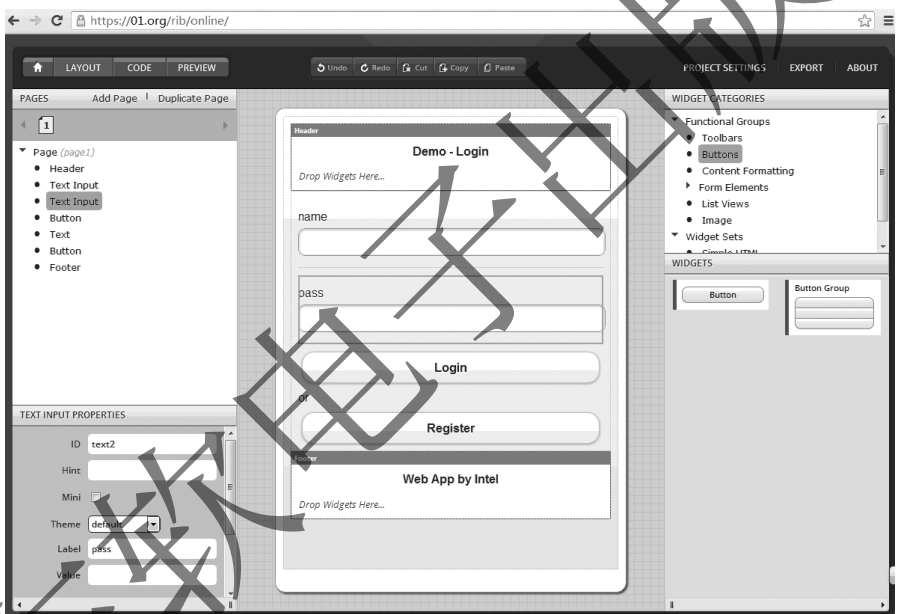

图 3-16　第一页的制作

当前的状态是 LAYOUT，左上角切换成 CODE，显示了自动生成的代码。如图3-17。

```
<! DOCTYPE html>
<html>
    <head>
        <title>title</title>
        <meta name = "viewport" content = "width = device-width, initial-scale = 1">
        <script src = "lib/jquery - 1.6.4.js"></script>
        <script src = "lib/jquery.mobile - 1.1.0.js"></script>
        <link href = "src/css/jquery.mobile.structure - 1.1.0.css" rel = "stylesheet">
        <link href = "src/css/jquery.mobile.theme - 1.1.0.css" rel = "stylesheet">
```

