
# برنامه‌نویس

اولین نشریه الکترونیکی جامعه برنامه نویسان

شماره اول - اسفندماه ۱۳۸۷

## آشنایی با زبان برنامه نویسی F#
جدیدترین زبان برنامه نویسی مایکروسافت یک زبان تابعی است ...

## مقدمه ای بر دنیای VoIP
راه اندازی مرکز تلفن داخلی، مزایا و معایب *VoIP*، ...

## استفاده از کنترلهای .NET در برنامه های MFC


## اینترفیس ها در دلفی
آشنایی با اینترفیس ها و کاربرد آن ها در دلفی

## خود آموز برنامه نویسی در ۱۰ سال
چرا همه برای یادگیری برنامه نویسی عجله دارند؟

## آموزش پایتون

## آشنایی با مفاهیم امنیت نرم افزار
*... debugger, disassembler, decompiler*

صاحب امتیاز
Www.BarnameNevis.Org

مدیر مسئول
**مهدی کرامتی**

سردبیر
**مهدی عسگری**

طراح و گرافیست
**صالح باقری**

صفحه آرا
**محمد جعفر کرمانپور**

همکاران این شماره
**بهروز راد – حمیدرضا متقیان**
**علی کشاورز – سعید موسوی فرید**
**محمد خرسندی – مهدی بیاضی**
**مهدی موسوی – نیما نیک فطرت**

با تشکر از
**الهام قدوسی، حسین جزایری، آقای مزدادوست**
**و تمام عزیزانی که**
**مارا در این شماره یاری کردند.**



فهرست مطالب

# سخن سردبیر

به نام حق

معمولا سخت ترین قدم، قدم اول است. این قانون در مورد نشر یک مجله نیز صدق می کند.

اثری که پیش رو دارید حاصل فعالیتی است که کاملا خودجوش و بدون هیچگونه حمایت مالی ، توسط برخی از اعضای سایت برنامه نویس (www.barnamenevis.org) شروع شد و اکنون شاهد شمارهٔ اول آن هستید. سایت برنامه نویس حدود سال ۱۳۸۲ فعالیت خود را به عنوان یک انجمن برای برنامه نویسان فارسی زبان شروع کرد و به تدریج با مدیریت و نظارت دقیق و مسئولانه و فعالیت های مستمر اعضای آن ، تبدیل به مرجع برنامه نویسان فارسی زبان در اینترنت شد.

با توجه به سختی های "شمارهٔ اول" و کم تجربگی و نبود حمایت مالی و افراد ثابت ، به مشکلات زیادی برخوردیم که با زحمت های واقعا شبانه روزی اعضای مجله موفق شدیم مجله را سر موعد مقرر (با یکی دو روز تاخیر!) منتشر کنیم.

در تولید این اثر تلاش زیادی شده که حتی الامکان بدون عیب باشد (از نظر فنی ، ویرایش ادبی و غیره) ولی به هر حال هر اثری که آفریدهٔ انسان باشد مطمئنا خالی از نقص نیست؛ با توجه به تلاش های دوستان تیم مجله در تولید اثری با کیفیت و این که تصمیم گیرندهٔ نهایی در مجله من بودم ، هرگونه ضعف و ایراد در محتوا و فرم و ... ناشی از قصور من ، و تمام نقاط قوت آن به خاطر زحمت دوستان است. هرگونه پیشنهاد ، انتقاد یا درخواست خود برای همکاری با مجله را به آدرس الکترونیکی ezine.barnamenevis@gmail.com ارسال کنید.

همانطور که یک نمایش بدون تماشاچی بی معنی است ، یک نشریه نیز برای خوانندگانش ایجاد شده ؛ پس تقدیم می کنم حاصل این اثر را به شما که هدف نهایی این تلاش ها بودید.

به امید این که مطالب این اثر برایتان مفید باشد.

مهدی عسگری

اسفند ۱۳۸۷

# خودآموز برنامه نویسی در ۱۰ سال

## (Teach Yourself Programming in Ten Years)

ترجمه: مهدی عسگری

نویسنده: Peter Norvig

خلاصه: در این مقاله Peter Norvig (نویسنده ی کتاب معروف هوش مصنوعی مورد استفاده در اکثر دانشگاه های جهان) در مورد پدیده و تب یادگیری سریع برنامه نویسی صحبت می کنه و نشون میده که برای تخصص در برنامه نویسی بر خلاف کتاب های عامه پسند (آموزش در ۲۱ روز ، ۲۴ ساعت ، ...) حداقل ۱۰ سال زمان و کار لازمه تا یک نفر به تخصص برسه (در هر رشته ای ، نه فقط برنامه نویسی)

چرا همه این قدر عجله دارند ؟

تو هر کتاب فروشی که بروید ، کلی کتاب می بینید که می خواهند در چند ساعت یا چند روز به شما کامپیوتر یا برنامه نویسی یاد بدهند (از ویندوز و اینترنت گرفته تا ویژوال بیسیک و جاوا و ...) من جستجوی زیر را در آمازون انجام دادم:

pubdate: a er 1992 and tle: days and
(title: learn or title: teach yourself)

یعنی کتاب هایی که از سال ۱۹۹۲ تا الان چاپ شده و در عنوانشان کلمات days و "یادگیری یا خودآموز" وجود دارد. نتیجه شامل ۲۴۸ مورد بود. (مترجم: اعداد و آمار منتشر شده در این مقاله مربوط به سال ۲۰۰۱ هستند. ) ۷۸ کتاب اول کتاب های مربوط به کامپیوتر بودند. عبارت days را با hours جایگزین کردم ؛ نتایج مشابه قبلی بود: ۲۵۳ مورد که ۷۷ تای اول در مورد کامپیوتر بودند. از ۲۰۰ کتاب صدر نتایج جستجو (در کل) ۹۶ درصد مربوط به کامپیوتر بود.

خلاصه این که یا مردم خیلی عجله دارند که در مورد کامپیوتر ها چیز یاد بگیرند یا هم که یادگیری کامپیوتر خیلی آسان تر از یادگیری چیزهای دیگر است. وگرنه هیچ کتابی در مورد بتهوون ، یا فیزیک کوانتوم یا حتی تربیت سگ ، در چند روز وجود ندارد.

خب ببینیم عنوانی مثل "آموزش پاسکال در ۳ روز" چه معنی ای دارد:

- یادگیری: در ۳ روز شما وقت کافی برای نوشتن چند برنامه ی مهم و یادگیری از موفقیت ها و اشتباهاتتان و همینطور وقت برای کار کردن با یک برنامه نویس با تجربه را نخواهید داشت. خلاصه ، وقت کافی برای یادگیری چندانی نخواهید داشت. در واقع این کتاب ها بیشتر در مورد یک آشنایی کلی بحث می کنند نه درک عمیق از موضوع. به قول آلکساندر پوپ: "یادگیری سطحی ، چیز خطرناکی است"
- پاسکال: در ۳ روز شاید بتوانید سینتکس پاسکال را یاد بگیرید (ان هم به شرطی که قبلا با یک زبان مشابه کار کرده باشید) اما زیاد نمی توانید درباره ی کاربرد زبان یاد بگیرید. یعنی اگر قبلا با بیسیک برنامه نویسی کرده باشید ، یاد می گیرید که برنامه ها را در پاسکال ولی با روش بیسیک بنویسید اما نقاط ضعف و قوت پاسکال را یاد نمی گیرید. به قول آلن پرلیس: "زبانی که طرز فکر شما را درباره ی برنامه نویسی تغییر ندهد ، ارزش یادگیری ندارد". ممکن است شما بخواهید قسمتی از پاسکال (یا هر زبان دیگری) را یاد بگیرید تا بتوانید از ابزار خاصی استفاده کرده و کار مشخصی را انجام دهید؛ در این صورت دیگر برنامه نویسی یاد نمی گیرید ، بلکه یاد می گیرید چطور ان کار مشخص را انجام دهید.
- ۳ روز: همانطور که در بخش بعدی می بینید ، این مدت کافی نیست.

آموزش برنامه نویسی در ۱۰ سال

محققان (۱) نشان دادند که برای متخصص شدن در بسیاری از زمینه ها (از شطرنج گرفته تا آهنگ سازی ، اپراتوری تلگراف ، نقاشی ، نواختن پیانو ، شنا ، تنیس عصب شناسی ، ...) حدود ۱۰ سال زمان لازم است. نکته ی مهم

انجام کار پیوسته و با توجه است ، نه صرفا کاری را مدام تکرار کردن ، بلکه به چالش کشیدن خودتان با کارهایی که فراتر از توانایی فعلی تان است و تحلیل کارایی تان قبل و بعد از انجام ان کار و اصلاح اشتباهاتتان. و بعد تکرار کنید. و دوباره تکرار کنید. هیچ میانبری وجود ندارد: حتی موزارت که در ۴ سالگی اعجوبه ای در موسیقی بود ، ۱۳ سال طول کشید تا بتواند اولین کار حرفه ای و جهانیش را بسازد. گروه موسیقی بیتلز در سال ۱۹۶۴ با وارد شدن به صحنه ی موسیقی اکثر کارهایشان جزو کارهای محبوب (و به قولی "نامبر وان") بودند ولی فراموش نکنید که اعضای گروه از سال ۱۹۵۷ در کلوب های کوچک لیورپول و هامبورگ فعالیت می کردند و اولین موفقیت بزرگشان ( Sgt. Peppers) در سال ۱۹۶۷ عرضه شد. در یک مطالعه بر روی دانش آموزان در آکادمی برلین ، محققان اعضای برتر ، متوسط و پایین کلاس را مقایسه کرده و از آن ها پرسیدند که چقدر تمرین کرده اند:

همه (از هر سه گروه) نواختن را از حدود ۵ سالگی شروع کرده و در سال های اول هر کس به یک میزان فعالیت می کرد (حدود ۲ یا ۳ ساعت در هفته) اما در حدود ۸ سالگی تفاوت های واقعی نمایان شد. کسانی که در کلاس هایشان بهترین بودند ، بیش از بقیه تمرین می کردند: ۶ ساعت در هفته در سن ۹ سالگی ، ۸ ساعت در ۱۲ سالگی ، ۱۶ ساعت در ۱۴ سالگی و الی آخر تا سن ۲۰ که هفته ای بیش از ۳۰ ساعت کار می کردند. در سن ۲۰ سالگی افراد برتر نزدیک به ۱۰۰۰۰ ساعت از عمرشان را به تمرین گذرانده بودند. دانش اموزان رده بعدی حدود ۸۰۰۰ ساعت و دانش اموزان معمولی (معلمین موسیقی در آینده!) حدود ۴۰۰۰ ساعت.

