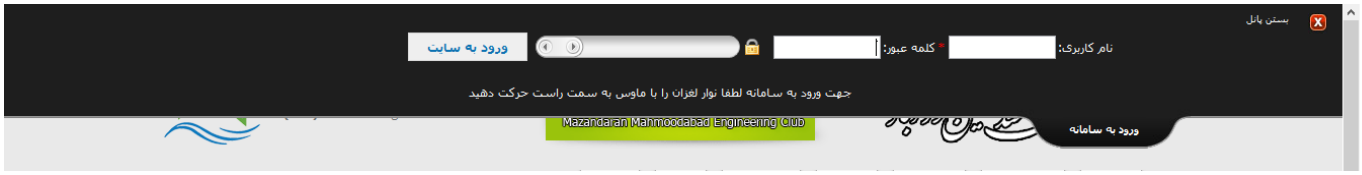
عنوان: **ایجاد قسمت‌های Toggle در سایت با jQuery**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۰/۲۷ ۲۱:۳۵
آدرس: www.dotnettips.info
برچسب‌ها: MVC, jQuery, ASP.Net 4.5, Design

البته قبلش بگم که عنوان بهتری به ذهنم نرسید.
بسیاری از مواقع پیش می‌آید که در سایت خود بخواهیم کادری داشته باشیم که با کلیک بروی آن ظاهر و با کلیک دوباره بروی آن محو شود. مانند تصویر زیر

سپس با کلیک بروی قسمت مشخص شده از تصویر بالا تصویر مانند زیر ظاهر شود.



در این نوشته قصد داریم کادری به این صورت حالا به هر منظوری طراحی نماییم.

برای کار سه قسمت کد داریم:

کدهای طراحی قسمت مورد نظر در صفحه وب
نوشتن کدهای CSS مربوطه
نوشتن کدهای jQuery

در مرحله اول ابتدا صفحه وب خود را به نحو زیر ایجاد می‌نماییم.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>کادر لغزان با jQuery</title>
    <script src="Scripts/jquery-1.7.1.min.js"></script>
    <link href="CSS/site.css" rel="stylesheet" />
</head>
<body>
    <div id="loginPanel">
        <div style="height: auto;" id="login">
            <div>
                <div>
                    <br />
                    محتویات دلخواه خود را در این قسمت قرار دهید
                </div>
            </div>
            <div><a href="#" id="closeLogin"></a></div>
```

```
        </div>
        <div id="container">
            <div id="top">
                <!-- login -->
                <ul>
                    <li> </li>
                    <li><a id="toggleLogin" href="#">پانل باز شو</a></li>
                </ul>
                <!-- / login -->
            </div>
            <!-- / top -->

        </div>
    </div>
    <div id="main">
        محتویات سایت در این قسمت قرار می‌گیرد
    </div>
    </body>
</html>
```

در صفحه ایجاد شده قسمتی را برای نگهداری پانل مورد نظر قرار دادیم و در div با نام loginContent مواردی را که می‌خواهیم در پانل مربوطه نمایش داده شود، قرار می‌دهیم. نگهدارنده کل قسمت مربوطه (کادر لغزان می‌باشد)، و قسمت قسمتی است که دکمه یا عنوان مورد نظر برای باز شدن یا بستن کادر استفاده می‌شود.
در مرحله دوم کدهای CSS بخش‌های مورد نظر (جهت رنگ و تصاویر و شکل و شمایل کادر مورد نظر) را مانند زیر ایجاد می‌کنیم.