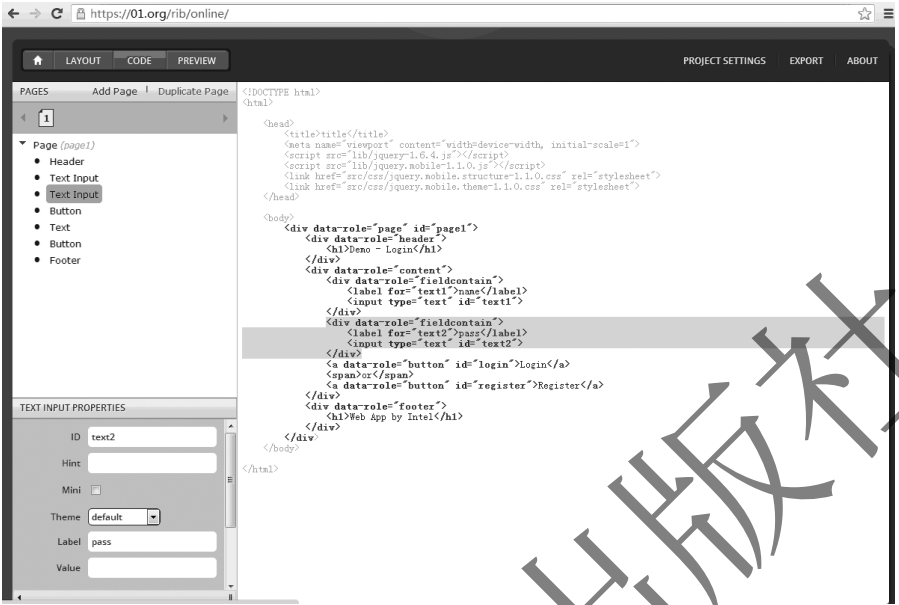

图 3-17　第一页代码

```
</head>

<body>
    <div data-role = "page" id = "page1">
        <div data-role = "header">
            <h1>Demo - Login</h1>
        </div>
        <div data-role = "content">
            <div data-role = "fieldcontain">
                <label for = "text1">name</label>
                <input type = "text" id = "text1">
            </div>
            <div data-role = "fieldcontain">
                <label for = "text2">pass</label>
                <input type = "text" id = "text2">
            </div>
            <a data-role = "button" id = "login">Login</a>
            <span>or</span>
            <a data-role = "button" id = "register">Register</a>
        </div>
        <div data-role = "footer">
            <h1>Web App by Intel</h1>
        </div>
```

```
    </div>
</body>

</html>
```