شاید هم ۱۰۰۰۰ ساعت ان عدد جادویی باشد نه ۱۰ سال. به نظر ساموئل جانسون (1709-1784) زمان بیشتری لازم است: "برتری در هر رشته ای فقط با یک عمر کار به دست می آید و با قیمت کمتری قابل خرید نیست" و اما دستورالعمل من برای موفقیت در برنامه نویسی:

- به برنامه نویسی علاقه داشته باشید. اطمینان حاصل کنید که انقدر از برنامه نویسی لذت می برید که حاضر باشید ۱۰ سال از عمرتان را صرفش کنید.
- با برنامه نویسان دیگر ارتباط داشته باشید. کد برنامه های دیگر را مطالعه کنید. این خیلی مهم تر از هر کتاب یا دوره ی آموزشی است.
- برنامه بنویسید. بهترین نوع آموزش ، انجام دادن است. در واقع حداکثر سطح کارایی برای افراد در هر رشته ای ، به طور خودکار با تجربه ی بیشتر حاصل نمی شود بلکه نیاز به تلاش تعمدی برای بهبود دارد.
- می توانید تحصیلاتتان را در دانشگاه هم ادامه بدهید. این کار باعث می شود از رشته تان درک عمیق تری به دست آورده و همچنین به کارهایی مشغول شوید که نیاز به مدرک دانشگاهی دارند. البته اگر از مدرسه و درس لذت نمی برید ، می توانید تجربه ی مشابهی را در کار کسب کنید (البته با تلاش بیشتر) Eric Raymond در دیکشنری هکر ها می گوید: "مطالعه و تحصیلات آکادمیک در رشته ی کامپیوتر یک شخص را تبدیل به یک برنامه نویس حرفه ای نمی کند ، همانطور که یک فرد فقط با مطالعه ی رنگ و قلم مو نمی تواند نقاش شود". یکی از بهترین افرادی که در عمرم استخدام کردم ( Jamie Zawinski) ، فقط تا دبیرستان درس را ادامه داده بود؛ با این وجود این شخص نرم افزار های خیلی خوبی تولید کرده (موزیلا و xemacs) و گروه خودش را در googlegroups دارد و حتی انقدر پول دراورد که بتواند یک کلوب شبانه بخرد.
- روی پروژه های تیمی کار کنید. در بعضی از پروژه ها بهترین و در بعضی دیگر بدترین برنامه نویس باشید. وقتی بهترین هستید توانایی های خودتان به عنوان یک رهبر گروه را تست کرده و به دیگر افراد الهام می دهید. وقتی هم که بدترین باشید یاد می گیرید که حرفه ای ها چه کار می کنند و همچنین چه کارهایی را دوست ندارند انجام دهند (چون این کارها را می سپارند به شما!)
- روی پروژه های دیگران کار کنید. برنامه هایی که دیگران نوشتند را مطالعه کنید. سعی کنید باگ های برنامه های دیگران را رفع کنید. به این فکر کنید که چطور برنامه هایتان را طراحی کنید که کار کسانی که می خواهند ان را نگهداری کنند آسان تر شود.
- کلی زبان برنامه نویسی یاد بگیرید. یک زبان که از شی گرایی و کلاس ها پشتیبانی کند (مثل جاوا یا سی پلاس

پلاس) ، یک زبان تابعی (مثل لیسپ یا ML) (م: یا Haskell) ، یک زبان اعلانی (مثل پرولوگ یا قالب ها در سی پلاس پلاس) ، یک زبان که از coroutine ها پشتیبانی کند (مثل Scheme یا Icon) (م: امروزه زبان های دیگری از جمله سی شارپ هم coroutine را دارند) و یک زبان که از برنامه نویسی موازی پشتیبانی کند (مثل Sisal) (م: و Erlang)

- یادتان باشد که ما با کامپیوتر سر و کار داریم. بدانید که چقدر طول می کشد تا کامپیوترتان یک دستورالعمل را اجرا کند، یک کلمه را از حافظه واکشی کند (در هر دو حالت وجود و نبودش در کش) ، کلماتی متوالی را از دیسک بخواند ، یا به یک مکان از دیسک برود (seek) (م: منظور نویسنده آشنایی با معماری کامپیوتر است. آقای Stroustrup خالق سی پلاس پلاس هم روی این نکته تاکید دارند)
- درگیر استاندارد سازی یک زبان شوید. حالا می تواند کمیته ی ANSI C++ باشد یا استاندارد شخصی تان برای کدنویسی (مثلا این که برای تورفتگی از ۲ فاصله استفاده کنید یا ۴ تا) در هر حال یاد می گیرید که بقیه چه چیزهایی را در یک زبان دوست دارند و همینطور علت این دوست داشتن را.

با در نظر گرفتن این موارد ، این سوال پیش می آید که صرفا با مطالعه ی کتاب چقدر می توانید یاد بگیرید. قبل از تولد اولین بچه ام ، تمام کتاب های "چگونه..." (How To...) را خواندم بودم و باز هم حس می کردم مبتدی ام. ۳۰ ماه بعد (پس از تولد دومین فرزندم) دیگر سراغ کتاب هایم نمی رفتم؛ به جایش به تجربه ی شخصی ام تکیه کردم که معلوم شد خیلی مفیدتر و مطمئن تر از هزاران صفحه کتاب هایی است که توسط متخصصین نوشته شده اند. آقای Fred Brooks در مقاله ی "No Silver Bullets" (۲) طرحی سه بخشی برای پیدا کردن بهترین طراحان نرم افزار مشخص کرد:

۱- به طور سیستماتیک و هر چه زودتر طراحان برتر را مشخص کنید
۲- یک مربی حرفه ای به آن ها اختصاص دهید تا مسئول و مراقب کار باشد
۳- فرصت هایی را فراهم بیاورید که این افراد از طریق ارتباط و برانگیختن همدیگر ، رشد کنند

آلن پرلیس می گوید: "هر کسی می تواند حجاری یاد بگیرد: میکل آنژ باید یاد می گرفت چطور این کار را نکند. در مورد برنامه نویسان بزرگ هم همینطور است"

باشه ، بروید و ان کتاب جاوا را بخرید؛ احتمالا به دردتان بخورد ولی زندگی تان یا تخصصتان در کل در طول ۲۴ ساعت ، روز یا حتی ماه تغییر چندانی نخواهد کرد.

(۱)

Bloom, Benjamin (ed.) Developing Talent in Young People, Ballan ne, 1985.
Hayes, John R., Complete Problem Solver Lawrence Erlbaum, 1989.
Bryan, W.L. & Harter, N. "Studies on the telegraphic language: The acquisition of a hierarchy of habits. Psychology Review, 1899, 8, 345-375
Chase, William G. & Simon, Herbert A. "Perception in Chess" Cognitive Psychology, 1973, 4, 55-81.

(۲)

h p://citeseer.nj.nec.com/context/7718/0

## ضمیمه: انتخاب زبان

چندین نفر از من پرسیدند که کدام زبان برنامه نویسی را باید اول یاد بگیرند. هیچ جواب قطعی ای وجود ندارد ، اما به نکات زیر توجه کنید:

- دوستان: وقتی کسی از من می پرسد از چه سیستم عاملی استفاده کنم ، جواب می دهم: "از هر چیزی که دوستانت استفاده می کنند" مزیتی که از یادگیری از دوستانتان به دست می آورید ، هرگونه تفاوت بین سیستم عامل ها یا زبان های برنامه نویسی را خنثی می کند. البته دوستان آینده تان را هم در نظر بگیرید: جامعه ای از برنامه نویسان که شما در آینده جزئی از ان ها خواهید بود. آیا زبانی که انتخاب کردید یک جامعه ی بزرگ و فعال و در حال رشد دارد یا این که روز به روز از استفاده کنندگانش کم می شود ؟ آیا کتاب ، وب سایت و فروم های اینترنتی برای رسیدن به جواب سوال هایتان وجود دارد ؟ آیا از اعضای این فروم ها خوشتان می اید ؟
- سادگی: زبان هایی مثل جاوا و سی پلاس پلاس برای پروژه های بزرگ و حرفه ای طراحی شده اند که توسط تیمی از برنامه نویسان حرفه ای استفاده می شوند که

کارایی زمان اجرا برایشان مهم است ؛ در نتیجه این زبان ها بخش های پیچیده ای دارند تا بتوانند نیاز های این افراد را براورده کنند. اما شما نیازی به این زبان ها ندارید ، چون می خواهید برنامه نویسی یاد بگیرید. شما نیاز به زبانی دارید که یادگیری اش آسان باشد و یک برنامه نویس تازه کار بتواند به راحتی با آن کار کرده و سینتکسش در یادش بماند.

- تفریح: کدام روش را برای یادگیری پیانو می پسندید: روش معمولی و تعاملی که به محض فشار دادن هر کلید نتی را می شنوید یا روش دسته ای که پس از اتمام یک آهنگ نت ها را می شنوید ؟ مسلما روش تعاملی یادگیری را آسان تر می کند. از زبانی استفاده کنید که حالت تعاملی (interactive mode) دارد.

با توجه به این سه مورد ، توصیه ی من برای اولین زبان برنامه نویسی ، پایتون یا Scheme است. البته اگر سنتان تک رقمی است توصیه می کنم از Alice یا Squeak شروع کنید. مهم ترین چیز این است که انتخاب کرده و شروع کنید.