```
body {
     margin:0;
     padding:0;
     width:100%;
     background: #e9e9e9 url(images/header_bg.gif) top repeat-x;
     direction: rtl;
}
html {
     padding:0;
     margin:0;
}

#main {margin-top: 100px;}

#loginPanel {
    margin: 0px;
    position: absolute;
    overflow: hidden;
    height: auto;
    z-index: 3000;
    width: 100%;
    top: 0px;
    color: #fff;
}
#top {
    background: url(images/login_top.jpg) repeat-x 0 0;
    height: 38px;
    position: relative;
}
#top ul.login {
    display: block;
    position: relative;
    float: right;
    clear: right;
    height: 38px;
    width: auto;
    margin: 0;
    right: 150px;
    color: white;
    text-align: center;
    background: url(images/login_r.png) no-repeat right 0;
    padding-right: 45px;
}
#top ul.login li.left {
    background: url(images/login_l.png) no-repeat left 0;
    height: 38px;
    width: 45px;
    padding: 0;
    margin: 0;
    display: block;
    float: left;
```

```
}
#top ul.login li {
    text-align: left;
    padding: 0 6px;
    display: block;
    float: left;
    height: 38px;
    background: url(images/login_m.jpg) repeat-x 0 0;
}
#top ul.login li a {
    color: #fff;
    text-decoration: none;
}
#top ul.login li a:hover {
    color: #ff0000;
    text-decoration: none;
}
#login {
    width: 100%;
    color: white;
    background: #1E1E1E;
    overflow: hidden;
    position: relative;
    z-index: 3;
    height: 0px;
}
#login a {
    text-decoration: none;
    color: #fff;
}
#login a:hover {
    color: white;
    text-decoration: none;
}
#login .loginContent {
    width: 900px;
    height: 80px;
    margin: 0 auto;
    padding-top: 25px;
    text-align: right;
}
#login .loginClose {
    display: block;
    position: absolute;
    right: 15px;
    top: 10px;
    width: 70px;
    text-align: left;
}
#login .loginClose a {
    display: block;
    width: 100%;
    height: 20px;
    background: url(images/button_close.jpg) no-repeat right 0;
    padding-right: 10px;
    border: none;
    color: white;
}
#login .loginClose a:hover {
    background: url(images/button_close.jpg) no-repeat right -20px;
}
.cen { text-align: center;}
.w_100p{ width: 100%;}
```

خوب تا اینجای کار فقط کادر با قالب مورد نظر ایجاد شد، برای اینکه عمل مورد نظر انجام شود با استفاده از تکنیک‌های jQuery به صورت زیر کار را به پایان می‌رسانیم. در انتهای صفحه اسکریپت زیر را قبل از قسمت </body> می‌نویسیم.

```
<script type="text/javascript">
        $(document).ready(function () {
            $("#login").hide(0);
            $("#toggleLogin").click(function () {
                $("#login").slideToggle("slow");
            });
            $("#closeLogin").click(function () {
                $("#login").slideUp("slow");
            });
        });
```