再切换成 PREVIEW，如图 3-18。

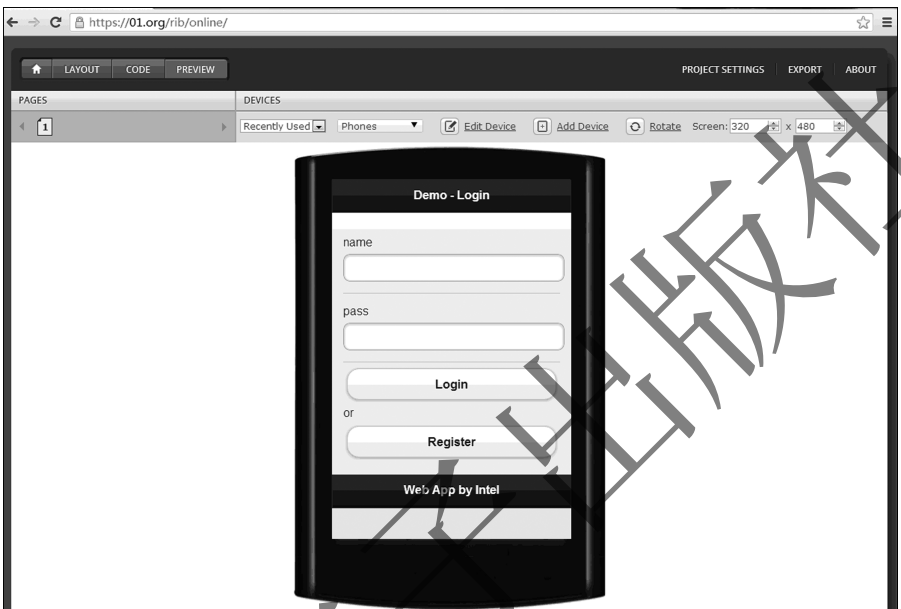

图 3-18 第一页预览

2. 制作第二页

(1)选择 LAYOUT，在左上部，点击 Add Page，弹出新页面设定，只选择 Header，Add Page。

(2)选择 header，左侧的属性栏修改 Text 内容为 Demo - Register。

(3)然后从右侧的拖拽下列元素到中间 header 下方的区域。

(4)插入 Form Elements/Text Input，设置 Label 为 name，其缺省 ID 是 text3。

(5)插入 Form Elements/Text Input，设置 Label 为 pass，其缺省 ID 是 text4。

(6)插入 Form Elements/Slider，设置 Label 为 age，其缺省 ID 是 slider1。

(7)插入 Form Elements/Radio Group，保留两条，分别设置 Label 为 Male，Female，缺省 name 是 radio1。

(8)插入 Button，设置 Text 为 ok，修改 ID 是 ok。

如图：rib-5.gif。

预览效果，如图 3-20。

3. 制作第三页

(1)选择 LAYOUT，在左上部，点击 Add Page，弹出新页面设定，只选择 Header，Add Page。

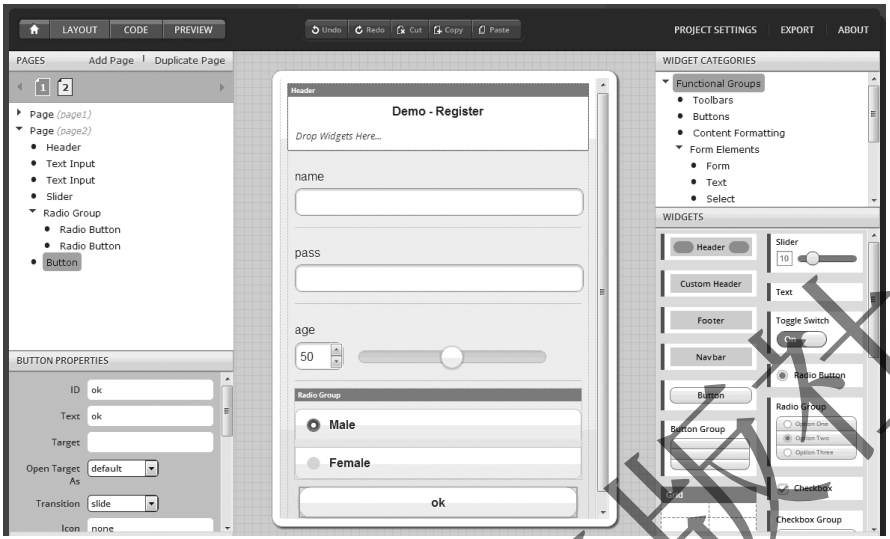

图 3-19　第二页的制作

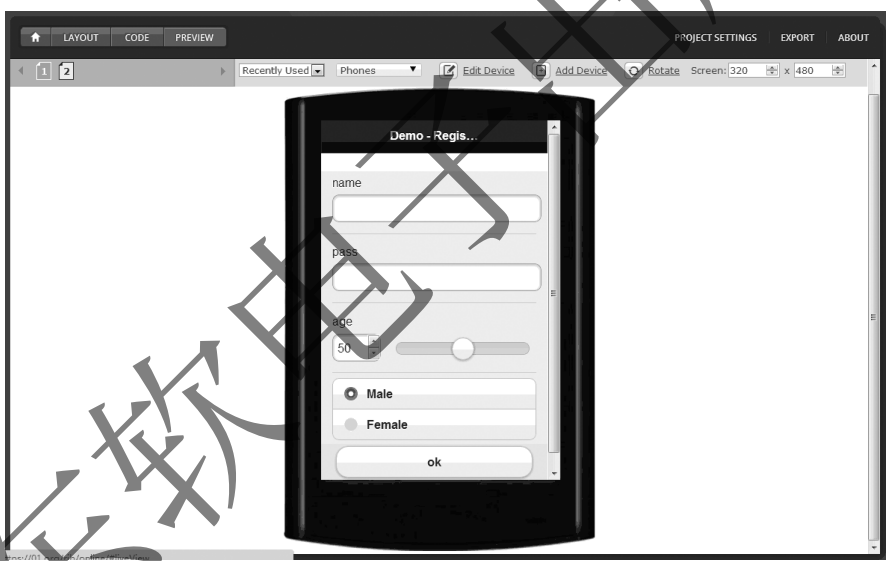

图 3-20　第二页预览

(2)选择 header,左侧的属性栏修改 Text 内容为 Welcome。

(3)插入 Button,设置 Text 为 Logout,修改 ID 是 Logout。

这样三页都做好了。如图 3-21。

预览效果,如图 3-22。

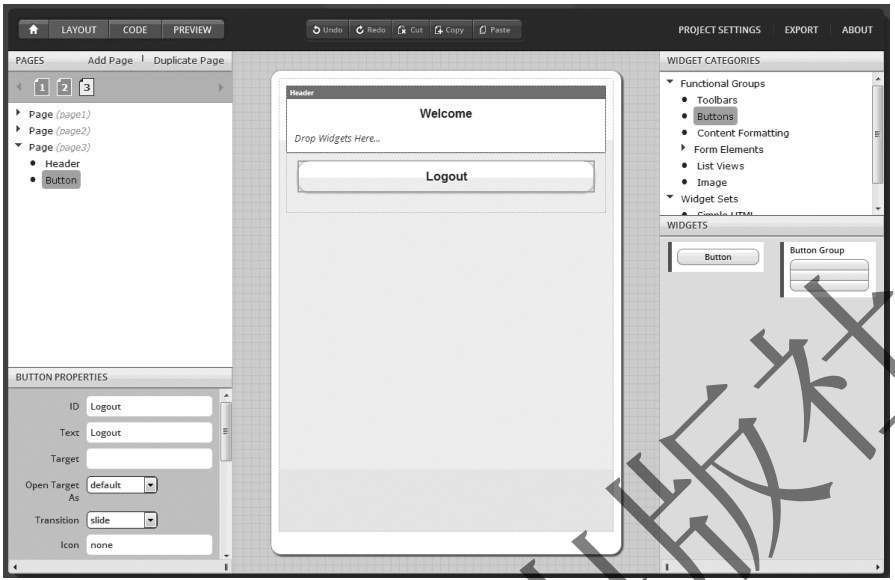

图 3-21　第三页制作

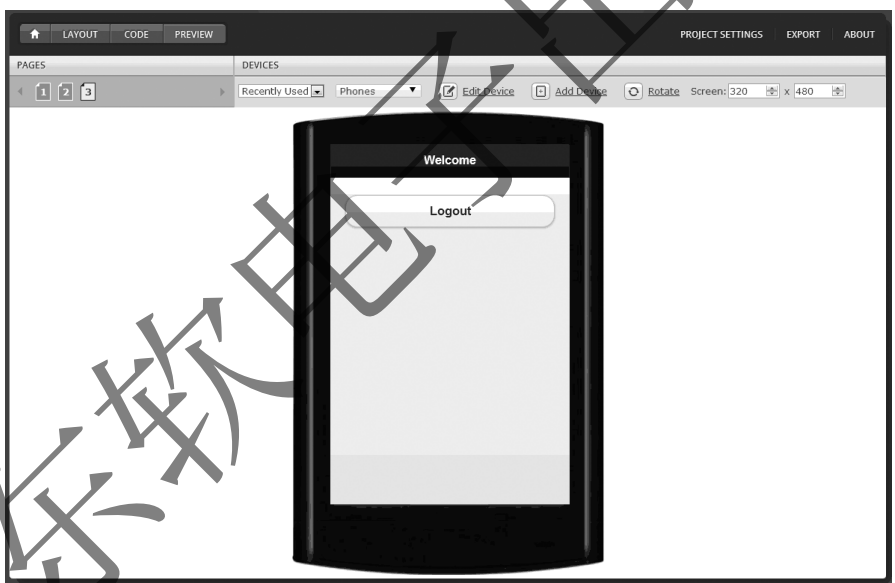

图 3-22　第三页预览

菜单顶部可以设定屏幕的尺寸,本例使用缺省的 320 x 480 像素。点击 code,可以看到已经自动生成了如下代码。

```
<! DOCTYPE html>
<html>

<head>
<title>title</title>
<meta name = "viewport" content = "width = device-width, initial-scale = 1">
```

```
<script src = "lib/jquery - 1.6.4.js"></script>
<script src = "lib/jquery.mobile - 1.1.0.js"></script>
<link href = "src/css/jquery.mobile.structure - 1.1.0.css" rel = "stylesheet">
<link href = "src/css/jquery.mobile.theme - 1.1.0.css" rel = "stylesheet">
</head>

<body>
<div data-role = "page" id = "page1">
<div data-role = "header">
<h1>Demo - Login</h1>
</div>
<div data-role = "content">