***

## نظر مترجم:

مقاله ی آقای Norvig الان با وجود گذشت فقط ۸ سال از نگارش ، جزو مقالات کلاسیک محسوب می شود. بعضی از قسمت ها با توجه به گذر زمان مطمئنا قدیمی هستند (مثل زبانی با قابلیت coroutine) و زبانی امروزی مثل F# اکثر قسمت هایی را که ایشان روی آن ها تاکید دارند پشتیبانی می کند (شاید برای این بخش بهتر باشد به نصیحت Don Syme خالق F# گوش کنیم که می گوید حتما Haskell,Prolog, F# , Python و سی شارپ را یاد بگیرید) ولی نکته ای که هست و هیچ وقت غبار زمان به خودش نمی گیرد این است که به صرف خواندن چند کتاب و پاس کردن چند ترم درس و چند هزار خط کد نوشتن ، کسی حرفه ای نمی شود و اگر می خواهیم در دنیای به شدت پویای نرم افزار زنده بمانیم ، باید حاضر باشیم عمرمان را (درست) صرف شغلمان کنیم.

متن اصلی :

h p://www.norvig.com/21-days.html

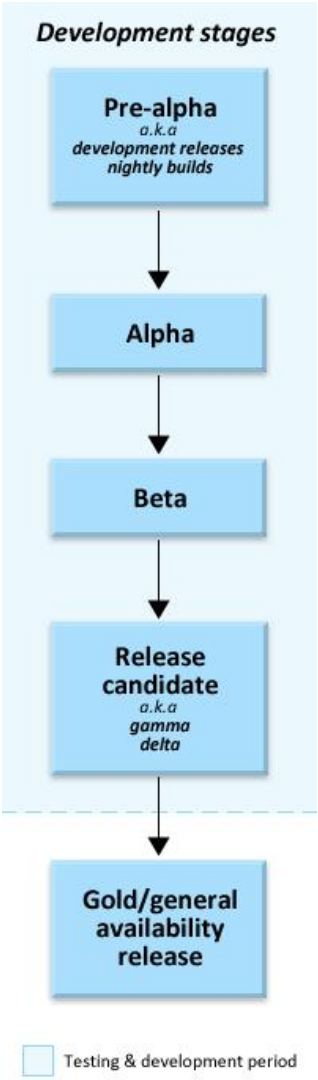
# Version

سطح : مبتدی

نویسنده : محمد خرسندی

اصطلاح "version" معمولاً در زمینه نرم افزار های کامپیوتری استفاده می شود که با هر تغییر در نرم افزار ، version محصول نرم افزاری نیز تغییر می کند، در حقیقت version ، وابسته به تغییرات و نوع تغییرات است. در نرم افزارهای تجاری اولین انتشار نرم افزار دارای ورژن 1.0 است. اعداد زیر 1 به معنی Alpha یا Beta بودن نرم افزار است، یعنی نسخه هایی که به منظور تست یا استفاده داخلی ایجاد شده یا نسخه هایی که به اندازه ی کافی برای استفاده عمومی و کاربردی پایدار (Stable) نیستند.

شکل زیر، شکل کلی استفاده از Version Information هست ، می بینید که build را موقعی میتوان به کار برد که قبل از آن از revision استفاده شده باشد.




Major.Minor[.Revision[.Build]]

در واقع در نسخه های بعدی نرم افزار، عدد Major، وقتی اضافه میشود که در قابلیتهای نرم افزار جهش و ارتقاء قابل ملاحظه و بزرگی ایجاد شده باشد.عدد Minor وقتی اضافه میشود که فقط ویژگی و خصوصیات کوچک اضافه شده باشد.

Revision : موقعی اضافه میشود که باگهای کوچک رفع شده باشد. این بخش ممکن ست شامل حروف نیز باشد، مانند :

Lotus 1-2-3 Release 1a

ممکن ست توسعه دهندگان بعضی مواقع از عدد 5.0 به عدد 5.5 بروند برای اینکه نشان دهند که خصوصیات قابل ملاحظه ای اضافه شده است، اما توجیه کافی برای بالا بردن عدد major وجود نداشته باشد، امیدوارم که متوجه منظورم شده باشید.

Build : عدد یست که تعداد دفعات ساخت نرم افزار را مشخص میکند . به این معنی که هر بار که شما نرم افزار را Build میکنید ، میبایست یکی به این عدد اضافه شود. (برای مثال این عدد توسط مایکروسافت خیلی استفاده میشود) . البته بعضی شرکتها تاریخ Build نرم افزار را قرار می دهند که این نیز امکان پذیر ست.

## Development Stages

در برنامه نویسی، اصطلاح development stage ، چگونگی جریان و پیشرفت پیاده سازی یک بخش از نرم افزار را شرح می دهد.

development stages شامل سه بخش اصلی ست :

۱- مرحله ای که ویژگیهای جدید به نرم افزار اضافه میشود ، که عموماً Alpha Stage نیز نامیده میشود.

۲- مرحله ای که فعالانه در حال رفع مشکل هستیم که Beta Stage نیز نامیده میشود.

۳- مرحله ای که تمام مشکلات مهم رفع شده اند که Stable Stage نامیده میشود.

شرح development stages :

Pre-Alpha : بعضی مواقع قبل از اینکه یک نسخه Alpha یا Beta منتشر شود، نسخه ای خودمانی به نام Pre-Alpha منتشر می شود که در مقایسه با نسخه Alpha یا Beta معمولاً خصوصیات و حتی چهره ای ناقص دارد. در این مرحله طراحان نرم افزار قصد دارند مشخص کنند که نرم افزار دقیقاً چه قابلیتهایی میبایست داشته باشد.

Alpha : در نسخه Alpha ی یک محصول، هنوز منتظر پیاده سازی کامل قابلیتهای آن هستیم. نسخه ای که در مرحله Alpha ساخته میشود، اولین نسخه ایست که به دست تسترهای این نرم افزار می رسد.

Beta : این نسخه معمولاً اولین نسخه از یک نرم افزار کامپیوتریست که پیاده سازی کامل آن طبق نیازهای اولیه شناخته شده نرم افزار صورت گرفته است. احتمالاً این نسخه هنوز هم ناپایدار است، اما برای نمایش (Demo) درون سازمانی/شرکتی و همچنین پیش نمایشی برای انتخاب مشتریان مفید ست، فقط در همین حد و نه برای انتشار.

Release Condidate : این اصطلاح اشاره به نسخه نهایی یا Final Product دارد که برای انتشار ، بدون بروز مشکلات خطرناک آماده ست. مایکروسافت اغلب از این واژه استفاده میکند، Apple هم از عبارت golden master استفاده می کند. از عبارت gamma هم برای نسخه ای که اساساً تکمیل شده ولی در حال تست هست استفاده میشود، بعضاً از عبارت delta نیز در این حالت استفاده میشود. از عبارت omega برای نسخه ای که تست نهایی هم شده استفاده میشود.

Gold Release : نسخه ای پایدار و بدون مشکل با کیفیت مناسب برای توزیع گسترده و استفاده ی کاربران نهایی.

***

# Smart & Gets Things Done

در هر شماره در این ستون کتاب های برتر برنامه نویسی را معرفی خواهم کرد. معیار من برای برتر بودن ، نقد های مثبت در اینترنت ، سفارش بزرگان و مهم تر از همه این است که خودم این کتاب (ها) را خوانده باشم. البته تلاشم بر این است که کتاب هایی را معرفی کنم که مختص یک پلتفرم یا زبان برنامه نویسی نبوده و توسط اکثر برنامه نویسان قابل استفاده باشند.

برای این شماره این کتاب را در نظر گرفتم:

Smart & Gets Things Done

این کتاب نوشته ی Joel Spolsky هست. احتمالا Joel را می شناسید. اگر بخواهیم ۳ وبلاگ برتر و پربیننده ی مربوط به برنامه نویسی را نام ببریم حتما یکی از آن ها وبلاگ Joel خواهد بود که در این آدرس واقع است:

www.joelonsoftware.com ؛ در ضمن در سایت ایشان یک فروم هم وجود دارد :

http://discuss.joelonsoftware.com/default.asp

Joel زمانی در مایکروسافت کار می کرد (در دهه ی ۹۰) که در این مدت در تیم Access بوده و وظیفه ی تعریف Specification مربوط به VBA را به عهده داشت. پس از مایکروسافت اندک مدتی هم در شرکت Juno کار کرده و بعد شرکت خودش را (FogBugz) تاسیس کرد که محصولاتی برای مدیریت پروژه (مثل Bug Tracking) توسعه می دهند.

از Joel کتاب های زیر هم چاپ شده:

User Interface Design for Programmers
Joel on Software
More Joel on Software

و کتاب Best Software Writing I که Joel گردآورنده ی مطالبش بود.

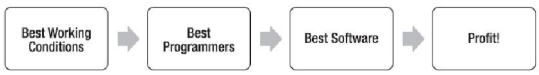
این کتاب Joel دربردارنده ی ۷ فصل و یک ضمیمه است. ایشان در این کتاب به موضوع مهم "چطور برنامه نویسان خوب را پیدا کرده و استخدامشان کنیم" می پردازد. Joel (همانطور که در روی جلد کتاب هم اشاره کرده) عقیده دارد برنامه نویسان برتر ده برابر برنامه نویسان معمولی کارا هستند. خب مسلما برنامه نویسان برتر حاضر نخواهند شد در هرجایی و با هر شرایطی کار کنند، و بسیاری از آن ها نیز قبل از این که ما پیدایشان کنیم توسط شرکت های بزرگ استخدام شده اند.