```
    </script>
```

با نوشتن این اسکریپت بعد از لود صفحه مورد نظر ابتدا کادر ما مخفی می‌شود، سپس برای دکمه (یا هر المانی که می‌خواهیم با کلیک روی آن کادر بسته یا باز شود) کد کلیک می‌نویسیم که با کلیک بروی آن عمل اسلاید (باز یا بسته شدن) رخ دهد. در نهایت در رویداد کلیک لینک close تعیین می‌کنیم که کادر به آرامی بسته شود.

مثال کامل از ^ قابل دانلود است

لینک‌های کمکی جهت آشنایی بیشتر با توابع استفاده شده: slideUp

slideToggle

نظرات خوانندگان

نویسنده: مرتضی مختاری
تاریخ: ۱۳۹۱/۱۲/۲۱ ۲۰:۲۸

سلام آقای فتح الله ممنون از پستی که قرار دادید بنده کد شما رو تویه سایتم قرار دادم ولی یک مشکلی که وجود داره وقتی رویه دکمه ارسال که درون یک آپدیت پنل هستش کلیک میکنم قسمت toggle بسته میشه چطوری میشه این رو موقع post back باز نگه داشت .با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۲۱ ۲۳:۵۵

مراجعه کنید به مطلب نمایش پیغامی به کاربر در هنگام استفاده از MS Ajax و همچنین استفاده‌ی همزمان از آپدیت پنل ASP.Net و پلاگین‌های جی‌کوئری

عنوان: **نشانه های طراحی ضعیف**
نویسنده: توسعه چابک
تاریخ: ۱۳۹۲/۰۹/۱۱ ۱۲:۲۵
آدرس: www.dotnettips.info
برچسب‌ها: Design, OOP

برای آنکه طراحی قوی و درست را یاد بگیریم، لازم است که نشانه-های طراحی ضعیف را بدانیم. این نشانه‌ها عبارتند از:

۱- Rigidity (انعطاف ناپذیری): یک ماژول انعطاف ناپذیر است، اگر یک تغییر در آن، منجر به تغییرات در سایر ماژولها گردد. هر چه میزان تغییرات آبشاری بیشتر باشد، نرم افزار خشک‌تر و غیر منعطف‌تر است.

۲- Fragility (شکنندگی): وقتی که تغییر در قسمتی از نرم افزار باعث به بروز اشکال در بخش-های دیگر شود.

۳- Immobility (تحرک ناپذیری): وقتی نتوان قسمت هایی از نرم افزار را در جاهای دیگر استفاده نمود و یا به کار گیری آن هزینه و ریسک بالایی داشته باشد.

۴- Viscosity (لزجی): وقتی حفظ طراحی اصولی پروژه مشکل باشد، می-گوییم پروژه لزج شده است. به عنوان مثال وقتی تغییری در پروژه به دو صورت اصولی و غیر اصولی قابل انجام باشد و روش غیر اصولی راحت‌تر باشد، می‌گوییم لزج شده است. البته لزجی محیط هم وجود دارد مثلا انجام کار به صورت اصولی نیاز به Build کل پروژه دارد که زیاد طول می-کشد.

۵- Needless Complexity (پیچیدگی اضافی): زمانی که امکانات بدون استفاده در نرم افزار قرار گیرند.

۶- Needless Repetition (تکرارهای اضافی): وقتی که کدهایی با منطق یکسان در جاهای مختلف برنامه کپی می-شوند، این مشکلات رخ می‌دهند.

۷- Opacity (ابهام): وقتی که فهمیدن یک ماژول سخت شود، رخ می‌دهد و کد برنامه مبهم بوده و قابل فهم نباشد.

چرا نرم افزار تمایل به پوسیدگی دارد؟

در روش‌های غیر چابک یکی از دلایل اصلی پوسیدگی، عدم تطابق نرم افزار با تغییرات درخواستی است. لازم است که این تغییرات به سرعت انجام شوند و ممکن است که توسعه دهندگان از طراحی ابتدایی اطلاعی نداشته باشند. با این حال ممکن است تغییرایی قابل انجام باشد ولی برخی از آنها طراحی اصلی را نقض می‌کنند. ما نباید تغییرات نیازمندیها را مقصر بدانیم. باید طراحی ما قابلیت تطبیق با تغییرات را داشته باشد.

