

疯狂

前端开发讲义

jQuery+AngularJS+Bootstrap

前端开发实战

李刚 编著

疯狂源自梦想

技术成就辉煌

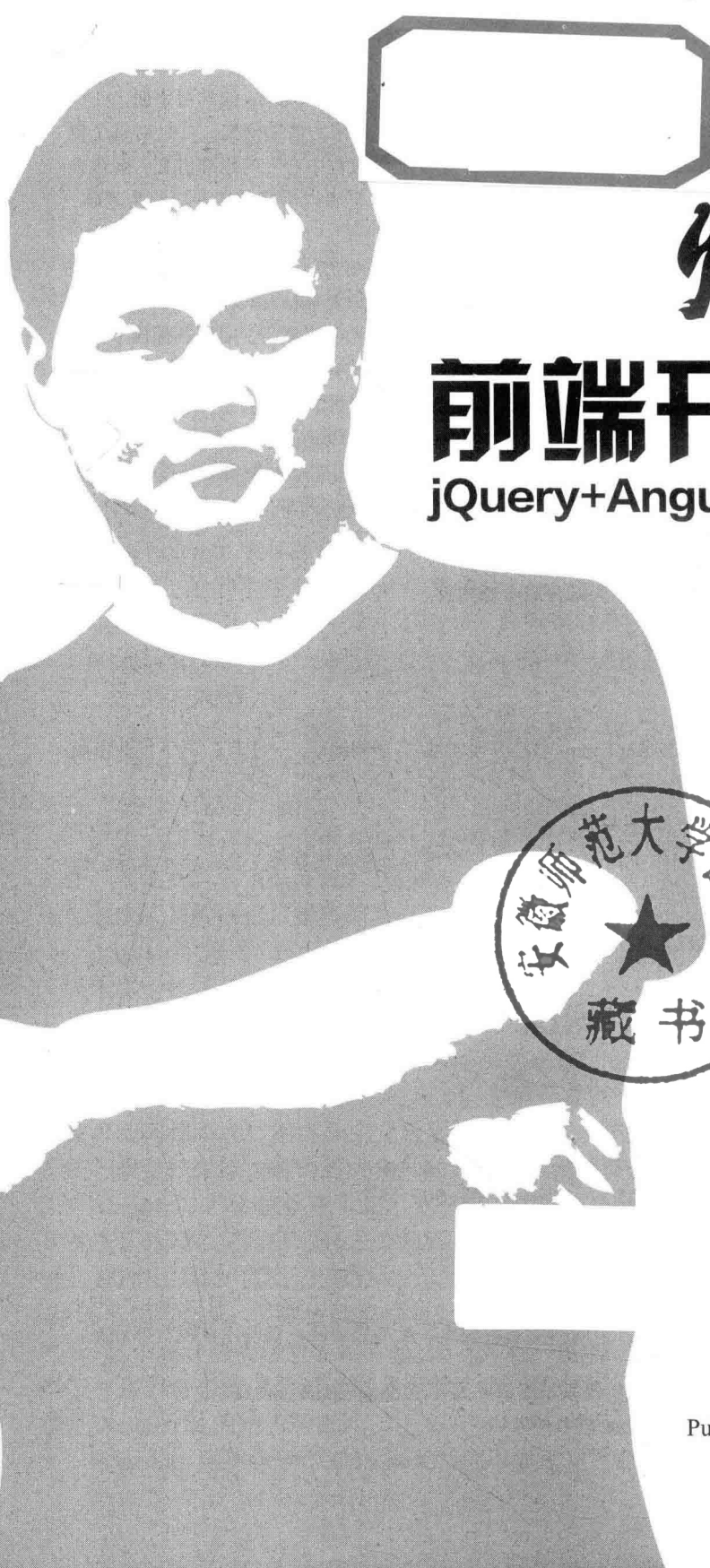
疯狂源自梦想
技术成就辉煌



中国工信出版集团



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn



疯狂

前端开发讲义

jQuery+AngularJS+Bootstrap

前端开发实战

李刚 编著



电子工业出版社

Publishing House of Electronics Industry

北京•BEIJING

内 容 简 介

本书基于《疯狂 Ajax 讲义 (第 3 版)》部分升级而来, 全书升级了 HTML 5.1 支持的 XMLHttpRequest, jQuery 升级到 3.1。本书的重点是新加入的两个目前十分主流的前端框架: AngularJS 和 Bootstrap。本书详细、全面地介绍了 AngularJS 和 Bootstrap 的知识, 由于这两个框架是本书的重点, 因此本书花了近 400 多页来介绍它们的功能和用法, 这部分内容独立出来完全可以作为 AngularJS 和 Bootstrap 的学习手册。

全书主要就是讲解 jQuery 3.1、AngularJS 1.6、Bootstrap 3.3 这三个最常用的前端框架, 并针对每个框架都提供了实用的案例, 让读者理论联系实际。这部分内容是“疯狂软件教育中心”的标准讲义, 既包含了实际前端开发的重点和难点, 也融入了大量学习者的学习经验和感悟。笔者以丰富的授课经验为基础, 在讲解时深入浅出, 力求使读者真正掌握前端开发的精髓。

本书最后提供了两个综合性案例: 图书管理系统和电子拍卖系统, 这两个项目都综合利用了 jQuery、AngularJS、Bootstrap 前端开发技术, 并在后端采用了最流行、最规范的轻量级 Java EE 架构: 控制器层→业务逻辑层→数据持久化层。这两个案例对实际项目具有极好的指导价值和借鉴意义。案例既提供了 IDE 无关的、基于 Ant 管理的项目源码, 也提供了基于 Eclipse IDE 的项目源码, 最大限度地满足读者的需求。如果在阅读本书时遇到任何技术问题, 都可登录 <http://www.crazyit.org> 与本书庞大的读者群交流。

本书并非针对零基础的读者, 本书不再包含 HTML、CSS、JavaScript 相关知识, 这些知识是阅读本书的基础。本书适合有初步 HTML、CSS、JavaScript 基础的读者, 或对企业应用前端开发不太熟悉的开发人员。如果您已经掌握本书上篇:《疯狂 HTML 5/CSS 3/JavaScript 讲义》的内容, 将非常适合阅读本书。