سر فصل ها:

۱- Hitting the High Notes
۲- Finding Great Developers
۳- A Field Guide to Developers
۴- Sorting Resumes
۵- The Phone Screen
۶- The Guerrilla Guide to Interviewing
۷- Fixing Suboptimal Teams

ضمیمه: The Joel Test

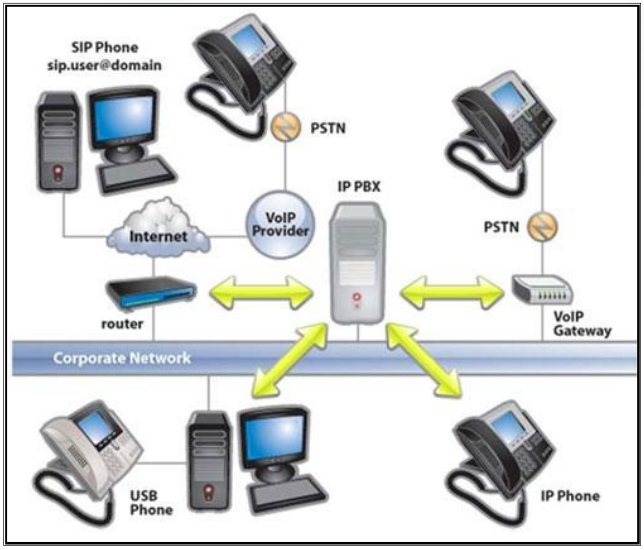
نسخهٔ pdf این کتاب ۱۷۰ صفحه است که با توجه به حاشیه ی زیاد و فونت بزرگ ، اندازه ی واقعی اش (چاپی) از این کمتر خواهد شد و خواندنش وقت زیادی نمی گیرد. خواندن این کتاب فقط به مدیران پروژه و مدیران شرکت های نرم افزاری توصیه نمی شود؛ حتی اگر یک برنامه نویس معمولی هم هستید این کتاب را بخوانید تا بفهمید فرق یک برنامه نویس عالی با یک برنامه نویس متوسط چیست و در شرکت های بزرگ برای استخدام روی چطور افرادی تاکید دارند. بیشترین تاکید کتاب بر روی دو ویژگی است: باهوش بودن و قابلیت انجام کار. یعنی صرفا افراد باهوش به درد نمی خورند. ما افرادی نیاز داریم که بتوانند کار انجام دهند. نکته ی دیگری هم که Joel در اکثر نوشته هایش به آن اشاره دارد ، فراهم اوردن محیط کاری خوب برای برنامه نویسان است. شکل زیر را از کتاب برداشتم:




این کتاب هم مثل بقیه ی نوشته های Joel نثر فوق العاده روان و جذابی دارد که با کمک داستان های واقعی از تجارب Joel ، مطالعه ای لذت بخش خواهد بود. کتاب در سال ۲۰۰۷ و توسط انتشارات APress چاپ شده است.

# مقدمه ای بر دنیای VOIP

نویسنده : سعید موسوی فرید




## ۱. پیشگفتار

هنوز مدت زیادی (کمتر از ۳۰ سال) از گفته الوین تافلر در مورد "تغییر ماهیت قدرت" در کتاب "موج سوم" مبنی بر اینکه در آینده ای نه چندان دور، اطلاعات بیانگر قدرت و ثروت افراد، جوامع و دولت ها خواهد بود نگذشته است؛ که ما به عینه شاهد این موضوع در کشور خودمان نیز هستیم. بسیاری از اعضاء هم دوره دانش آموخته در این رشته؛ تا چند سال پیش با توجه به انحصاری بودن سیاست گذاری‌های کلان اطلاعاتی کشور و همچنین رشد بی منطق، نادرست و قارچ گونه اینترنت و شرایط نامطلوب ستون فقرات انتقال داده[۱] در کشورمان، حتی فکر ورود تکنولوژی های رقیب اطلاعاتی و سر فرود آوردن قدرت های تجاری در برابر توسعه دهندگان و شرکت های متخصص غیردولتی را حداقل در دهه جاری به ذهن راه نمی دادند ولی با کمال شگفتی اکنون شاهد برتری گرفتن آن ها و ورود قدرتمند به دنیای صنعت و همچنین درخواست ناگزیر نهادها در به کارگیری این تکنولوژی ها می باشیم[۲] و این گفته کاملا بر ما تحقق یافته که در عصر کنونی هیچ پدیده ای چون "تغییر" ثابت نیست.

موضوعی که لازم دانستم در ابتدا جهت به وجود نیامدن ابهام و سوء برداشتی از مقاله‌ای که پیش رو دارید عرض کنم این است که تمام مطالب ذکر شده درصورتی که منبع و ماخذ آن ذکر نگردد (و یا فراموش نشود!) و از رده مطالب علمی خارج گردد، دیدگاه و نقطه نظرات شخصی بنده می‌باشد و ارتباطی به گروه یا شرکت خاصی ندارد و تذکر، انتقاد یا پیشنهادات در مورد آن را پذیرا می‌باشم.

در نگارش این مقاله (و به امید خدا مقاله های بعد) کوشش بر این بوده است که تنها بر جنبه های تئوریک و نظریه پردازی بسنده نشود و افزون بر جنبه تئوریک از دیدگاه عملی نیز به موضوع پرداخته شود. شایان ذکر است آگاه ساختن اینجانب از لغزشها و کاستی های احتمالی دوستان جامعه برنامه نویس و اساتید محترم و خوانندگان گرامی منتی است که موجب سرافرازی بنده خواهد شد.

## ۲. مقدمه

در سالیان اخیر افراد و شرکت های کوچک و بزرگ بسیاری به مبحث VoIP[۳] پرداخته اند به طوری که شما با یک جستجوی ساده در سایت گوگل میلیون ها مطلب در مورد آن خواهید یافت، بنابراین پرداختن به جزئیات تعریفی و معنایی علاوه براین که تکرار مکررات خواهد بود، در حوصله این مقاله نیز نمی گنجد؛

[۱] Internet Backbone : ر.ک. به: http://en.wikipedia.org/wiki/Internet_backbone
[۲] به عنوان نمونه ر.ک. به: http://www.ict.gov.ir/newsdetail-fa-3215.html

[۳] Voice over Internet Protocol: جهت آشنایی بیشتر با مبانی تئوری مراجعه نمایید به:
- ویکی انگلیسی : http://en.wikipedia.org/wiki/VoIP
- ویکی فارسی : http://fa.wikipedia.org/wiki/VOIP

بنابراین با تاریخچه ای مختصر سراغ مباحث عملی آن خواهیم رفت.

در سال ۱۹۹۵ شرکت کوچک Vocaltec موفق شد با بهینه سازی کارت های صوتی، میکروفون و بلندگوهای کامپیوتر، اولین سیستم VoIP با نام "Internet Phone" را با پروتکل H.323[۴] ابداع کند و این شروع آن چیزی بود که ما هم اکنون آنرا با نام Skype™ می شناسیم، این شرکت یک سال بعد نیز موفق به گرفتن تاییدیه IPO (تاییدیه پیشبرد عرضه عمومی اولیه) گردید. در پی آن پروتکل های ارتباطی جدید و codec[۵] ها و مولفه های بسته های اطلاعاتی جهت کار با DSP[۶] ها، سیگنالینگ، کنترل تماس و ... باعث به وجود آمدن عدم هماهنگی فوق العاده ای در این عرصه شدند به طوری که کارایی سیستم ها در اولویت دوم قرار گرفت و بالطبع آمادگی جهت ورود به بازار بسیار به کندی پیش می رفت. تا اینکه در سال ۱۹۹۸ با پدید آمدن سرویس های "کامپیوتر-تلفن" برای شرکت های کوچک، و در پی آن سرویس "تلفن-تلفن" (با رابط کامپیوتر) و همزمان با آن تبدیل شدن ایمیل، موبایل و اینترنت به عنوان استانداردهای ارتباط عمومی باعث شد تا تقریبا یک درصد از کل ترافیک صوتی به وسیله ارتباطات VoIP برقرار گردند و این نقطه شروعی برای حرکت شرکت های بزرگ تجهیزات شبکه ای همچون Cisco و Lucent جهت معرفی تجهیزات VoIP و در نتیجه رشد سه درصدی ترافیک صوتی در ایالات متحده گردید. در نهایت در مورد پیشرفت غیرقابل کنترل این صنعت گفتنی است سود خالص VoIP تنها در زمینه فروش محصولات، در سال گذشته میلادی (۲۰۰۸) در آمریکای شمالی چیزی بالغ بر هشت و نیم میلیارد دلار[۷] بوده است!

## ۳. تفاوت های VoIP با PSTN (شبکه تلفنی سوئیچ عمومی) – مزایا و معایب

قبل از صحبت در این مورد لازم به ذکر است که اکنون با فراگیر شدن VoIP و احساس نیاز روزافزون به این پدیده مقرون به صرفه و هوشمند، در کنار شبکه کاملا جا افتاده و آشنای تلفنی، نمی توان کاملا از مزایای یکی در مقابل دیگری چشم پوشید و تمام شرکت های بزرگ سعی در ادغام و محو نمودن خطوط فرضی ایجاد شده مابین این دو تکنولوژی دارند. بنابراین هدف تنها مقایسه توانایی ها و نقاط ضعف بوده و سعی در مغلوب نمودن هیچکدام در مقابل دیگری نمی باشد.

می توان گفت با گرفتن یک شماره تلفن از گوشی تلفن معمولی (کشیده شده توسط زوج سیم به درب منزلتان ) شما در حقیقت وارد سیستم سوئیچینگ مداری[۸] مخابرات می شوید؛ به صورت بسیار خلاصه این سیستم با برقراری یک خط مستقیم (یا مداری) بین دو نقطه، ارتباط دو طرف را برقرار می نماید که این ارتباط در نهایت بر پایه انتقال اطلاعات آنالوگ صوتی بر روی ذوج سیم های مسی[۹] استوار می باشد، با این مقدمه کوتاه در مورد PSTN اکنون می توان تفاوت های این نوع شبکه ها را با شبکه های مبتنی برVoIP به صورت ذیل خلاصه نمود :

---

[۴] استاندارد VoIP که توسط ITU-T (International Telecommunication Union) ارائه شده و مبتنی بر پشته پروتکل ها می باشد. از جمله پروتکل های مورد استفاده می توان : RTP، RTCP، RAS، H.225.0 نام برد؛ در آینده بیشتر از این پروتکل صحبت خواهیم نمود.

