افزایش قابلیت نگهداری برنامه رسم نمودار فروش با بکارگیری از MVP Pattern و Fluent NHibernate ORM

نیاز روز افزون به کامپیوتر و مکانیزه کردن و سپردن تقریبی تمامی امور به دست ماشین امری است که انکار ناشدنی است . در این بین تولید کنندگان نرم افزار نیز تلاش میکنند تا نرم افزاری تولید کنند تا بتواند اکثر نیاز های متقاضیان را به بهترین نحو ممکن تامین کند ودر همین راستا در تلاش هستند که روند تولید نرم افزار را به سمتی بکشانند که ساختار استاندارد و تائید شده‌ای داشته باشد.

نامبردن تمامی خصوصیات یک نرم افزار خوب در این مقال نمی گنجد اما تعداد محدود و مهمی از آنها عبارتند از:

- صحت (Correctness): چقدر نرم افزار نیازهای مشتری آن را به شکل درست برآورد می کند.
- قابلیت اطمینان و توانمندی (robustness & reliability): به جز صحت، چقدر نرم افزار در شرایط غیر طبیعی و ورودی های اشتباه مقاوم است و واکنش درست نشان می دهد.
- کارایی (efficiency): چقدر از منابع سخت افزاری و نرم افزاری درست و به شکل بهینه استفاده می شود.
- قابلیت جابجایی (Portability): چقدر می توان نرم افزار را از یک Platform به دیگری منتقل کرد، مثلا روی سیستم عامل های مختلف کار می کند یا نه...
- قابلیت استفاده مجدد (reusability): اینکه از یک قسمت از نرم افزار یا همه آن بتوان در همان نرم افزار و یا در سیستمهای دیگر مجدد استفاده کرد.
- قابلیت استفاده (usability) یا استفاده آسان (easy to use): چقدر نرم افزار برای گروه کاربران هدفش آسان و قابل استفاده است و راهنما و یا واسط کاربری مناسب دارد.
- قابلیت نگهداری (maintainability): با چه هزینه ای می توان نرم افزار را نگهداری و پشتیبانی کرد.

- قابلیت وارسی (verifiability): چقدر آسان می توان نرم افزار را تست کرد و از صحت و قابلیت اطمینان آن مطمئن شد.
- سازگاری (compatibility) یا (Interoperability): آیا نرم افزار در کار سایر نرم افزار ها اختلال ایجاد می کند و آیا می تواند با آنها کار کند مثلا داده به شکل فرمتهای استاندارد به عنوان وروردی یا خروجی بگیرد.
- و....

اگر برنامه شما، طوری نوشته شود که کدهای مربوط به دسترسی داده، مثلا دسترسی به پایگاه‌داده که محل ذخیره داده است، و رابط کاربری، در یک یکدیگر ترکیب شده باشند و در قسمتهای جداگانه نوشته نشده باشند، مشکلاتی زیادی پیش خواهد آمد. برای مثال، در صورتی که بعدا شما بخواهید این برنامه را توسعه دهید، در صورت تغییر دادن یک قسمت، باید بقیه قسمتها را نیز تغییر بدهید. همچنین در صورتی که این برنامه دچار مشکل خطایی باشد، با توجه به عدم جداسازی و در هم آمیختگی قسمتهای مختلف، رفع خطای برنامه، دشوار و پیچیده خواهد بود. همچنین امکان تغییر و بهینه سازی این برنامه، دشوار و یا حتی غیر ممکن خواهد بود. فرض کنید شما بخواهید پنجره‌ها و فرم‌های جدیدی را اضافه کنید، و کدهای قسمتهای مختلف در هم ترکیب شده‌اند، تصور کنید که بهینه سازی و اضافه کردن قسمتهای جدید به این برنامه چقدر دشوار است.