未经许可, 不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有, 侵权必究。

图书在版编目 (CIP) 数据

疯狂前端开发讲义: jQuery+AngularJS+Bootstrap 前端开发实战 / 李刚编著. —北京: 电子工业出版社, 2017.10

ISBN 978-7-121-32680-6

I. ①疯… II. ①李… III. ①网页制作工具—JAVA 语言—程序设计②网页制作工具—超文本标记语言—程序设计 IV. ①TP393.092.2②TP312.8

中国版本图书馆 CIP 数据核字 (2017) 第 223292 号

策划编辑: 张月萍

责任编辑: 牛 勇

印 刷: 三河市鑫金马印装有限公司

装 订: 三河市鑫金马印装有限公司

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱

邮编: 100036

开 本: 787×1092 1/16 印张: 33.5

字数: 900 千字

版 次: 2017 年 10 月第 1 版

印 次: 2017 年 10 月第 1 次印刷

定 价: 79.00 元

凡所购买电子工业出版社图书有缺损问题, 请向购买书店调换。若书店售缺, 请与本社发行部联系, 联系及邮购电话: (010) 88254888, 88258888。

质量投诉请发邮件至 zlts@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式: 010-51260888-819, faq@phei.com.cn。



前言

现在企业应用开发越来越重视前端开发，很多企业已经独立出一个专门的岗位：前端开发工程师，前端开发工程师专门负责企业应用的前端编程。前端工程师与美工不同，美工负责的是应用界面的规划和设计，他们只负责做出静态的界面构图，有时也包括规划界面交互效果，但美工通常并不懂编程实现；而前端开发工程师则负责使用 HTML 5、CSS、JavaScript 以及各种前端框架来实现整个前端界面：包括应用界面的搭建、用户交互的实现，这些内容其实都属于 JavaScript 编程。

有些初学者往往容易混淆美工和前端开发工程师的区别：他们以为会用网页设计工具制作网页就算前端开发工程师。实际上会用网页设计工具既算不上美工，也算不上前端开发工程师。美工的重点在“美”，他们需要有扎实的美术功底，他们可用任何工具来“设计”静态界面，甚至可直接在纸上绘制界面草图，因此网页制作工具只是其中的工具之一；而前端开发工程师往往并不使用网页设计工具，他们通常会使用专门的代码编写工具，他们用 HTML、CSS 来编写静态界面，再使用 JavaScript 或各种前端框架来实现界面的动态交互效果。因此，前端开发工程师的本质依然是程序员，他们常用 JavaScript 以及各种 JavaScript 框架，如 jQuery 和 AngularJS 等。

这里必须说明，前端开发并不简单，原因主要有如下两点：

- JavaScript 作为一门编程语言，有其自身的优势和特点，要想真正掌握并熟练使用它，有一定的难度。
- 各浏览器之间存在一定的差异（虽然这种差异正在被相应的规范减少），而且同一个浏览器的不同版本之间也存在较大差异（如 IE 8、IE 9、IE 10 等），这同样给广大开发者带来了较大的困难。

正因为前端开发存在的复杂性，使得前端开发的相关框架也相当活跃，除了最主流的 jQuery 之外，其他还有如 AngularJS、Bootstrap 等。尤其是 jQuery，目前它已经超出一般框架的概念，甚至变成了某种规范，很多前端框架底层都会借鉴或依赖 jQuery。

AngularJS 则是一个非常实用的前端框架，这个框架不是一种简单的工具，它甚至会强迫开发者采用一种优雅的前端架构，这也是笔者对这个框架情有独钟的原因。

对于初级甚至中级的前端开发者来说，他们有一定的 JavaScript 代码功底，如果单纯交代他们实现一个前端功能，他们可能也可以实现出来，但他们的实现风格要么乱七八糟，要么“随心所欲”：今天可能这样实现，明天可能会那样实现——如果整个前端都由一个开发者完成，这样做可能问题不大。但实际企业应用的前端往往需要多人协作开发，这时各前端开发者的风格不统一就会带来巨大的沟通成本，从而导致项目进度受阻。

AngularJS 的价值就在于此，AngularJS 可以极好地规范前端开发的风格，AngularJS 对前端开发进行分层，它将前端开发分为 Controller 层、Service 层、DAO 层和 Model 层，Model 对象则与 HTML 页面（视图）上 HTML 元素进行双向绑定，这样开发者可通过 Controller 调用 Service、DAO 与后端交互，获取后端数据之后，只要修改其中 Model 对象的值，视图页面也会随之动态改变。这个设计架构层次非常清晰，而且具有一定的“强制性”，整个前端团队一旦采用 AngularJS 框架，那么整个前端开发风格会变得简单、清晰，所有团队成员都能采用一致的开发风格，这就是 AngularJS 的魅力所在。

Bootstrap 则更像一个 CSS 框架，使用起来非常方便。对于大部分前端开发者，最令人头疼

的可能并不是“如何实现”某个界面，而是不清楚到底“要实现什么”界面？怎样的界面才能算得上优雅、美观？而 Bootstrap 正好解决了这个痛点，该框架提供了大量优雅、美观的 CSS 样式，开发者直接应用这些 CSS 样式即可实现优雅、美观的页面。从这个角度来看，Bootstrap 的上手非常简单，甚至不要求开发者掌握 JavaScript 知识，只要开发者略懂 CSS 样式即可，因此 Bootstrap 完全是真正简单且强大的前端框架。

本书有什么特点



本书只是一本介绍前端开发的图书，不是一本关于所谓“思想”的书，更不是一本看完之后可以“吹嘘、炫耀”的书——因为本书并没有堆砌一堆“深奥”的新名词、一堆“高深”的思想，本书保持了“疯狂 Java 体系”的一贯风格：操作步骤详细，编程思路清晰，语言平实。

本书带给读者的只是 9 个字：“看得懂、学得会、做得出”，本书并不能让你认识一堆新名词，只帮助你掌握扎实的企业前端开发基础。对于本书，光“看”是不够的，一定要“做”，阅读本书的同时，应该把所有知识点的配套示例都做出来，这样才能真正掌握本书的知识。

无论如何，读者在阅读本书时遇到知识上的问题，都可以登录疯狂 Java 联盟 (<http://www.crazyit.org>) 与广大 Java 学习者交流，笔者也会通过该平台与大家一起交流、学习。访问 www.broadview.com.cn/32680 下载相关资源。