[۵] سر واژه ***Co***mpressor/***De***compressor ، که عموما به معنی **فشرده نمودن و از فشردگی خارج کردن** استفاده می شود. {http://en.wikipedia.org/wiki/Codec} ولی در VoIP به معنی سیستم های تبدیل آنالوگ به دیجیتال و بالعکس مورد استفاده قرار می گیرند و در کیفیت صوت و پهنای باند و ... از هم متفاوت می باشند؛ برای اطلاعات بیشتر، ر.ک. به: http://www.voip-info.org/wiki-Codecs

[۶] Digital Signal Processor: پردازشگر سیگنالهای دیجیتال، پردازنده‌های DSP دسته‌ای از پردازنده‌های خاص می‌باشند که بیشتر برای انجام بلادرنگ پردازش سیگنال‌های دیجیتال استفاده می‌شوند؛ ر. ک. : http://en.wikipedia.org/wiki/Digital_signal_processor

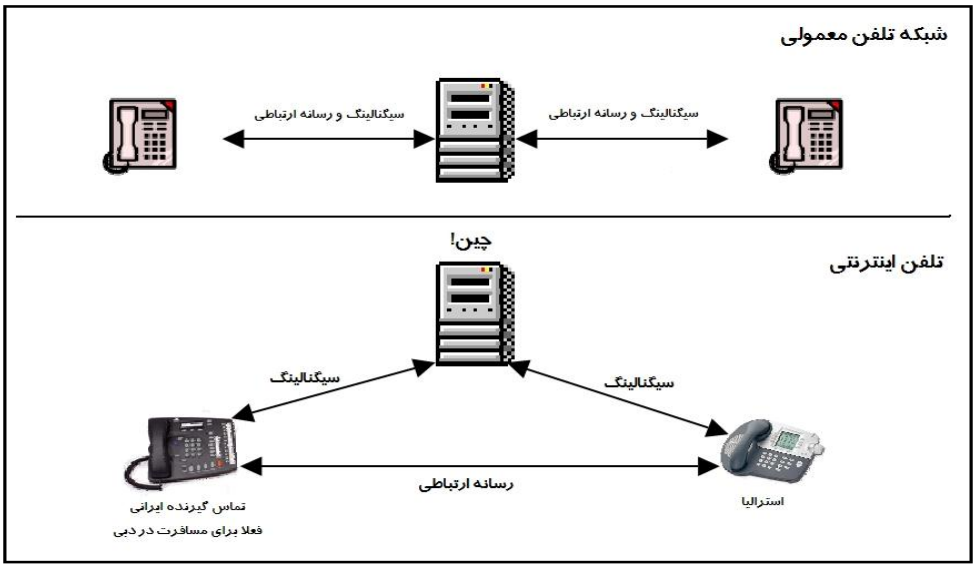
[۷] به نقل از : http://inst.eecs.berkeley.edu

[۸] Circuit Switching ر.ک. : http://en.wikipedia.org/wiki/Circuit_switching

[۹] در اصطلاح بازار معروف به سیم رانژه می باشد : ر.ک. : http://en.wikipedia.org/wiki/Electrical_wiring

| تلفن اینترنتی -VoIP | تلفن معمولی -PSTN |
|---|---|
| تمام کانال ها روی یک خط اینترنتی حمل می شوند.<br>فشرده سازی مکالمات منجر به داشتن پهنای باند ۱۰ kbps می شود (در هر طرف)<br>دریافت سرویس های ارزش افزوده همچون نمایشگر شماره، انتظار مکالمه و...) معمولا بصورت مجانی با سرویس تحویل می گردند.<br>بروز رسانی معمولا فقط شامل پهنای باند و بروز رسانی نرم‌افزاری می باشد.<br>در فاصله های زیاد معمولا هزینه ای ماهانه دریافت می گردد.<br>با رفتن برق، سرویس تلفنی بدون تامین کننده پشتیبان برق در محل تعطیل خواهد شد.<br>با گرفتن شماره اورژانسی نمی توان همیشه قادر به ردیابی محل جغرافیایی تماس گیرنده بود. | خطوط اختصاصی<br>هر خط دارای ۵۶ kbps پهنای باند می باشد (در هر طرف)<br>دریافت سرویس های ارزش افزوده (همچون نمایشگر شماره، انتظار مکالمه، مکالمه گروهی، سرویس پی در پی و...) معمولا نیاز به پرداخت مبلغ جداگانه ای دارند.<br>جهت توسعه یا بروز رسانی نیاز به تجهیزات جدید و اعمال فیزیکی وجود دارد.<br>برای فاصله های دور معمولا هزینه شارژ بالاتر به ازای هر دقیقه یا پالس دریافت می گردد. (یا هزینه های رومینگ)<br>گوشی های تلفنی (البته آن هایی که بدون آداپتور هستند) به هنگام رفتن برق معمولا کار می کنند.<br>وقتی مثلا شماره ۱۱۰ (اورژانسی) را شماره گیری می کنید، محل تماس شما به صورت فیزیکی قابل پیگیری می باشد. |

برای واضح تر شدن مطلب می توان تفاوت حمل داده های صوتی و سیگنالینگ بر روی شبکه های PSTN و VoIP را به شکل زیر درنظر گرفت (البته توجه شود که این فقط یک نمای بسیار ساده شده برای فهم مطلب است و هنگام پیاده‌سازی قضیه بسیار متفاوت از این خواهد بود! ) :



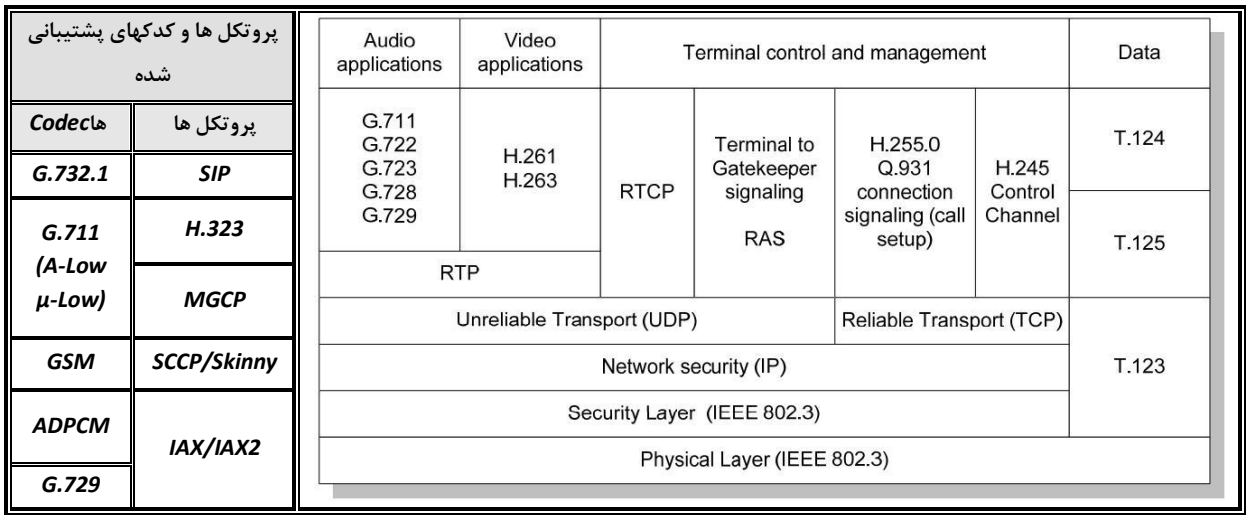

شکل ۱

## ۴. تفاوت گپ اینترنتی (چت) صوتی با VoIP

VoIP (ارسال صدا بر روی پروتکل اینترنت)، یک اصطلاح عمومی است که به خانواده ای از تکنولوژی های انتقال جهت ارتباط صوتی اینترنتی و یا سایر شبکه های سوئیچ بسته اطلاق می گردد. از طرف دیگر سیستم های VoIP عموما به وسیله رابطی با PSTN ها در ارتباط می باشند تا یک ارتباط شفاف جهانی برقرار گردد؛ درصورتی که گفتگوهای اینترنتی روش جدیدی برای ارتباطات اینترنتی است که به یک نرم‌افزار خاص مثل Windows Messenger، Yahoo! Messenger، AOL و ... وابسته بوده و ارتباطی نقطه به نقطه نرم‌افزاری بین دو طرف گفتگو کننده برقرار می نماید که کاملا در سطح نرم افزار می باشد. به عبارت تخصصی تر VoIP در لایه انتقال ( لایه چهارم ) مدل OSI رده بندی می گردد درصورتی که گپ اینترنتی در لایه برنامه ( لایه هفتم ) قرار دارد.

## ۵. معرفی اختصاری پروتکل‌های VoIP

VoIP با ورود خود به عرصه علاوه بر تکنولوژی های همراه و ایده های جدید کلمات اختصاری فراوانی! نیز به همراه آورد که از آن جمله پروتکل های اختصاصی‌اش همچون SIP، IAX و H.323 می باشند. جهت ایجاد دیدگاهی اولیه در ذهن خواننده گرامی ناآشنا به VoIP و همچنین مراجعه بعدی به این مبحث، در شکل زیر پروتکل های مورداستفاده VoIP دسته بندی و نمایش داده شده اند. امید است در فرصت های بعد به هرکدام از این پروتکل ها (مربوط به VoIP) به صورت مفصل پرداخته شود.




| پروتکل ها و کدکهای پشتیبانی شده | |
|---|---|
| پروتکل ها | Codecها |
| SIP | G.732.1 |
| H.323 | G.711 (A-Low μ-Low) |
| MGCP | |
| SCCP/Skinny | GSM |
| IAX/IAX2 | ADPCM |
| | G.729 |

## ۶. روش های دسترسی و استفاده از خدمات VOIP برای کاربران نهایی[۱۰]

در قسمت های بعد، سعی در ارائه یک سناریو پیشنهادی VoIP و پیشبرد آن به صورت عملی خواهیم نمود ولی ابتدا باید پیش زمینه ای - هرچند مختصر- در مورد اجزاء مختلف آن داشته باشیم که از آن جمله می توان به تجهیزات کاربران نهایی سیستم، سرویس دهنده و رابط VoIP با شبکه جاری مخابرات (PSTN) و اینترنت اشاره کرد، ضمنا درصورتی که پروتکل های ذکر شده در شکل قبل برایتان ناآشنا می باشد نگران نباشید؛ به تدریج در مراحل پیشبرد سناریو به هرکدام (درصورت لزوم) خواهیم پرداخت.