یک تیم چابک از تغییرات استقبال می‌کند. وقت بسیار کمی را روی طراحی اولیه کل کار می‌گذارد و سعی می-کند که طراحی سیستم را تا جایی که ممکن است ساده و تمیز نگه دارد با استفاده از تست‌های واحد و یکپارچه از آن محافظت کند. این طراحی را انعطاف پذیر می‌کند. تیم از قابلیت انعطاف پذیری برای بهبود همیشگی طراحی استفاده میکند. بنابراین در هر تکرار نرم افزاری خواهیم داشت که نیازمندی‌های آن تکرار را برآورده می‌کند.

عنوان: **تفاوت Desktop Application با Web Application**
نویسنده: یاسر مرادی
تاریخ: ۱۳۹۳/۰۴/۲۰ ۱۲:۴۵
آدرس: www.dotnettips.info
گروه‌ها: Architecture, Design

در هنگام گفتگو با افراد مختلفی که در پروژه‌های توسعه نرم افزار، نقش‌های مختلفی را دارا می‌باشند، یکی از جالب‌ترین و اساسی‌ترین بحث‌ها تفاوت بین Desktop App و Web App می‌باشد، و این که پروژه بر اساس کدام مدل باید نوشته شود.

در اینترنت و در منابع معتبر، تفسیرهای متفاوتی از این دو وجود دارد، که گاه دقیقا با نظر من یکی بوده و گاه تا 180 درجه بر عکس هستند، آنچه که در ادامه می‌خوانید می‌تواند لزوما نظر شما نباشد.
گروهی از افراد بر این باور هستند که اجرای برنامه در محیط مرورگر (ظاهر مرورگر و نه Sandbox آن)، یکی از ملاک‌های ما بین Desktop App و Web App است، گروهی دیگر نیز اجرا شدن برنامه بر روی بستر اینترنت و یا شبکه‌ی محلی را جزو ملاک‌ها می‌دانند، و گروهی دیگر نیز زبان برنامه نویسی برنامه را ملاک می‌دانند، برای مثال اگر با HTML/JS باشد Web App است، اگر نه Desktop App است.
اما آنچه که در عمل می‌تواند تفاوت بین یک Desktop App را با یک Web App مشخص کند، **رفتار و عملکرد خود آن برنامه** است، نه بستر اجرای آن و این که آن رفتار منتج شده از چه کدی و چه زبان برنامه نویسی ای است.
اگر کمی دقیق به مطلب نگاه کنیم، می‌بینیم این که یک برنامه در چارچوب ظاهری یک مرورگر (نه Sandbox آن) اجرا شود، اصلا مقوله ای اهمیت دار نیست، کما این که برای مثال Silverlight اجازه می‌دهد، برنامه هم در داخل مرورگر و هم در بیرون از آن اجرا شود، و این کار با یک کلیک امکانپذیر است، آیا با همین یک کلیک برنامه از Web App به Desktop App تبدیل می‌شود یا بالعکس ؟
آیا یک برنامه مبتنی بر دلفی که تا همین یک ساعت پیش بر روی شبکه محلی در حال اجرا بوده، با انتقال پیدا کردن آن بر روی شبکه‌ی اینترنت، تبدیل به یک Web App می‌شود؟
آیا اگر ما با HTML/JS یک برنامه Native برای ویندوز فون بنویسیم که تک کاربره آفلاین باشد و اصلا سروری هم نداشته باشد، آیا Web App نوشته ایم ؟ **اصلی‌ترین** تفاوت مابین Web App و Desktop App که به **تفاوت در عملکرد آنها و مزایا و معایب آنها** منجر می‌شود، این است که انجام کارهایی که اپراتور با آنها در سمت کلاینت و سیستم مشتری سر و کار دارد، در کجا صورت می‌پذیرد؟
برای مثال در نظر بگیرید که یک دیتاگرید داریم که دارای Paging است، و ما از Page اول به Page بعدی می‌رویم، در یک Desktop App تنها **اطلاعات** از سرور گرفته می‌شود، و ترسیم خطوط و ستون‌ها و ردیف‌ها و ظاهر نمایشی دیتاگرید بر عهده کلاینت است، برای مثال اگر ستون قیمت داشته باشیم، و بخواهیم برای ردیف هایی که قیمت آنها زیر 10000 ریال است، قیمت به شکل سبز رنگ نمایش داده شود و برای بقیه ردیف‌ها به رنگ قرمز باشد، پردازش این مسئله و این if به عهده کلاینت است، اما در یک Web App، علاوه بر اطلاعات، تعداد زیادی tagهای مختلف، مانند table - tr - td و ... نیز به همراه اطلاعات آورده می‌شوند، که وظیفه نمایش ظاهری اطلاعات را بر عهده دارند، و آن if مثال ما یعنی رنگ سبز و قرمز در سمت سرور مدیریت شده است، و کلاینت در اینجا نمایش دهنده‌ی آن چیزی است که به صورت آماده از سرور آورده شده است.