除此之外，本书还有如下几个特点。

1. 通俗易懂，适合自学

该书吸收了大量学习者的学习体会和心得，并重点讲解了学习过程中难以理解和掌握的知识，降低了学习者的学习难度。

2. 知识丰富，内容全面

本书全面、详细地介绍了 jQuery 3.1、AngularJS 1.6、Bootstrap 3.3 三个框架，它们是企业开发中最主流的前端框架，具有很强的代表性。掌握本书内容即可具备扎实的前端开发功底。

3. 深入实用，实践性强

本书并不是一本前端开发的入门图书，本书全面、深入地介绍了企业开发中最主流、最具代表性的前端框架，并将它们真正融入 Java 企业应用开发中，这对实际企业应用开发具有极好的指导意义。

本书写给谁看



本书并非针对零基础的读者，如果您具有 HTML 5、CSS 3、JavaScript 基础，认真学习此书即可让您成为前端开发的实战型人才；如果已经阅读过本书上篇：《疯狂 HTML 5/CSS/JavaScript 讲义》，阅读本书非常合适。如果完全没有 HTML、CSS、JavaScript 基础知识，建议您暂时不要购买、阅读此书。

2017-07-28

目 录 CONTENTS

第1章 前端开发与 Ajax 技术.....1	
1.1 重新思考 Web 应用.....2	
1.1.1 应用系统的发展史.....2	
1.1.2 传统 Web 应用的优势和缺点.....3	
1.2 重新设计 Web 应用.....4	
1.2.1 富 Internet 应用.....4	
1.2.2 改进的服务器通信.....5	
1.2.3 丰富的客户端交互.....6	
1.3 前端开发介绍.....7	
1.3.1 XMLHttpRequest 简介.....7	
1.3.2 前端开发的核心技术.....7	
1.3.3 前端 Ajax 的特征.....9	
1.3.4 Ajax 带来的优势.....10	
1.4 前端开发体验: Ajax 聊天室.....11	
1.4.1 实现业务逻辑组件.....12	
1.4.2 注册、登录控制器.....15	
1.4.3 注册、登录视图.....16	
1.4.4 异步发送请求.....17	
1.4.5 聊天控制器.....18	
1.4.6 解析服务器响应.....21	
1.4.7 何时发送请求.....21	
1.5 前端开发的技术难点.....24	
1.6 本章小结.....25	
第2章 HTML 5 增强的 XMLHttpRequest 对象.....26	
2.1 XMLHttpRequest 对象的方法和属性.....27	
2.1.1 XMLHttpRequest 对象的方法.....27	
2.1.2 XMLHttpRequest 对象的属性.....30	
2.1.3 XMLHttpRequest 对象的事件.....32	
2.2 发送请求.....33	
2.2.1 发送简单请求.....33	
2.2.2 发送 GET 请求.....34	
2.2.3 发送 POST 请求.....36	
2.2.4 发送 XML 请求.....37	
2.2.5 发送表单数据.....40	
2.2.6 发送 Blob 对象.....42	
2.3 处理响应.....44	
2.3.1 处理响应的时机.....44	
2.3.2 使用文本响应.....44	
2.3.3 使用 JSON 响应.....45	
2.3.4 使用 Blob 或 ArrayBuffer 响应.....48	
2.4 XMLHttpRequest 对象的运行周期.....50	
2.5 跨域请求和安全性问题.....50	
2.5.1 跨域请求.....50	
2.5.2 安全性问题.....53	
2.5.3 性能问题.....54	
2.6 本章小结.....56	
第3章 jQuery 库详解.....57	
3.1 jQuery 入门.....58	
3.1.1 理解 jQuery 的设计.....58	
3.1.2 下载和安装 jQuery.....59	
3.1.3 让 jQuery 与其他 JavaScript 库共存.....60	
3.2 获取 jQuery 对象.....61	
3.2.1 jQuery 核心函数.....61	
3.2.2 jQuery 与 jQuery.holdReady.....62	
3.2.3 以 CSS 选择器访问 DOM 元素.....63	
3.2.4 以伪类选择器访问 DOM 元素.....65	
3.2.5 表单相关的选择器.....70	
3.3 jQuery 操作类数组的工具方法.....72	
3.3.1 过滤相关方法.....74	
3.3.2 仿 DOM 导航查找的相关方法.....76	
3.3.3 串联方法.....78	
3.4 jQuery 支持的方法.....79	
3.4.1 jQuery 命名空间的方法.....80	
3.4.2 数据存储的相关方法.....83	
3.4.3 操作属性的相关方法.....84	
3.4.4 操作 CSS 属性的相关方法.....86	
3.4.5 操作元素内容的相关方法.....89	
3.4.6 操作 DOM 节点的相关方法.....90	
3.5 jQuery 事件相关方法.....96	
3.5.1 绑定事件处理函数.....96	
3.5.2 特定事件相关的方法.....98	
3.5.3 事件对象.....99	
3.6 动画效果相关的方法.....100	
3.6.1 简单动画和复杂动画.....100	
3.6.2 操作动画队列.....103	
3.7 jQuery 的回调支持.....104	