اجمالا می توان روش های ارتباط با شبکه VoIP را در سه دسته زیر قرار داد:

مبدل تلفن آنالوگ به دیجیتال یا ATA[۱۱] : این مبدل ها به Gateway نیز مشهورند و عموما به صورت رایگان از طرف سرویس دهندگان خدمات VoIP در اختیار مشترکانشان قرار می گیرد. کاربر شبکه VoIP گوشی تلفن معمولی خود را به وسیله این رابط به شبکه VoIP متصل نماید و بوسیله نرم افزار

[۱۰] VoIP Endpoint devices
[۱۱] Analog Telephone Adaptor

همراه (اگر بصورت Built-in روی مبدل نباشد) آنرا می توان جهت کار با VoIP پیکربندی نمود.

گوشی VoIP[۱۲]: این تجهیزات اکثرا ظاهری مشابه با تلفن های معمولی دارند با این تفاوت که ورودی آن ها به جای اتصال RJ11 که در گوشی های معمولی استفاده می شوند از اتصال RJ45 (مشابه اتصالات شبکه) استفاده کرده و تمام نرم افزارها و سخت افزارهای موردنیاز را به صورت Built-in دارند. هم اکنون این نوع تجهیزات یکی از گزینه های مقرون به صرفه و کاربرپسند (البته نه برای ما که هزینه ای بابت نرم افزار نمی پردازیم!) می‌باشند.

تلفن نرم افزاری[۱۳] : این نوع تماس که از آن به عنوان تماس رایانه به رایانه نیز یاد می‌شود ساده‌ترین راه برقراری یک تماس VOIP می باشد که کاربران کمترین هزینه را برای تماس‌های خود می‌پردازند. در این روش مشترکین برحسب نوع دسترسی به شبکه تنها هزینه خدمات دسترسی را خواهند پرداخت و تماس‌ها چه راه دور و چه نزدیک هیچ هزینه اضافی دیگری را به وی تحمیل نمی‌کنند. (لازم به‌ذکر است که ما نیز از همین نوع تماس در سناریو خود استفاده خواهیم نمود)

## ۷. سرویس دهنده های VoIP[۱۴]:

به سیستم های تجاری تلفنی اطلاق می گردد که صوت را از یک شبکه داده یا سوئیچ عمومی تلفنی (PSTN) تحویل گرفته و آنرا در شبکه جاری تلفنی یا اینترنتی مسیریابی می نماید و اکثرا قادر به برقراری ارتباط با سایر PBX[۱۵] ها می باشند؛ که به هر دو صورت مجازی (نرم افزاری) یا بصورت Built-in در سخت افزار ارائه می گردند. از معروفترین نمونه های نرم افزاری آنها می توان به Asterisk™، Yate، FreeSwitch، PBXWare، Berekele، SipX و OpenPBX اشاره نمود. در این مقاله ما سعی در استفاده و پیکربندی اولیه Asterisk™ خواهیم نمود و مبنای کار پروتکل SIP خواهد بود،

جدول زیر نحوه تراکنش API های Asterisk™ با سایر درگاه ها و همچنین توضیحاتی مختصر را ارائه می دهد.

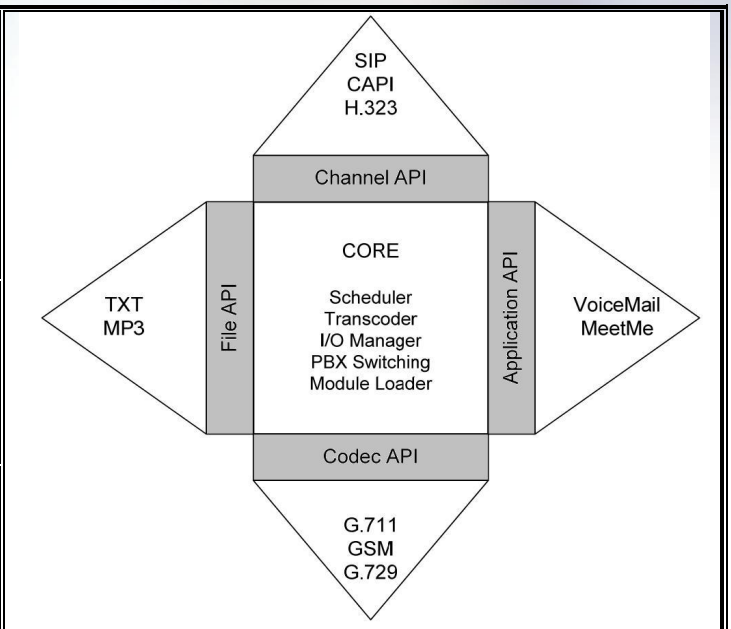
---

[۱۲] IP Phones : جهت دیدن نمونه هایی از این گوشی ها می توانید رجوع کنید به: http://www.vogtec.com/web/hotproduct.htm

[۱۳] Soft Phones : ر.ک. به: http://en.wikipedia.org/wiki/Softphone می‌توانید نمونه های از این نوع تلفن های نرم افزاری VoIP مجانی را در این لینک مشاهده نمایید : http://blog.voipsupply.com/new-products/free-sip-softphone-roundup

[۱۴] IP PBX: که به Voice Gateway نیز مشهور هستند، ر.ک. به : http://en.wikipedia.org/wiki/IP_PBX

[۱۵] Private Branch Exchange: مرکز تلفن خودکار که به یک دفتر یا سازمان با تعداد زیاد تماس های تلفن سرویس داده و مکالمات را مسیریابی می نماید. ر.ک. به : http://en.wikipedia.org/wiki/Private_branch_exchange




Asterisk نرم افزاری متن باز است که تمام کارایی های یک PBX متداول را دارا بوده و تحت لینوکس، BSD، OS X و شبیه سازی آن تحت ویندوز کار می کند. و از طیف وسیعی از پروتکلهای VoIP پشتیبانی کرده و قادر به متصل نمودن شبکه های PSTN و ISDN (BRI, PRI, E1 یا T1) با هزینه سخت افزاری بسیار کمتری می باشد.

Asterisk توسط مارک اسپنسر از Digium ابداع گردید. با وجود این بسیاری از برنامه های کاربردی و توسعه های راهگشای انجام گرفته روی آن توسط توسعه دهندگان دیگر صورت گرفته است.

بسیاری از تولیدکنندگان سیستمهای نرم افزاری PBX VoIP امروزه بجای صرف زمان و هزینه های اضافی جهت ایجاد واسط VoIP از Asterisk بهره می گیرند.

## ۸. مرحله اول پیشبرد سناریو- پیش نیازها :

همانطورکه ذکر شد شما ابتدا نیاز به تهیه مقدمات راه اندازی سناریوی مطرح شده خواهید داشت؛ با فرض اینکه خواننده گرامی آشنایی مقدماتی با مفاهیم اولیه شبکه و UDP ،HTTP داشته و شبکه ای متشکل از چند کاربر و یک سرور دارد

| سیستم مورد نظر | تعداد کانال ها | حداقل نیازمندی ها |
|---|---|---|
| سیستم آزمایشگاهی | کمتر از ۵ | 400-Mhz x86 Cpu – 256M.B. Ram |
| سیستم های SOHO[۱۶] | ۵ تا ۱۰ | 1-GHz x86 Cpu – 512M.B. Ram |
| سیستم های تجاری کوچک | حداکثر ۱۵ | 3-GHz x86 Cpu – 1G.B. Ram |
| سیستم های متوسط تا بزرگ | بیشتر از ۱۵ | Dual/Quad Cpu – ترجیحاً سرورهای چندگانه با معماری توزیع شده |

مراحل کار را ادامه خواهیم داد. لازم به ذکر است به علت فراگیر بودن سیستم عامل ویندوز فرض بر این است که شما سیستم عاملی از این خانواده را روی سیستم های خود نصب کرده اید، درضمن حداقل نیازمندی های سیستم به شرح ذیل می باشد :

پس از دانلود[۱۷] Asterisk™ که تحت پروانه GNU[۱۸] و متن باز می باشد، مراحل نصب آن را روی سرویس دهنده ای که به همین منظور تهیه کرده اید (برای کار آزمایشگاهی و کاهش هزینه ها می توانید بجای سرور اختصاصی بدین منظور، از Virtual Machine[۱۹] استفاده کرده و مراحل نصب را روی آن انجام دهید که در این صورت احتمالا نیاز به دو کارت شبکه مجزا روی سیستم خواهد داشت که یکی از آنها مختص ماشین مجازی Asterisk™ خواهد بود.) طی خواهید نمود که مراحل عادی نصب یک نرم افزار خواهد بود. کافیست مسیر نصب را جهت مراحل پیکربندی به خاطر بسپارید! ضمنا حتما در انتهای نصب Readme را مطالعه کنید و این دفعه مطمئن

[۱۶] Small Office/home Office: به دسته ای از فعالیتهای تجاری اطلاق می شود که بین ۱ تا ۱۰ کارمند دارند.ر.ک. به : http://www.soho.org

[۱۷] مراجعه نمایید به : http://downloads.digium.com/pub/asterisk/releases/asterisk-1.6.0.5.tar.gz

[۱۸] The GNU General Public License : http://en.wikipedia.org/wiki/GNU_General_Public_License

[۱۹] ماشین مجازی : در اینجا منظور نرم افزاریست که یک ماشین (کامپیوتر) را جهت نصب و اجرای نرم افزارهای اجرایی و سیستم عامل مستقل در یک ماشین واقعی شبیه سازی می نماید. ر.ک. به : http://www.vmware.com

باشید که برخلاف اکثر نرم افزارها راهنمایی های کوتاه و بسیار مفیدی در ۲-۳ صفحه ارائه خواهد شد.

نرم افزار بعدی که در مراحل کار به آن نیاز خواهیم داشت، یک تلفن نرم افزاری است که آن هم تحت پروانه GNU و متن باز می باشد.(البته از Windows Messenger نیز می توان استفاده نمود ولی در این صورت امکانات کمتری خواهید داشت!) Softphone موردنظر X-Lite است که نسخه ۳ آن مجانی می باشد[۲۰]

## ۹. شروع کار – راه اندازی مرکز تلفن داخلی:

تا اینجا حداقل مقدمات لازم برای یک سیستم اولیه VoIP PBX داخلی (بدون ارتباط با دنیای خارج!) را فراهم کرده ایم؛ در ادامه کار با تعریف چند کاربر و پیکربندی Asterisk™ و X-Lite یک PBX واقعی را شبیه سازی خواهیم کرد.