<div data-role = "fieldcontain">
<label for = "text1">name</label>
<input type = "text" id = "text1">
</div>
<div data-role = "fieldcontain">
<label for = "text2">pass</label>
<input type = "text" id = "text2">
</div>
<a data-role = "button" id = "login">Login</a>
<span>or</span>
<a data-role = "button" id = "register">Register</a>
</div>
<div data-role = "footer">
<h1>Web App by Intel</h1>
</div>
</div>
<div data-role = "page" id = "page2">
<div data-role = "header">
<h1>Demo - Register</h1>
</div>
<div data-role = "content">
<div data-role = "fieldcontain">
<label for = "text3">name</label>
<input type = "text" id = "text3">
</div>
<div data-role = "fieldcontain">
<label for = "text4">pass</label>
<input type = "text" id = "text4">
</div>
<div data-role = "fieldcontain" id = "slider1">
```

```
<label for = "slider1-range">age</label>
<input type = "range" id = "slider1-range" value = "50" min = "0" max = "100" step = "1">
</div>
<fieldset data-role = "controlgroup">
<legend></legend>
<input type = "radio" id = "radio1" name = "radio1" checked = "checked">
<label for = "radio1">Male</label>
<input type = "radio" id = "radio2" name = "radio1">
<label for = "radio2">Female</label>
</fieldset>
<a data-role = "button" id = "ok">ok</a>
</div>
</div>
<div data-role = "page" id = "page3">
<div data-role = "header">
<h1>Welcome</h1>
</div>
<div data-role = "content">
<a data-role = "button" id = "Logout">Logout</a>
</div>
</div>
</body>

</html>
```

点击右侧的 Project Settings,起名为 rib。

点击右侧的 EXPORT,所有网页相关文件被打包成 rib.zip 文件下载到本地。