3.7.1	回调支持的基本用法	104	5.2.3	AngularJS 表达式的容错性	152
3.7.2	创建 Callbacks 对象支持的选项	106	5.2.4	AngularJS 表达式与 JavaScript 表达式	152
3.8	Ajax 相关方法	108	5.3	模块与控制器	153
3.8.1	三个工具方法	108	5.3.1	模块的加载	153
3.8.2	使用 load 方法	109	5.3.2	控制器初始化\$scope 对象	155
3.8.3	jQuery.ajax(options)与 jQuery.ajaxSetup(options)	111	5.3.3	\$rootScope 作用域	157
3.8.4	使用 get/post 方法	112	5.3.4	\$watch 方法的使用	158
3.9	jQuery 的 Deferred 对象	115	5.4	过滤器	159
3.9.1	jQuery 的异步调用	115	5.4.1	内置过滤器	159
3.9.2	为多个耗时操作指定回调函数	119	5.4.2	在表达式中使用过滤器	160
3.9.3	为普通对象增加 Deferred 接口	119	5.4.3	在指令中使用过滤器	162
3.9.4	jQuery 对象的 promise 方法	120	5.4.4	自定义过滤器	162
3.10	扩展 jQuery 和 jQuery 插件	121	5.5	函数 API	164
3.11	本章小结	122	5.5.1	扩展型函数	164
第4章	基于 jQuery 的应用: 电子相册系统	123	5.5.2	jqLite 函数	168
4.1	实现持久层	124	5.5.3	判断型函数	169
4.1.1	实现持久化类	124	5.6	指令	170
4.1.2	配置 SessionFactory	126	5.6.1	表单相关的指令	170
4.2	实现 DAO 组件	127	5.6.2	表单的输入校验	175
4.2.1	开发通用 DAO 组件	127	5.6.3	事件相关的指令	178
4.2.2	DAO 接口定义	130	5.6.4	流程控制相关的指令	179
4.2.3	完成 DAO 组件的实现类	131	5.6.5	绑定相关的指令	183
4.3	实现业务逻辑层	132	5.6.6	DOM 及 DOM 状态相关指令	187
4.3.1	实现业务逻辑组件	132	5.6.7	自定义指令	193
4.3.2	配置业务逻辑组件	134	5.6.8	自定义指令的 scope 属性	195
4.4	实现客户端调用	135	5.6.9	自定义指令的 transclude 属性	197
4.4.1	访问业务逻辑组件	135	5.6.10	自定义指令的 link 和 compile 属性	198
4.4.2	处理用户登录	136	5.6.11	自定义指令的 controller 和 controllerAs 属性	202
4.4.3	获得用户相片列表	138	5.6.12	自定义指令的 require 属性	203
4.4.4	处理翻页	140	5.7	调用内置服务	205
4.4.5	使用 jQuery 实现文件上传	141	5.7.1	\$animate 服务	205
4.4.6	加载页面时的处理	144	5.7.2	\$cacheFactory 服务	207
4.5	本章小结	145	5.7.3	\$compile 服务	209
第5章	AngularJS 详解	147	5.7.4	\$document、\$window、\$timeout、\$interval 和 \$rootElement	211
5.1	AngularJS 入门	148	5.7.5	\$parse 服务	214
5.1.1	理解 AngularJS 的基本设计	148	5.7.6	\$interpolate 服务	214
5.1.2	下载和安装 AngularJS	149	5.7.7	\$log 服务	215
5.2	表达式	150	5.7.8	\$q 服务	216
5.2.1	简单表达式	150	5.7.9	\$templateCache 服务	218
5.2.2	复合对象表达式	151			

5.8	自定义服务	219	6.3.2	Less 的两种用法	270
5.8.1	使用 factory()方法创建自定义服务	219	6.3.3	Less 的变量和运算符	274
5.8.2	使用 service()方法创建自定义服务	220	6.3.4	mixin	274
5.8.3	使用 provider()方法创建自定义 服务	221	6.3.5	内嵌规则	275
5.8.4	使用 \$provide 服务创建自定义服务	223	6.3.6	Bootstrap 网格系统的变量和 mixin	276
5.8.5	在过滤器中使用自定义服务	225	6.4	Bootstrap 排版相关样式	278
5.9	依赖注入	226	6.4.1	标题元素和样式	278
5.9.1	依赖注入机制简介	226	6.4.2	段落	279
5.9.2	使用 \$injector 对象获取组件	228	6.4.3	增强的 HTML 元素	280
5.9.3	隐式依赖注入	230	6.4.4	对齐	282
5.9.4	行内数组式依赖注入	230	6.4.5	改变大小写	283
5.9.5	标记式依赖注入	231	6.4.6	列表	283
5.10	与服务器交互	232	6.5	表格相关样式	286
5.10.1	\$http 服务	232	6.5.1	基础表格	286
5.10.2	\$http 的快捷方法	235	6.5.2	条纹表格	287
5.10.3	使用 \$http 上传文件	237	6.5.3	边框表格	287
5.10.4	使用 \$resource 服务	239	6.5.4	鼠标高亮	288
5.11	多视图和路由	240	6.5.5	紧凑型表格	289
5.11.1	使用 \$routeProvider 配置路由规则	240	6.5.6	响应式表格	289
5.11.2	创建多视图	242	6.5.7	表格行状态	290
5.11.3	通过路由切换视图	244	6.6	图片和图标	291
5.11.4	使用 \$location 实现多视图切换	246	6.6.1	图片相关样式	291
5.12	使用 ui-router 框架实现多视图	248	6.6.2	图标	293
5.12.1	ui-router 的下载和安装	248	6.7	辅助样式	294
5.12.2	使用 \$stateProvider 配置路由	248	6.7.1	情境背景色	294
5.12.3	多视图切换与 \$state	250	6.7.2	情境文本颜色	295
5.12.4	多个命名的嵌套视图	252	6.7.3	关闭按钮和三角箭头	295
5.13	本章小结	255	6.7.4	快速浮动	296
			6.7.5	显示或隐藏内容	296
			6.7.6	屏幕阅读器和键盘导航	297
			6.7.7	图片替换	297
第 6 章	Bootstrap 全局样式	256	6.8	响应式布局相关样式	297
6.1	Bootstrap	257	6.8.1	显示/隐藏相关样式	298
6.1.1	Bootstrap 简介	257	6.8.2	打印相关样式	299
6.1.2	下载和安装 Bootstrap	257	6.9	表单相关样式	299
6.2	网格布局	260	6.9.1	基础表单	299
6.2.1	网格布局基础	261	6.9.2	行内表单	300
6.2.2	多余的列另起一行	264	6.9.3	水平表单	302
6.2.3	响应式列重置	264	6.9.4	多选框和单选框	303
6.2.4	单元格偏移	266	6.9.5	表单控件的大小	305
6.2.5	单元格排序	267	6.9.6	静态控件	307
6.2.6	嵌套网格	268	6.9.7	表单控件的状态	309
6.3	Less 和 mixin	269	6.9.8	帮助文本	309
6.3.1	Less 简介	269			