۹-۱. پیکربندی Asterisk™ جهت کاربران :

به علت استفاده از پروتکل SIP و چشم پوشی ازDial-Plan و سایر موارد در پروژه آزمایشگاهی ما در این مقاله ، با ۴ فایل پیکربندی که ساختاری مشابه فایل های ini ویندوز دارند، کار خواهیم نمود. این فایل ها در پوشه [Drive]:\cygroot\asterisk\etc (بسته به محل نصب Asterisk™) قرار گرفته اند:

sip.conf : این فایل شامل بسیاری از تنظیمات کاربران می باشد که به صورت پیش فرض توضیحاتی در مورد هر کدام از موارد استفاده درون فایل قرار دارد، ولی ما فقط در اینجا نیاز به تعریف کاربر داشته و بنابراین تنها نیاز به افزودن مشخصات موردنظر در قسمت (Section) [authentication] داریم و از بقیه موارد پیکربندی صرف نظر خواهد شد، بنابراین برای تعریف کاربری به نام ۳۰۰۱ خواهیم داشت :

```
[authentication]
[3001]
type = friend
context = default
username = 3001
host = dynamic
mailbox = 3001
dtmfmode = rfc2833

;... and so on for other users
```

voicemail.conf : به علت اینکه ما صندوق پست صوتی برای کاربران تعریف نخواهیم کرد، کافیست فقط تعریفی اولیه از کاربر در این فایل در section مربوطه یعنی [default] قرار دهیم، یعنی چیزی شبیه این :

```
[default]
; ...
3001 = 4242,user 3001,,,

;... and so on for other users without any other Voicemail config!
```

exten_gvars.inc : در این فایل فقط نام کاربر با پروتکل موردنظر و صندوق صوتی را وارد می کنیم :

```
[globals]

ip3001 = SIP/3001
mbx3001 = 3001
```

extensions.conf : این فایل، مرجع توسعه سیستم می باشد. ما در اینجا فقط لازم است ارتباطات اولیه را در section مربوطه یعنی [internal] برقرار نماییم:

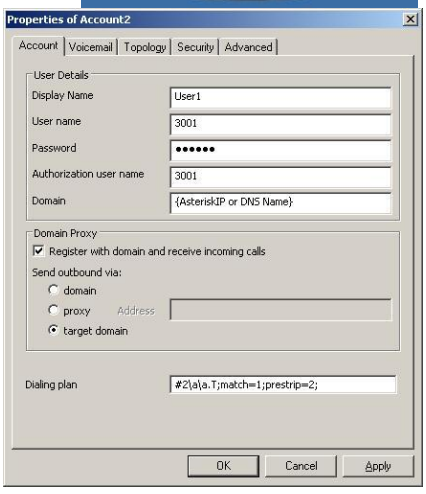
```
[internal]

exten => 3001,1,Dial(${ip3001},30,Ttm)
exten => 3001,2,Voicemail(u3001)
exten => 3001,3,Hangup
exten => 3001,102,Voicemail(b3001)
exten => 3001,103,Hangup

;... and so on for other users
```

[۲۰] جهت دانلود میتوانید به لینک روبرو مراجعه نمایید : http://www.counterpath.net/X-Lite-Download.html و یا درصورت بروز مشکل http://www.download.com/X-Lite/3000-2349_4-10547103.html را امتحان کنید.

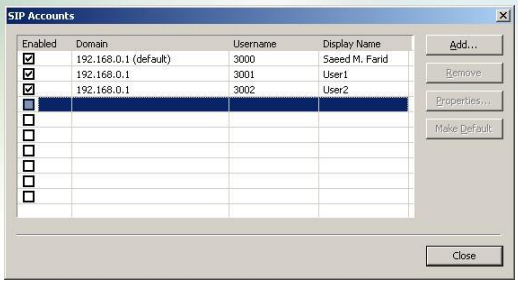
* به همین راحتی توانستیم یک (یا چند) کاربر به سیستم خود اضافه نماییم. البته باید توجه نمایید که این یک سیستم تجاری نیست.

۹-۲. اضافه نمودن کاربر به لیست کاربران X-Lite :

برای اضافه نمودن کاربری که در سرویس دهنده VoIP تعریف کردیم، در سمت سرویس گیرنده (یعنی جایی که X-Lite را نصب کرده ایم) پس از کلیک روی SIP Account Settings کافیست کاربر موردنظر را اضافه نماییم :


این کار را برای تمام کاربرانی که در سیستم تعریف کرده بودید، تکرار نمایید. در نهایت باید توجه نمایید که در هر کلاینت X-Lite را جهت کاربر موردنظر تنظیم نمایید. در شکل بعد سه کاربر برای یک کلاینت تعریف شده اند، باید توجه نمایید که در سمت هر کاربر، حساب منحصر به فردی تعریف نمایید و مثلا یک حساب مشترک برای دو کاربر پیکربندی و تعریف نشود، به عنوان مثال در اینجا (شکل بعد) کاربر مورد نظر دارای سه خط داخلی می باشد!




اکنون درصورتی که از تنظیمات شبکه خود اطمینان دارید، یک بار سرور Asterisk™ خود را راه اندازی مجدد نمایید تا تنظیمات شما برای کاربران تعریف شده در سیستم اعمال گردند.

تبریک! هم اکنون کاربران سیستم شما می توانند با X-Lite خودشان به صورت کاملا مشابه مرکز تلفن خودکار داخلی اما بدون هیچگونه هزینه جانبی با هم صحبت نمایند.

امیدوارم این آشنایی مقدماتی نقطه شروعی برای حرکت به سمت سیستم های تجاری موفق و بزرگ برای شما باشد و بتوانیم در شماره های بعدی کمی بیشتر به جزئیات پیاده سازی و قضایای پشت پرده! سیستم های بزرگتر بپردازیم.

## ۱۰. برخی از شرکت‌های بزرگ ارائه دهنده خدمات VoIP :

شاید زمانی نه چندان دور امکان ردیابی و تهیه آمار توسعه دهندگان و تامین کنندگان تجهیزات و سرویس‌دهندگان VoIP منطقی می‌بود ولی مطمئنا هم اکنون تهیه لیستی از شرکت ای (حتی بزرگ و چند ملیتی) فعال در این زمینه بعید بنظر می رسد، نمونه‌های ذیل چند نمونه انتخابی از بین تمامی آنهاست تا خواننده گرامی از وسعت این تکنولوژی آگاه گردد (گفتنی است که از ذکر تهیه‌کنندگان تجهیزات و دستگاه‌های جانبی صرف‌نظر شده است چون بیشتر بحث ما روی نرم‌افزار و پروتکل‌های ارتباطی می‌باشد) :

VONAGE[۲۱] : تاریخ تاسیس این غول VoIP در سال ۲۰۰۱ بوده و در توصیف اعتبار این کمپانی کافیست بدانیم که طبق گزارش TIA[۲۲] تا انتهای سال ۲۰۰۸ این شرکت دارای ۲۶ میلیون خط تماس تلفنی اینترنتی بوده و تا نیمه دوم سال قبل

[۲۱] http://www.vonage.com

[۲۲] http://www.tiaonline.org : Telecommunications Industry Association

میلادی (۲۰۰۸) سودی بالغ بر ۱.۱ تریلیون دلار ( 1.1x1012 $) کسب نموده است!

AT&T[۲۳] : در مورد شهرت و اعتبار این کمپانی نیاز به توضیح اضافی نیست. این شرکت در عرصه ارتباطات هم اکنون کاملا قادر به تامین داده، صوت، ارتباطات چندرسانه‌ای و ... در هر زمان و هر مکان دلخواه (البته بجز کشور ما که تحریم شده‌ایم!) بوده و از پیشروان عرصه ارتباطات می باشد.

Inode[۲۴] : از پروتکل G.729 (با نرخ ۱۳ kbps) و MPLS جهت تامین کامل کیفیت خدمات استفاده می نماید.

Telekom Austria[۲۵] : این کمپانی اتریشی یکی از بزرگترین تامین کننده های سرویسهای IP برای کمپانی های سرویس دهنده خدمات VoIP می باشد

...

## ۱۱. وضعیت VoIP در کشور :

این موضوع بر همگان عیان است که پیشرفت در سایه رقابت است که حاصل می‌شود، و معمولا مدیران جهت افزایش بازدهی و پیشرفت علمی سعی در ایجاد شرایط رقابتی در بخشهای پیشرو و تحقیقاتی شرکت خود بر می‌آیند. ولی متاسفانه همچون اکثر فعالیت های تجاری و صنعتی در کشور عزیزمان، در زمینه ICT هم شاهد وضعیت مشابهی از لحاظ عدم هماهنگی و تصمیمات کلان ناامید کننده و ضعیف هستیم! این موضوع مکررا در این حوزه مشاهده شده و مخصوصا در مورد تکنولوژی های جدید و دارای آینده درخشان که در سایر کشورها به نتیجه رسیده‌اند، بیشتر چشمگیر است و نمی‌توان آن را تنها به صورت اشتباه یا عدم داشتن سابقه و ... تلقی نمود![۲۶]

در کشورمان بنا به آمار انجمن شرکت‌های اینترنتی، هم اکنون حدود ۱۰۰ شرکت فعال و دارای مجوز VoIP در این زمینه وجود دارند که با توجه به برخی سیاست‌های جدید شرکت زیرساخت و تصویب قانون های دست و پا گیر امکان فعالیت اکثر آن ها به صورت فراگیر و مفید کاملا در ابهام است.

به هر صورت، جهت تلفیق شبکه های NGN[۲۷] با شبکه‌ی فعلی مخابرات و پیشبرد سناریوی قابل پیاده سازی و عملی که ناگزیر به استفاده از فناوری‌های پیشرفته انتقال که آن هم به صورت تنگاتنگ در ارتباط با تکنولوژی های VoIP و WiMax[۲۸] می باشد؛ مدیران صنایع مرتبط با ICT دیر یا زود باید سعی در سپردن کارهای تخصصی به شرکت های متخصص و فعال خصوصی در این زمینه کرده و مسائل مالی و دیدگاه‌های شخصی را در درجه بعدی اهمیت قرار دهند، چرا که در این صنعت جایی برای انحصار طلبی وجود نخواهد داشت.