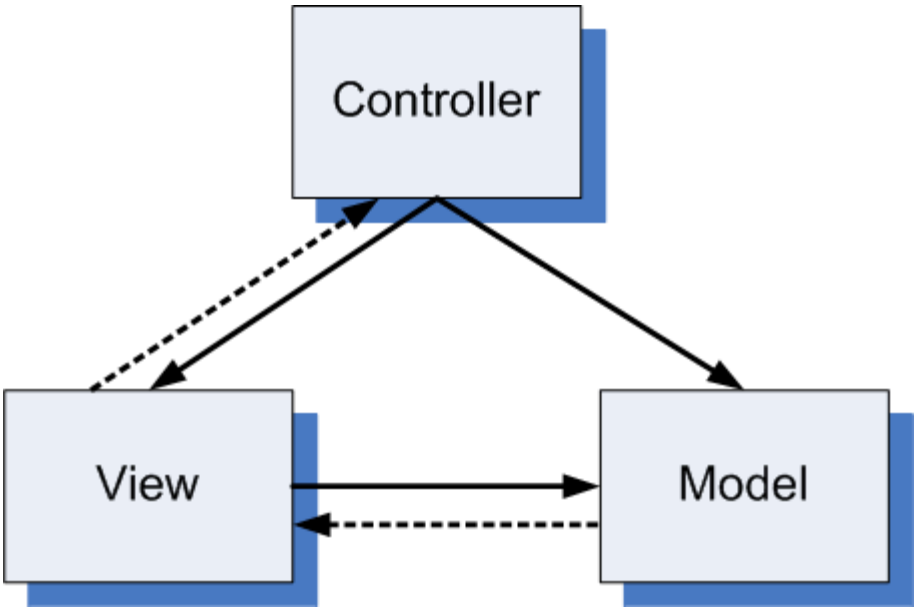
برای افزایش کیفیت نرم افزار و به خصوص قابلیت نگهداری نرم‌افزار از راهکارهای زیادی استفاده می‌شود. یکی از راه کار هایی که امروزه بیشتر شاهد استفاده ان هستیم، تولید نرم‌افزار بر اساس ساختارهای لایه‌ای می باشد، بدین صورت که کل نرم‌افزار به تعداد لایه‌هایی تقسیم می شود. هر لایه وظیفه خاص خود را دارد و لایه‌ها از نتایج لایه‌های دیگر استفاده می‌کنند.

تعداد این لایه‌ها بسته به نرم‌افزار و طراحی می‌تواند ۲، ۳، ۴ یا ۵ لایه یا حتی بیشتر باشد. اما استاندارد ان که بیشتر از بقیه استفاده می‌شود ۳ لایه است و به روشی که بر اساس این تئوری پیاده‌سازی می‌شود اصطلاحا "three-tier Architecture"گفته می‌شود.

در تئوری ۳ لایه، لایه‌ها عبارتند از:

- Data Access layer
- Business Logic Layer
- Presentation Layer

معماری ۳ لایه یک معماری انتزاعی است و با وجود این‌که وجود این ۳ لایه را بیان می‌کند، درباره کلاس-های موجود در این لایه‌ها و ارتباط آن‌ها با هم مطلب دقیقی بیان نمی‌کند. دو مدل دقیق‌تر برای پیاده‌سازی آن ارایه شده‌است: Model-View-Controller (MVC) و Model-View-Presenter (MVP)



- **MODEL**

مدل در اصل حاوی اطلاعات مرکزی برای برنامه کاربردی است که شامل دسترسی به داده‌ها ، داده‌ها (اطلاعات پایگاه داده)، قوانین اعتبار سنجی (validation rules) و منطق مجتمع سازی است.