6.9.9 校验状态	310	7.6.1 标签	354
6.9.10 校验状态的图标	311	7.6.2 徽章	355
6.10 本章小结	313	7.7 面板	355
第 7 章 Bootstrap 内置组件	314	7.7.1 面板的基础结构	355
7.1 按钮	315	7.7.2 面板嵌套表格	358
7.1.1 按钮大小	316	7.7.3 面板嵌套列表组	359
7.1.2 按钮状态	317	7.8 巨幕、页头和 Well	361
7.2 下拉菜单	319	7.8.1 巨幕	361
7.2.1 对齐	321	7.8.2 页头	362
7.2.2 禁用菜单项	322	7.8.3 well	362
7.2.3 按钮式下拉菜单	323	7.9 缩略图	363
7.2.4 分裂式按钮下拉菜单	324	7.10 警告框	365
7.2.5 大小	326	7.10.1 警告框基础	365
7.3 按钮组	327	7.10.2 警告框中的链接	367
7.3.1 基本按钮组	327	7.11 进度条	367
7.3.2 工具栏	328	7.11.1 各种样式的进度条	367
7.3.3 控制按钮组的大小	329	7.11.2 带进度值的进度条	369
7.3.4 按钮组嵌套下拉菜单	329	7.11.3 动画效果	370
7.3.5 两端对齐的按钮组	331	7.11.4 多进度效果	371
7.4 输入框组	332	7.12 媒体对象	372
7.4.1 基本输入框组	332	7.12.1 媒体对象的基本组成	372
7.4.2 控制输入框组的大小	334	7.12.2 对齐方式	374
7.4.3 单选框或多选框作为附加元素	335	7.12.3 嵌套媒体对象	375
7.4.4 按钮式下拉菜单作为附加元素	336	7.12.4 媒体对象列表	376
7.4.5 多按钮	337	7.13 列表组	377
7.5 导航	338	7.13.1 列表组基础	378
7.5.1 简单导航的基础样式	338	7.13.2 链接列表组	379
7.5.2 两端对齐	340	7.13.3 按钮列表组	379
7.5.3 嵌套下拉菜单	340	7.13.4 列表项的状态	380
7.5.4 路径导航	341	7.13.5 定制内容	381
7.5.5 基础导航条	342	7.14 本章小结	381
7.5.6 导航条中的品牌图标	344	第 8 章 Bootstrap 的 JS 插件	382
7.5.7 导航条中的按钮	344	8.1 插件库概述	383
7.5.8 导航条中的表单	345	8.1.1 使用插件的两种方式	383
7.5.9 导航条中的文本和链接	346	8.1.2 解决命名冲突	384
7.5.10 导航条中的组件的排列方式	347	8.2 对话框	384
7.5.11 设置导航条的位置	347	8.2.1 静态对话框	384
7.5.12 响应式导航条	349	8.2.2 使用 data-* 属性弹出对话框	387
7.5.13 分页导航	351	8.2.3 使用 JS 弹出对话框	388
7.5.14 控制分页导航的大小	352	8.2.4 对话框事件	389
7.5.15 翻页导航	352	8.2.5 基于事件源改变对话框内容	390
7.6 标签和徽章	353	8.3 下拉菜单	392

8.3.1	使用 data-*属性触发下拉菜单	392
8.3.2	使用 JS 触发下拉菜单	393
8.3.3	下拉菜单事件	394
8.4	滚动监听	395
8.4.1	通过 data-*属性实现滚动监听	395
8.4.2	使用 JS 实现滚动监听	397
8.5	标签页	398
8.5.1	静态标签页	398
8.5.2	使用 data-*属性切换标签页	399
8.5.3	使用 JS 切换标签页	401
8.5.4	胶囊式标签页	402
8.5.5	标签页事件	403
8.6	工具提示	404
8.6.1	使用 data-*属性和 JS 触发工具提示	405
8.6.2	工具提示支持的属性	406
8.6.3	工具提示的事件	407
8.7	弹出框	408
8.7.1	使用 data-*属性和 JS 触发弹出框	409
8.7.2	焦点触发的弹出框	411
8.7.3	弹出框支持的属性	411
8.7.4	弹出框的事件	412
8.8	警告框	412
8.8.1	使用 data-*属性关闭警告框	412
8.8.2	使用 JS 关闭警告框	413
8.8.3	警告框事件	413
8.9	按钮	414
8.9.1	切换按钮状态	414
8.9.2	单选按钮或多选按钮	414
8.9.3	使用 JS 方法改变按钮文本	416
8.10	折叠插件	416
8.10.1	简单折叠效果	416
8.10.2	手风琴效果	417
8.10.3	使用 JS 触发折叠元素	419
8.10.4	折叠插件的相关事件	420
8.11	轮播图	420
8.11.1	静态轮播图	420
8.11.2	通过 data-*属性激活轮播图	422
8.11.3	通过 JS 触发轮播图	424
8.11.4	轮播图事件	425
8.12	本章小结	426

第 9 章	Angular+Bootstrap 整合开发: 图书管理系统	427
9.1	总体说明和概要设计	428
9.1.1	系统的总体架构设计	428
9.1.2	数据库设计	429
9.2	实现 Hibernate 持久化类	430
9.2.1	设计 Domain Object	430
9.2.2	实现 Domain Object	431
9.3	DAO 层实现	435
9.3.1	DAO 的基础配置	435
9.3.2	实现 DAO 组件	436
9.3.3	部署 DAO 组件	437
9.4	业务逻辑层实现	438
9.4.1	设计业务逻辑组件	439
9.4.2	依赖注入 DAO 组件	441
9.4.3	业务逻辑组件的异常处理	441
9.4.4	实现业务逻辑组件	442
9.4.5	事务管理	443
9.4.6	配置业务层组件	444
9.5	前端整合开发	445
9.5.1	定义 AngularJS 路由	445
9.5.2	Spring MVC 控制器的异常处理	447
9.5.3	管理图书种类	447
9.5.4	修改图书种类	451
9.5.5	管理图书	453
9.5.6	修改图书	457
9.5.7	图书入库	459
9.5.8	销售图书	465
9.6	本章小结	470
第 10 章	jQuery+Bootstrap 整合开发: 电子拍卖系统	471
10.1	总体说明和概要设计	472
10.1.1	系统的总体架构设计	472
10.1.2	数据库设计	473
10.2	实现 Hibernate 持久化类	474
10.2.1	设计 Domain Object	474
10.2.2	实现 Domain Object	475
10.3	DAO 层实现	479
10.3.1	DAO 的基础配置	480
10.3.2	实现 DAO 组件	481
10.3.3	部署 DAO 组件	484
10.4	业务逻辑层实现	485