در برنامه‌های Desktop آنچه که در سمت سرور وجود دارد، برای مثال یک WCF Service یا ASP.NET Web API است که فقط به رد و بدل کردن اطلاعات می‌پردازد، اما در Web Appها در سمت سرور ASP.NET MVC، ASP.NET Web Forms و PHP وجود دارند که علاوه بر اطلاعات برای کلاینت شما ظاهر صفحات را نیز آماده می‌کنند، و ظاهر اصلی صفحات از سمت سرور به سیستم مشتری ارسال می‌شوند، اگر چه که ممکن است در سمت کلاینت تغییراتی را داشته باشند.
به هر میزان رفتار برنامه ما شبیه به حالت اول باشد، برنامه ما Desktop App بوده و به هر میزان برنامه ما به حالت دوم نزدیک‌تر باشد، برنامه ما Web App است.
مزیت اصلی Web Appها در عدم انداختن بار زیاد بر روی دوش کلاینت‌های بعضا نحیف بوده، و عملا کلاینت به علت این که کار خاصی را انجام نمی‌دهد، پیش نیاز نرم افزاری و یا سخت افزاری خاصی احتیاج ندارد، و این مورد Web Appها را به یک گزینه ایده آل برای وب سایت هایی تبدیل کرده است که با عموم مردم در ارتباطند، زیرا که امکان ارائه آسان برنامه وجود دارد و تقریبا همه می‌توانند از آن استفاده کنند.
با توجه به شناخت عموم از برنامه‌های Web App به توضیح بیشتر برنامه‌های Desktop App می‌پردازم.
مزیت اصلی Desktop Appها در سرعت عمل بالاتر(به علت این که فقط دیتا را رد و بدل می‌کند)، توانایی بیشتر در استفاده از منابع سیستمی مانند سرویس نوشتن، و امکانات محلی مانند ارائه Notification و ... است، و در کنار آن برای مثال یک Desktop App

می‌تواند به نحوی طراحی شود که به صورت Offline نیز کار کند.
این مزیت‌ها باعث می‌شود که Desktop Appها گزینه ای مناسب برای برنامه‌های سازمانی باشند.
ضعفی که از گذشته در Desktop Appها وجود داشته است، که البته به معماری Desktop App بر نمی‌گردد، بلکه متاثر از امکانات است، عدم Cross Platform بودن آنها بوده است، تا آنجا که Desktop App در نظر خیلی از افراد همان نوشتن برنامه برای سیستم عامل ویندوز است.
با توجه به رویکرد جدی ای که در طول دو سال اخیر برای نوشتن برنامه Desktop App به شکل Cross Platform رخ داده است، خوشبختانه این مشکل حل شده است و اکنون لااقل دو راهکار جدی برای نوشتن یک برنامه Cross Platform با ویژگی‌های Desktop وجود دارد، که یکی از آنها راه حل‌های مبتنی بر HTML/JS است و دیگری راه حل‌های مبتنی بر C#/XAML
در راه حل‌های مبتنی بر HTML/JS در صورتی که شما برنامه را به شکل Web App طراحی نکرده باشید، و برای مثال در آن از ASP.NET Web Forms و ASP.NET MVC، PHP و ... استفاده نکرده باشید، می‌توانید یک خروجی کاملا Native با تمامی ویژگی‌های Desktop App برای انواع پلتفرم‌ها بگیرید.
استفاده از فریم ورک هایی که با طراحی Desktop App سازگار هستند، مانند Angular JS، Kendo UI و Jay-data، Ext JS و ... و استفاده از مدل طراحی Single Page Application می‌تواند سیستم کدنویسی ای ساده را فراهم آورد، که در آن شما با یک بار نوشتن برنامه می‌توانید خروجی اکثر پلتفرم‌های مطرح را داشته باشید، اعم از ویندوز فون، اندروید، iOS و ویندوز