至此,UI 设计好了,修改一下 index.html,在文件的最后增加这些代码:

```
<script>
$(function(){
    //第一页的 button 功能绑定
    // $ ('#login'). click 是 jQuery 给节点绑定点击功能的方法,里面的函数在点击时触发
    $ ('#login'). click(function(){
        var name = $ ('#text1'). val();
        var pass = $ ('#text2'). val();

        //用户名和密码都储存在本地存储里
        var ls = localStorage;
        if(name = = ls. name && pass = = ls. pass){
            alert('login success! ');
            // $ . mobile. changePage 是 jquery. mobile 的换页命令
            $ . mobile. changePage( $ ('#page3'));
```

```
        }else{
            alert('login fail! ');
        }
    });
    $('#register').click(function(){
        $.mobile.changePage($('#page2'));
    });

    //第二页的 button 功能绑定
    $('#ok').click(function(){
        var ls = localStorage;
        var name = $('#text3').val();
        if(name){
            ls.name = name;
            ls.pass = $('#text4').val();
            ls.age = $('#slider1-range').val();
            alert('register success! ');
            $.mobile.changePage($('#page1'));
        }else{
            alert('you must input a name! ');
        }
    });

    //第三页的 button 功能绑定
    //注意 Logout 大小写
    $('#Logout').click(function(){
        $.mobile.changePage($('#page1'));
    });
});
</script>
```

最后，再修改一下第 25 行 pass 输入框的 type 属性为 password。

```
<input type="password" id="text2">
```

好了，一个简单的 Web 应用做好了。

第一页，输入用户名和密码，如果正确，提示 login success! 错误则提示 login fail!，或者点击 register 进入第二页。

第二页，输入用户名和密码，设置年龄和性别，点击 ok。注册成功，提示 register success! 进入第三页，错误则提示 you must input a name!

第三页，只有一个退回第一页的 Logout 按钮。

（大家可以试试在第三页显示用户名、年龄和性别）