10.4.1	设计业务逻辑组件	485	10.6	前端整合开发	500
10.4.2	依赖注入 DAO 组件	487	10.6.1	定义系统首页	500
10.4.3	业务逻辑组件的异常处理	488	10.6.2	浏览所有流拍物品	502
10.4.4	处理用户竞价	489	10.6.3	处理用户登录	503
10.4.5	判断拍卖物品状态	491	10.6.4	管理物品	508
10.4.6	事务管理	492	10.6.5	管理物品种类	513
10.4.7	配置业务层组件	493	10.6.6	查看竞得物品	516
10.5	开发前端 JSON 接口	494	10.6.7	查看自己的竞价记录	518
10.5.1	初始化 Spring 容器	494	10.6.8	浏览拍卖物品	519
10.5.2	开发 Spring MVC 控制器	496	10.6.9	参与竞价	522
10.5.3	处理前端权限控制	498	10.7	本章小结	526

第 1 章

前端开发与 Ajax 技术

本章要点

- ✎ C/S 模式应用的结构和缺点
- ✎ B/S 模式应用的结构和优势
- ✎ 传统 Web 应用的不足
- ✎ 如何改进传统的 Web 应用
- ✎ 现代 Web 应用与前端开发
- ✎ 改进的通信方式和增强的 UI 界面
- ✎ 体验前端 Ajax 开发
- ✎ 使用 Servlet 生成文本响应
- ✎ 使用 JSP 生成文本响应
- ✎ 获取服务器的响应内容
- ✎ 通过 DOM 加载服务器响应
- ✎ 前端开发的技术难点

1.1 重新思考 Web 应用

传统的 Web 应用经过多年的发展，在很多方面都是相当完善的。特别是 Java EE、PHP 等技术的广泛应用，使得目前绝大部分应用都采用了基于请求—响应的 Web 架构。

1.1.1 应用系统的发展史

早期应用软件系统大都采用 C/S（客户机/服务器模式）结构，C/S 结构的软件分为客户机和服务器两层。客户机不是毫无运算能力的输入/输出设备，在客户端需要部署大量的应用程序，而且可能还具有一定的数据存储能力。

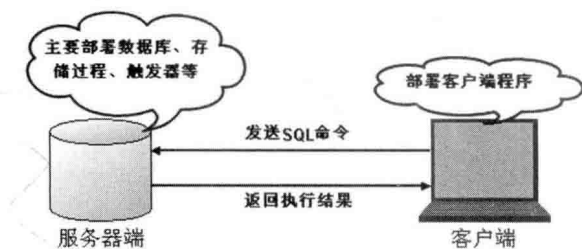


图 1.1 C/S 结构的应用

C/S 结构应用的服务器端通常主要安装数据库管理系统，当然也可能包含一些业务逻辑实现（这些业务逻辑实现通常以函数、存储过程和触发器的形式存在）。通过把软件系统的计算和数据合理地分配在客户机和服务器两端，可以有效地降低网络通信量和服务器运算量。

C/S 结构应用的结构图如图 1.1 所示。

对于 C/S 结构的应用，因为可以直接在客户端部署应用程序，所以可以让应用的人机交互界面更加友好，并可充分美化应用程序的人机界面。但由于服务器连接个数和数据通信量的限制，这种结构的软件适用于用户数目不多的局域网内。早期的大部分 ERP 软件产品即属于此类结构。

随着 Internet 技术的兴起，B/S（浏览器/服务器模式）结构得到了大规模应用。B/S 结构是对 C/S 结构的一种改进。在这种结构下，应用的业务逻辑完全在应用服务器端实现，用户表现完全在 Web 服务器上实现，客户端只需要浏览器即可进行操作，是一种全新的软件系统结构技术。这种结构是当今应用软件的首选体系结构。

在这种应用结构下，客户端的所有处理请求都以 HTTP 请求形式发送，而服务器端则将响应以 HTML 页面的形式送回客户端，由客户端浏览器负责显示 HTML 页面。B/S 结构应用的结构图如图 1.2 所示。