- **VIEW**

این بخش که در واقع همان بخش Presentation Layer در معماری ۳ لایه میباشد وظیفه بر قراری ارتباط با کاربر نهایی و گرفتن داده از کاربر و نمایش داده های اماده با کاربراز طریق برقراری ارتباط با دو

بخش دیگر یعنی Model و controller است. در واقع نکته مهمی که در بخش View باید ان را مد نظر داشت این است که این لایه مسئول کنترل صحت داده‌های وارد شده از طریق کاربر و همچنین مسئول صحت داده‌های نشان داده شده به کاربر نیست. در واقع این بخش با داده‌های خام کار میکند.

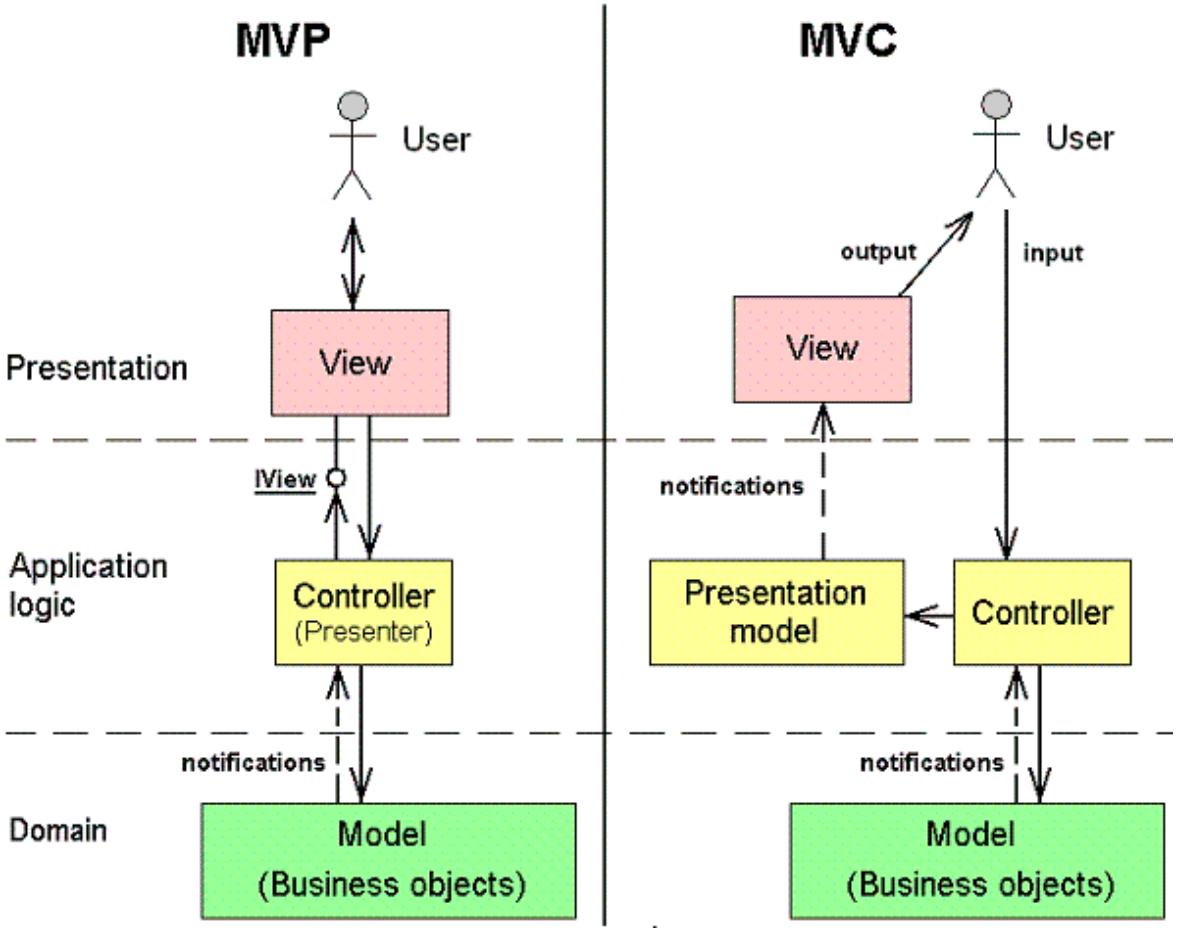
- **Controller**

کنترلگر کنترل‌کننده جریان منطقی برنامه کاربردی است، این بخش با تعامل با Model و View می تواند اجرا و جریان اطلاعاتی برنامه کاربردی را کنترل کند. می توان گفت که کنترل‌گر واسط بین Model و View می باشد؛ یعنی با Model کار می کند و در انتها نمایشگری را برای نشان دادن واسط کاربری انتخاب می کند. ورودی کاربر را مدیریت کرده و به آنها پاسخ می دهد و با کاربر تعامل می کند. برای مثال، Controller عبارت های پرس و جوی بانک اطلاعاتی را مدیریت کرده و آن ها را به Model ارسال می‌کند، وظیفه اجرای پرس و جوها با Model است.

MVP مدل بهبود یافته MVC است با دو تفاوت عمده:

۱. در مدل MVC کنترل‌کننده‌ها ورودی‌های کاربر را دریافت و پردازش می‌کنند ولی در مدل MVP، Viewها ورودی‌های کاربر را دریافت می‌کنند و پردازش آن‌ها را به Controller مربوط به خود واگذار می‌کنند. به همین دلیل مدل MVP با واسط‌های کاربری مدرن که خودشان اعمال کاربر را دریافت می-کنند بیشتر سازگار است و مورد استقبال بیشتری قرار گرفته است.