امروزه حتی مرورگرها با فراهم آوردن امکاناتی مانند Client side databases و Manifest based deployment اجازه نوشتن برنامه Desktop با HTML/JS را که حتی می‌تواند Offline کار کند را به شما ارائه می‌کنند.
در کنار این راهکار، استفاده از C#/XAML برای نوشتن برنامه برای اکثر پلتفرم‌های مطرح بازار اعم از اندروید، iOS و Windows Phone و ویندوز، نیز به عنوان راهکاری دیگر قابلیت استفاده را دارا است.
حرکت پر شتاب و پر انرژی جهانی برای توسعه Cross Platform Desktop Development، خوشبختانه توانسته است تا حد زیادی امتیاز نوشتن برنامه‌های Desktop را در سیستم‌های Enterprise بالا ببرد.

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۲۰ ۱۳:۵

مطلبی هم من چند سال قبل در این مورد نوشته بودم
« چرا در سازمان‌ها برنامه‌های وب جایگزین برنامه‌های دسکتاپ شده‌اند (یا می‌شوند)؟ »

نویسنده: علیرضا
تاریخ: ۱۳۹۳/۰۴/۲۱ ۱۰:۱۱

به نظر من این بحث به همین سادگی نیست و انتخاب پلتفرم اجرای پروژه به پارامترها و ویژگی‌های زیادی مرتبط هست. بطور مثال سرعت توسعه برنامه‌های ویندوز حداقل در قسمت طراحی رابط کاربری سریعتر و ساده‌تر از وب هست. و یا در مثال دیگر رفتار غیر یکسان مرورگرها مشکلاتی را در طراحی نرم افزارهای بزرگ ایجاد می‌کنه و مشکل ساز میشه من بعد از سال‌ها طراحی سیستم‌های سازمانی روش استفاده ترکیبی از پلتفرم‌های مختلف را انتخاب کردم بطور مثال قسمت مدیریت یک سیستم را بصورت ویندوزی و قسمت رابط ماربری را با وب و... طراحی کردم. متاسفانه طراحی اولیه زبان HTML با هدف نمایش اطلاعات بوده و بهبودهای اخیر از جمله وب 2 پاسخی منطقی به نیاز به توسعه نرم افزارهای Cross Platform بوده ولی هنوز هم با پیچیدگی‌های زیادی روبروست. به نظر قابلیت‌های نرم افزار تحت وب بیش از واقعیت بزرگ نمایی شده و هنوز هم در برخی راه کارها استفاده از نرم افزارهای تحت ویندوز گزینه مناسب‌تری خواهد بود اما این به معنی چشم پوشی بر مزایای منحصر به فرد وب نخواهد بود و هنوز انتخاب پلتفرم بستگی زیادی به نیازمندی‌ها و امکانات پروژه خواهد داشت.

نویسنده: رحیم
تاریخ: ۱۳۹۳/۰۴/۲۱ ۲۲:۴۴

سلام لطفا در مورد این جمله بیشتر توضیح دهید
در کنار این راهکار، استفاده از C#/XAML برای نوشتن برنامه برای اکثر پلتفرم‌های مطرح بازار اعم از اندروید، iOS و Windows Phone و ویندوز، نیز به عنوان راهکاری دیگر قابلیت استفاده را دارا است.
آیا منظور شما استفاده از نرم افزارهای شرکت ثالث نظیر xamarin و .... می‌باشد یا تکنولوژی جدیدی از سمت مایکروسافت ارائه شده که این امکان رو میسر می‌کنه

نویسنده: یاسر مرادی
تاریخ: ۱۳۹۳/۰۴/۲۲ ۸:۴۳

بله، منظور روش‌های ارائه شده مبتنی بر پلتفرم Xamarin است، البته در نظر بگیرید این کار بدون کمک‌های فنی ارائه شده توسط مایکروسافت و همچنین رفع مشکل لایسنس Portable Class Libraryها و ... از سوی مایکروسافت امکانپذیر نبود.
مایکروسافت بنظر قصد پشتیبانی فنی و مالی و در نهایت خرید Xamarin رو داره، و بنظر نمی‌آد که بخواد این مسیر رو از نو پیش بره، چون واقعا کار زیادی می‌بره