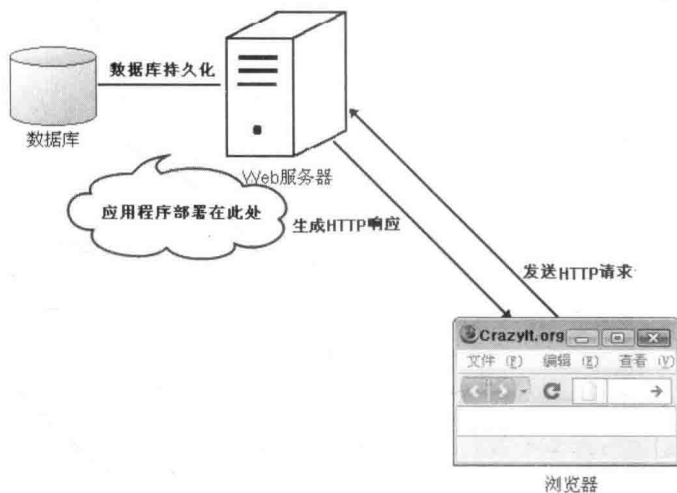


图 1.2 B/S 结构的应用

B/S 结构得到迅速推广是有原因的。在大部分情况下，B/S 结构的应用比 C/S 结构的应用更加优秀，适应性更广。相对而言，B/S 结构的系统有如下方面的优势：

- **数据安全性高。**由于 C/S 结构软件的数据分布特性，客户端所发生的火灾、地震、盗抢、病毒、黑客攻击等都成了可怕的数据杀手。另外，对于集团级的异地软件应用，C/S 结构的软件必须在各地安装多台服务器，并在多台服务器之间进行数据同步。因此，每个数据点上的数据安全都会影响整个应用的数据安全。对于跨区域的大型应用，C/S 结构软件的安全性是令人无法接受的。而对于 B/S 结构的系统，数据集中存放于总部的数据库服务器（服务器数据可以通过多种方式备份存放），客户端不保存任何业务数据和数据库持久化信息，无须进行数据同步，所以这些安全问题自然也就不存在了。
- **数据一致性好。**在 C/S 结构的解决方案里，对于跨区域的大型企业应用，都采用各地安装区域级服务器，然后再进行数据同步的模式。每天必须对这些服务器进行同步之后，总部才可得到最终的数据。由于局部网络故障等原因，可能造成个别数据库无法正常同步。即使都能正常同步，但各服务器往往不能同时同步，因而数据也无法绝对一致，这当然不能用于决策。而对于 B/S 结构的系统，数据是集中存放的，客户端发生的每次数据修改都直接进入到中央数据库，不存在数据一致性的问题。
- **数据实时性好。**对于大型的跨区域应用，C/S 结构不可能随时跟踪各客户端的业务发生情况，因为数据都不是实时更新的，因而看到的数据都是滞后的；而 B/S 结构则不同，因为其数据都是实时存入服务器端的数据库，因而服务器端可以实时看到当前发生的所有业务，能提供更好的企业决策支持。
- **系统更新方便。**软件供应商提供的软件不可能是完美无缺的。即使是一个绝对完美的软件系统，当具体的业务环境发生改变后，系统也应随之改变。因而，必须经常对已部署的软件产品进行维护、升级。对 C/S 结构的软件而言，由于其应用是分布式的，需要手动对每一个节点进行程序安装，所以，即使非常小的程序缺陷都需要很长的时间来重新部署。重新部署时，为了保证各程序版本的一致性，必须暂停业务进行更新（即“休克更新”）。在很多情景下，这是不可忍受的。而 B/S 结构的软件则不同，其应用程序集中于服务器上，各应用节点没有任何程序，应用的更新只需要更新服务器端程序即可，因而可以做到快速的服务响应。

传统的 C/S 结构软件开发工具早期的有 Visual Basic、Visual FoxPro 等，这些开发工具目前基本已经被淘汰。目前使用较少的开发工具如 PowerBuilder、Delphi 等也趋于被淘汰。除早期的一些系统外，现在新开发的系统大部分使用 B/S 结构。

目前的 B/S 结构应用通常都会严格遵守 MVC 模式。MVC 模式将应用分成 Model（M：模型）、View（V：视图）和 Controller（C：控制器）三个部分。MVC 模式分离的数据访问和数据表现，给系统提供了更好的解耦。MVC 架构的核心思想是，将程序分成相对独立而又能协同工作的三个部分。通过使用 MVC 架构，可以降低模块之间的耦合，提供应用的可扩展性。MVC 中的每个组件只关心组件内的逻辑，不应与其他组件的逻辑混合。

Java EE 的出现，则更加规范了 B/S 结构应用的开发。Java EE 推荐将应用分为数据持久层、业务逻辑层和 Web 层，各层之间以松耦合的方式组织在一起。

➤➤ 1.1.2 传统 Web 应用的优势和缺点

B/S 结构虽然是一种非常优秀的结构，但它依然存在如下方面的不足：

- **独占式的请求。**比如一个任务需要多步骤或多选项才能完成。在 HTML 里，一个多步

骤的任务可以在单页内表达出来。但是由于 HTML 的互动性有限，便可能产生一份很长的页面，使用户感到混乱、笨拙而难以使用。或者将多个步骤分成几个页面分别提交，但传统的独占式请求表现为，如果前一个请求没有得到完全响应，则后一个请求不能发送。用户在等待服务器的响应期间，浏览器一片空白。这种独占式请求如图 1.3 所示。

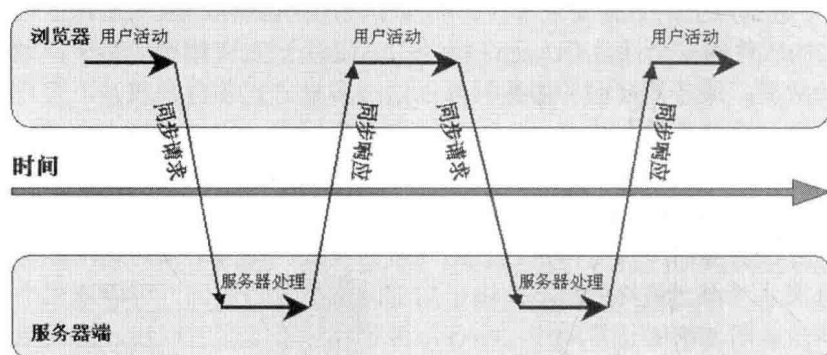


图 1.3 独占式请求的示意图

- 频繁的页面刷新。传统的 Web 应用基本上采用请求一页面的对应模式，每个请求都需要丢弃当前页面来重新加载新页面。频繁的页面刷新不仅让客户处于不连续的体验中，也使服务器的负担加重。
- 简陋的页面。传统 Web 应用因为需要频繁刷新页面，因而不可能制作出具有丰富表现力的页面。具有丰富表现力的页面导致页面文件过大，下载速度较慢，而且页面频繁刷新。一个表现丰富的页面下载需要相当多的时间，但随着请求的提交，又需要重新下载新页面，这样系统开销相当大。因而传统 Web 应用的页面不可能非常出色。

在传统的 Web 应用里，用户总是需要加载新页面时才能提交请求，而提交请求后又需要等待服务器响应，所以如果服务器响应还没有完全结束，则用户只能等待，不能继续发送请求。

在传统的 Web 应用里，每个请求对应一个页面，不管客户端以 POST 还是 GET 方式提交请求，每次请求都会丢弃当前页面，等待服务器生成新页面。在等待期间，旧的页面已经丢弃，新的页面还没有完全生成，整个浏览器将一片空白，而用户什么都做不了，只能等待。这对于用户而言，是一种不连续的体验，感觉非常不好。

1.2 重新设计 Web 应用

传统 Web 应用的不足，一直凸显在用户面前，用户常常抱怨系统的响应速度太慢。除了网络带宽的限制、业务逻辑复杂、硬件设备制约等因素外，频繁的页面刷新，每次响应都必须下载整个响应页面，也导致了响应速度慢。因此，传统 Web 应用必须重新改进。

➤➤ 1.2.1 富 Internet 应用

RIA，是 Rich Internet Application 的缩写，即富 Internet 应用。B/S 结构已成为应用程序开发的默认结构，用户对应用程序复杂性要求日增，Web 应用程序对完成复杂逻辑始终差强人意。

用户与复杂的 Web 应用程序交互时，体验并不能令人满意。Web 模型是基于 HTML 页面的模型，缺少客户端智能机制。传统的 Web 应用几乎无法完成复杂的用户交互（如传统的 C/S 应用程序和桌面应用程序中的用户交互）。因而，Web 在许多应用程序中难以发挥。