۲. در مدل MVC کنترل‌کننده‌ها با تغییر مدل ارایه‌دهنده میانی[1] View خود را تغییر می‌دهند. (با به-کارگیری از الگوی Observer) بدین صورت Viewها یک Observer خالص بدون دسترسی مستقیم به آن‌ها می‌شوند ولی در مدل MVP این مساله با یک ارتباط مستقیم از کنترل‌کننده به View حل می-شود. این باعث راحتر شدن MVP نسبت به MVC می‌شود.

[1] Intermediate Presentation Model



در واقع تبدیل برنامه کاربردی به لایه های مختلف (View-Controller-Model) باعث سرعت و انعطاف بیشتر در توسعه، تست و نگهداری برنامه کاربردی می شود، به طور مثال شما می توانید ظاهر برنامه (View) را تغییر دهید بدون اینکه کمترین تغییری در منطق تجاری (Model) برنامه ایجاد کنید. شما همچنین می‌توانید وظایف و نقش‌های کاری را یک پروژه به افراد مختلف واگذار کنید، بدون اینکه نگران ارتباط آنها باشید به عنوان مثال طراحان می توانند بروی لایه View کار کنند در حالی که برنامه نویسان در حال کار بروی لایه Model هستند.

برای استفاده از الگوی MVP باید وقت و هزینه بیشتری صرف کنیم و شاید نتوانیم به همین راحتی از آن استفاده کنیم، که برای استفاده از آن Frameworkهای متنوعی ارایه شده است که از میان نرم‌افزارهای کدباز[۲]

[2] Open Source

کاراترین آن برای .Net، MVC# است که قابلیت‌های فراوانی و یادگیری ساده‌ای دارد که پیاده‌سازی برنامه‌ها را سرعت می‌بخشد.

تهیه قسمت و یا لایه دسترسی به داده‌ها[3] در یک برنامه عموما تا ۳۰ درصد زمان کل تهیه یک محصول را تشکیل می‌دهد، اما باید در نظر داشت که این پروسه‌ی تکراری هیچ کار خارق العاده‌ای نبوده و ارزش افزوده‌ی خاصی را به یک برنامه اضافه نمی‌کند. تقریبا تمام برنامه‌های تجاری نیاز به لایه دسترسی به داده‌ها را دارند. پس چرا ما باید به ازای هر پروژه، این کار تکراری و کسل کننده را بارها و بارها تکرار کنیم؟

هدف ORM[4]، کاستن این بار از روی شانه‌های یک برنامه نویس است. با کمک این کتابخانه، دیگر رویه ذخیره شده‌ای[5] را نخواهید نوشت. دیگر هیچگاه با ADO.Net سر و کار نخواهید داشت. به این صورت می‌توان عمده وقت خود را صرف قسمت‌های اصلی و طراحی برنامه کرد تا کد نویسی یک لایه تکراری. همچنین عده‌ای از بزرگان این‌گونه ابزارها اعتقاد دارند که برنامه نویس‌هایی که لایه دسترسی به داده‌ها را خود طراحی می‌کنند، مشغول کلاهبرداری از مشتری‌های خود هستند! (صرف زمان بیشتر برای تهیه یک محصول و همچنین وجود باگ[6]های احتمالی در لایه دسترسی به داده‌های طراحی شده توسط یک برنامه نویس نه چندان حرفه‌ای)

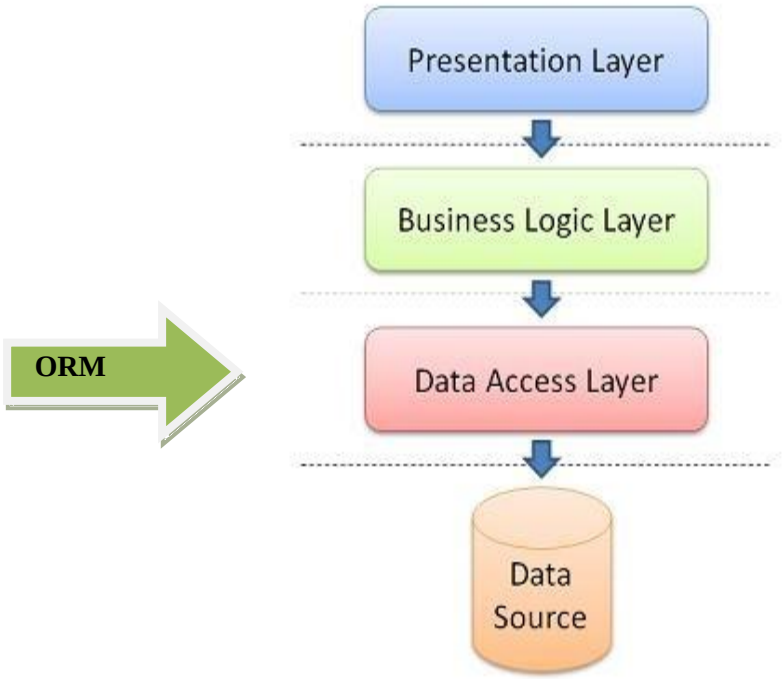
کار ابزارهای ORM خواندن ساختار دیتابیس شما بوده و سپس ایجاد کلاس‌هایی بر اساس این ساختار، برقراری ارتباط بین اشیاء ایجاد شده و جداول، دیدها[7]، رویه‌های ذخیره شده[8] و غیره می‌باشد. همچنین این ابزارها امکان تعریف روابط one-to-one، one-to-many، many-to-one و many-to-many بین اشیاء را نیز بر اساس ساختار پایگاه‌داده شما فراهم می‌کنند.

---

[3] Data Access Layer
[4] Object Relational Mapping
[5] Stored Procedure
[6] Bug
[7] View
[8] Stored Procedures

NHibernate یک ORM است که تحت پلت فرم Net. مایکروسافت کار می‌کند، و یک Frame Work را برای کار راحت‌تر و با پیچیدگی کمتر با پایگاه‌داده[9] پیشنهاد می‌کند. در این مدل فیلدهای پایگاه‌داده به Objectهای معادلی، برای استفاده در محیط Net. تبدیل (map) می‌شوند. با این کار برنامه‌نویسان از پیچیدگی‌های ارتباطی پایگاه‌داده‌ای خلاص می‌شوند.