为了提升用户体验,出现了一种新类型的 Web 应用,那就是 RIA。这些应用程序吸收了桌面应用程序的反应快、交互性强的优点,改进了 Web 应用程序用户交互差的缺点,可以提供一种更丰富、更具有交互性和响应性的用户体验。

可以将 RIA 架构理解为运行于 B/S 结构上的 C/S 应用。应用客户端采用标准的浏览器,但在浏览器内支持类似 C/S 应用的操作,所以 RIA 应用可以提供强大的功能,让用户有高交互性、高效率响应的体验。同时,RIA 又是基于 Internet 浏览器的应用,所以,用户使用 RIA 非常方便。

使用 RIA,用户无须安装任何的客户端软件,只需拥有浏览器即可。一个典型的富客户端应用是地图。大部分 Web 地图支持鼠标的拖动和放大、缩小。地图随着鼠标的拖动而移动——明显加载了新的数据,但页面本身却无须重新加载。如果鼠标拖动得太远,可能出现部分空白区域,但这种空白只是因为地图区域在加载,而不是整个页面在加载。

当使用鼠标单击地图上的提示点时,地图上会出现该点的更详细的介绍。图 1.4 显示了高德地图中的广州地图。



图 1.4 高德地图应用

RIA 代表着目前 Web 应用的发展趋势,RIA 应用相当于同时吸收了 C/S 结构和 B/S 结构优势后的产物:RIA 应用既能利用 B/S 应用中数据安全、系统更新方便的优势,也能利用 C/S 应用中界面美观、交互丰富的特点,因此 RIA 应用是目前 Web 应用的主流。

为了开发出 RIA 应用,仅依靠传统服务端编程是远远不够的,此时必须对传统的客户端表现界面进行改进——传统的客户端表现界面通常使用静态 HTML 页面来实现,这种静态 HTML 页面主要具有如下两个劣势:

- 无法与服务器实时地进行异步数据通信。
- 传统 HTML 页面界面比较丑陋,缺乏用户交互性。

➤➤ 1.2.2 改进的服务器通信

传统的 Web 应用亟须改变的就是浏览器与服务器的通信方式,传统的同步请求—响应的通信方式无法满足 RIA 应用的网络通信要求。

2005 年开始出现的 Ajax 正是对传统 Web 通信方式的改进,Ajax 使用 XMLHttpRequest 异步发送请求,XMLHttpRequest 发送请求不要求重新加载页面。

XMLHttpRequest 发送请求后,无须等待服务器响应,而是可以继续原来的操作。在服务

器完成响应后，客户端使用 JavaScript 通过 XMLHttpRequest 获取服务器响应。

图 1.5 显示了异步发送请求的示意图。

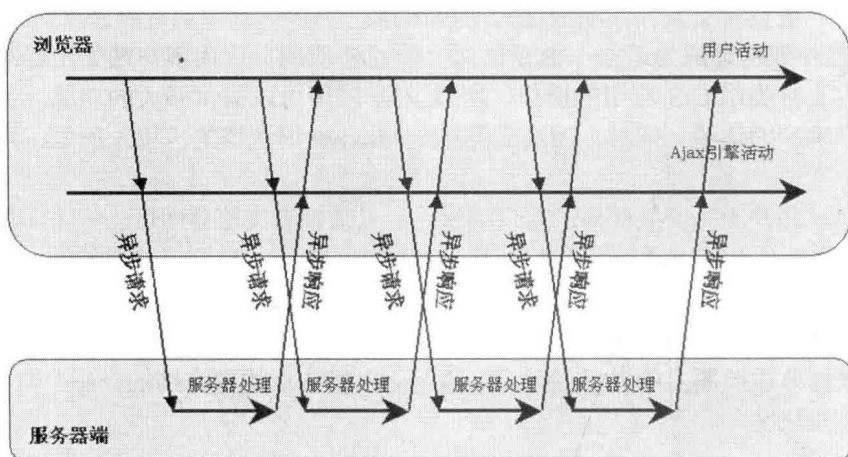


图 1.5 异步发送请求

Ajax 除了异步发送请求外，还能动态加载服务器响应数据。使用 Ajax 能避免频繁刷新页面，服务器响应的是数据，而不是整个页面内容。Ajax 负责获取服务器数据，然后将服务器数据动态加载到浏览器中。

此外，客户端与服务器端的通信还包括如下几种方式：

- **WebSocket 通信技术。**WebSocket 会在服务器与浏览器之间建立基于 TCP 协议的 Socket 连接，该连接一旦建立，浏览器可以实时地向服务器发送数据，服务器也可随时向浏览器发送数据。由此可见，WebSocket 可在浏览器和服务器之间进行双向的数据传输。重要的是，WebSocket 已经成为 HTML 5 技术规范之一，被所有主流浏览器所支持。
- **Server-sent Events 技术。**这是一种简单的服务器推送技术，这种推送技术只允许服务器将数据推送给浏览器，浏览器无法向服务器发送数据，因此这种技术规范也被称为服务器推送技术。与 WebSocket 相比，服务器推送技术更简单一些，但功能也弱一些。一般来说，WebSocket 适用于需要进行复杂双向数据通信的场景。对于简单的服务器数据推送场景，使用服务器推送事件就足够了。
- **COMET 技术。**COMET 技术采用的是长轮询的实现。这种实现方式在每次请求时，服务器端会保持该连接在一段时间内处于打开状态，而不是在响应完成之后就立即关闭。这样做的好处是在连接处于打开状态的时间段内，服务器端产生的数据更新可以被及时地返回给浏览器。当上一个长连接关闭之后，浏览器会立即打开一个新的长连接来继续请求。不过 COMET 技术的实现在服务器端和浏览器端都需要第三方库的支持。但遗憾的是，COMET 技术目前还没有成为 Web 规范。

➤➤ 1.2.3 丰富的客户端交互

除了改进的服务器端通信之外，现代 Web 应用还必须具有如下两个特征：

- 优雅、美观的用户界面。
- 丰富的客户端交互。

为了实现上述两个特点，HTML 5 做出了巨大的努力，HTML 5 添加了大量新的元素和属性，这些新元素和新属性极大地丰富了 Web 的前端开发，而且增强了用户交互性。此外，HTML