NHibernate یک نرم‌افزار کدباز (Open Source) رایگان است و قسمتی از java ORM Hibernate است که کدی در قالب XML از موجودیت‌ها (Entity) و روابط (Relation) ارائه می‌کند. به صورت خودکار برای ذخیره و بازیابی اطلاعات کد Sql تولید می‌کند، و شما می‌توانید با داده‌های اصلی (meta data) ذخیره شده در source code کار کنید.



اخیرا کتابخانه‌ی دیگری برای انجام نگاشت (Mapping) تهیه شده به نام Fluent NHibernate که بسیار مورد توجه علاقمندان به Net framework. واقع گردیده است. با کمک کتابخانه‌ی Fluent NHibernate عملیات نگاشت اشیاء به جداول، بجای استفاده از فایل‌های XML ، توسط کدهای برنامه صورت خواهند گرفت. این مورد

[9] Database

مزایای بسیاری را همانند استفاده از یک زبان برنامه نویسی کامل برای تعریف نگاشت‌ها، بررسی خودکار نوع‌های داده‌ای و حتی امکان تعریف منطقی خاص برای قسمت نگاشت برنامه، به همراه خواهد داشت.

یکی از مهم‌ترین قسمت‌های این نرم‌افزار که بسیاری از پیش‌بینی‌های نرم‌افزارهای ERP به‌وسیله آن انجام می‌شود، قسمت نمودارهای فروش است. در بسیاری از نرم‌افزارهای موجود این نمودار واسط کاربر ضعیفی دارد و همچنین انتخاب‌های (Option) کمی برای رسم نمودار به شما عرضه می‌کند. یک سیستم ERP از اجزای سخت‌افزاری و نرم‌افزاری متعددی برای دست یابی به این مجموعه عظیم اطلاعات استفاده می‌کند. یک عامل کلیدی در بیشتر سیستم‌های ERP استفاده از پایگاه داده واحد به منظور ذخیره سازی داده برای واحدهای (Modules) مختلف سیستم است. در بسیاری از نرم‌افزارهای ارایه شده این پایگاه داده به درستی طراحی نشده و از کارایی پایینی برخوردار است. در این پروژه قصد داریم کارایی قسمت نمودار فروش را با بکارگیری از ORM Fluent NHibernate و طراحی مجدد پایگاه داده و واسط گرافیکی افزایش دهیم.

برای این منظور به اطلاعات کامل محصولات، سایت‌های فروش، انبارهای سایت ها، ویژگی های محصولات در سایت ها و اطلاعات فروش احتیاج داریم. البته محصولات و سایت‌ها باید به صورت دسته بندی (hierarchy) شده باشند به طور مثال: ابزارها- وسایل برقی- مته- شماره ۴

و یا برای سایت ایران- تهران- خیابان ولیعصر- مغازه ...

هر چند وقت یک بار از سایت ها از موجودی آن ها آمار گرفته می شود. هر محصول یک تاریخ انقضا دارد. درهر آمار گیری میزان فروش هم ذخیره می شود که از چه تاریخی تا چه تاریخی چقدر از چه محصولی فروش رفته است. سایت ها خصوصیاتی مثل آدرس و توضیحات دارند. همچنین هر محصول در هر سایت می تواند تعدادی ویژگی مثل قیمت، تاخیر سفارش، موجود یودن و غیره به انتخاب کاربر داشته باشد.

برای پیش بینی فروش همچنین به نمودارهایی از این اطلاعات احتاج داریم. این نمودار در یک صفحه که از دو درخت برای نشان دادن محصولات و سایت ها وانتخاب کردن آن ها، یک گزینه برای انتخاب کردن دوره زمانی، دو تاریخ برای تعیین بازه، و گزینه (Option)های متفاوت تشکیل شده است، رسم می شود.