منابع :


http://en.wikipedia.org
http://www.voip-info.org
http://www.tmcnet.com
http://www.berklix.org
http://www.packetizer.com
http://www.voipforo.com
http://www.vocalcom.com
http://www.cisco.com/go/ciscoit
کتاب شبکه های نسل بعد NGN - نوشته : حسن جند، بابک احسنت، بهنام ولی زاده ،نشر: ناقوس (۱۵ آذر، ۱۳۸۵)
تالیف و ترجمه‌ای آزاد از سعید موسوی فرید


[۲۳] http://www.att.com
[۲۴] http://www.inode.at
[۲۵] http://www.telekom.at
[۲۶] ذکر تمام موارد کار بیهوده‌ای است ولی در حوزه VoIP به عنوان نمونه مدتی قبل قرار شد تصمیماتی درباره مزایده VOIP شرکت زیرساخت اتخاذ شود (http://www.itna.ir/archives/news/010435.php) هنوز چند ماه از مطرح شدن قضیه نگذشته بود که ماه پیش ۳ شرکت (عموما زیرنظر برخی مدیران میانی و البته قدیمی دست‌اندرکار این مزایده) برنده این مناقصه شده و متاسفانه فعالیت بقیه شرکت ها در هاله‌ای از ابهام فرو رفت (بگذریم که به سایر شرکت های خصوصی که اکثر سرمایه تحقیقاتی و علمی خود را روی این حوزه معطوف کرده‌اند، برچسب قاچاق تماس هم زده شد!) ...

[۲۷] Next Generation Network : شبکه‌های نسل بعد، که ایده اصلی آن مبتنی بر شبکه‌ای است که همزمان قابلیت ارسال اطلاعات و سرویس ها (صوت، داده و مجموعه‌های چندرسانه‌ای) به وسیله کپسوله سازی بسته‌ها مثل اینترنت داشته باشد. ر.ک.به: http://www.ngnsp.com و http://en.wikipedia.org/wiki/Next_Generation_Networking

[۲۸] Worldwide Interoperability for Microwave Access: ر.ک.به: http://en.wikipedia.org/wiki/WiMAX و http://www.wimax.com

# استفاده از کنترل های .NET در برنامه های MFC

نویسنده: نیما نیک فطرت

سطح : متوسط

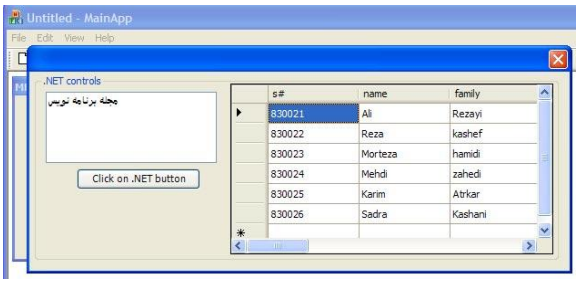
کامپایلر مورد استفاده: Visual C++ و Visual C# نسخه 2008

در حال حاضر هزاران برنامه وسیع و میلیون ها خط کد در داخل و خارج کشور به صورت بومی (native) با کلاس های MFC نوشته شده است که پس از گذشت مدت اندکی از ظهور .NET Framework و درک سادگی کار با آن و سرعت بخشیدن در توسعه نرم افزارها، این نیاز احساس شد که ترکیب این دو کد بومی و مدیریت شده در بسیاری از موارد بسیار سودمند خواهد بود. مایکروسافت از همان دوره آغازین با افزودن امکاناتی به زبان C++ با نام Managed C++ امکان نوشتن برنامه های مدیریت شده و ترکیب آن را فراهم کرد اما همچنان کامل نبود و چند سال بعد C++/CLI با سینتکس متفاوت و جدید به عنوان جایگزین روش قبلی برای کامپایلرهای Visual C++2005 به بعد عرضه کرد و امکاناتی نیز در طول این دوره برای سهولت در ترکیب به MFC اضافه کرد.

از جمله این موارد کلاس هایی مانند CWinFormsDialog ، CWinFormsView، CWinFormsControl و سایر موارد اضافه شده به MFC است که در این مقاله با چند مثال کاربردی به توضیح CWinFormsDialog می پردازیم. این مقاله برگرفته شده از نمونه مثال ها و آموزش های MSDN در سایت مایکروسافت می باشد که با جزییات و نکات بسیار بیشتر بیان شده است و به منظور کاربردی کردن این آموزش تصمیم گرفته شد تا در انتها یک DataGridView را به برنامه اضافه کنیم و از یک پایگاه داده SQL server به وسیله کنترل های ساخته شده توسط C# و با کمک از C++/CLI در پروژه بومی خود استفاده کنیم.

نکته مهمی که در کنار این موضوع یاد خواهید گرفت این است که چگونه برنامه MFC خود را در حالت CLR کامپایل کنید تا فقط دیالوگ های مرتبط با کنترل های .NET نیازمند کامپایل شدن از طریق CLR باشند و بقیه برنامه همچنان بدون تغییر native باقی بماند.

پیش نیاز: هرچند در این آموزش، آشنایی با مطالب اولیه موارد فوق مورد نیاز است، اما سعی شده مطالب با جزییات کامل و ساده بیان شود تا قابل استفاده توسط کاربران مبتدی نیز باشد.




شکل ۱

## نحوه عملکرد:

ما در این برنامه سه پروژه خواهیم ساخت که همگی در داخل یک Solution قرار خواهند گرفت و فایل نهایی اجرایی ما را ایجاد خواهند کرد. مورد اول پروژه اصلی MFC ما که همچنان native باقی خواهد ماند، مورد دوم control library توسط زبان C# که کنترل های Windows Forms مورد نیاز ما را ایجاد می کند و در قالب DLL برایمان ساخته می شود و سومین پروژه نیز یک فایل DLL دیگر است که ارتباط برنامه MFC ما را با کنترل های ساخته شده در پروژه دوم برقرار می کند، از آنجایی که تنها این DLL توسط CLR کامپایل می شود و ما نیز تنها در آن از C++/CLI استفاده می کنیم در نتیجه برنامه اصلی ما همچنان بومی می ماند.

- ساخت برنامه بومی MFC که کنترل های Windows Forms را در خود قرار می دهد:

در این قسمت یک برنامه ساده تک سندی MFC خواهیم ساخت تا به عنوان برنامه اصلی ما عمل کند.

۱- از طریق منوی File و گزینه New بر روی Project کلیک کرده و یک پروژه MFC Application بسازید. در فیلد Name عنوان پروژه را "MainApp" بنامید و Solution Name را "MFC_CLR" قرار دهید.

در تنظیمات ظاهر شده در هنگام ساخت در قسمت Application Type نوع پروژه را Single Document انتخاب کنید.

۲- از طریق Resource View به فولدر Menu رفته و با دابل کلیک کردن بر روی IDR_MAINFRAME آن را باز کنید و یک گزینه جدید با عنوان "Open A Form" در منو اضافه کنید و ID آن را در پنجره Properties به ID_FILE_OPENAFORM تغییر دهید.

۳- بر روی این گزینه جدید اضافه شده در نمای طراحی منو کلیک کنید و Add Event Handler را انتخاب کنید. در پنجره کلاس را CmainMFCApp انتخاب کنید و عنوان "OnFileOpenaform" را برای پیام قرار دهید. سپس Add and Edit را کلیک کنید.

تذکر: در قسمت های بعدی در داخل آن از دستوری برای ساخت پنجره مورد نظر ما استفاده خواهیم کرد.

۳- پروژه را از طریق منوی Build کامپایل کرده و برنامه را مشاهده کنید.

- ساخت پروژه Control Library توسط زبان #C:

در حالی که پروژه قبلی باز است مراحل زیر را انجام دهید.

۱- از طریق منوی File و گزینه New بر روی Project کلیک کرده و یک پروژه Control Forms Windows Library با #C بسازید.

نام پروژه را "DatabaseControlLibrary" قرار داده و حتما در هنگام ساخت پروژه در زیر نام و در قسمت Solution گزینه Add to Solution را انتخاب کنید تا در همین پروژه کنونی اضافه شود.

۲- کنترلی به عنوان UserControl1.cs ساخته می شود، از طریق پنجره Properties نام آن را به DatabaseControl.cs تغییر دهید، سپس وارد فایل های سورس کد آن شوید و هر جایی که عنوان UserControl1 وجود داشت آن را به DatabaseControl تغییر دهید.

DatabaseControl.cs را انتخاب کرده و در پنجره طراحی موارد زیر را همانند شکلی که در اول این مقاله قرار داده شده است بسازید:

TextBox، Button ، DataGridView به همراه یک BindingSource.

عناوین این کنترل ها را به ترتیب به موارد زیر تغییر نام دهید:

msgTextBox، okButton ، dataGridView_STD و bindingSource_STD

در قسمت های بعدی در برنامه برای این کنترل ها، یک دیالوگ نیز خواهیم ساخت.

۳- به فایل DatabaseControl.Designer.cs بروید و تعاریف تمامی کنترل های فوق را از حالت private به public تغییر دهید تا پس از کامپایل بتوانیم آن ها را در برنامه خود فراخوانی کنیم.

۴- از طریق Build پروژه را کامپایل کنید تا DLL ساخته شود.

- ساخت یک DLL حاوی کدها و دیالوگ های ارتباطی ما با Windows Form:

ما برای ساخت این پروژه از MFC extension DLL استفاده می کنیم تا کدهای کلاس ها و دیالوگ های ما را نیز در بر گیرد. توجه داشته باشید که دو دیالوگ خواهیم ساخت که اولی یک دیالوگ معمولی و مشتق شده از CDialog هست که فقط برای نمونه ساخته شده است و دارای دکمه Open برای باز کردن دیالوگ دوم می باشد، دیالوگ دوم نیز یک دیالوگ خالی است و User control ساخته شده توسط #C ما در آن قرار می گیرد.

۱- از طریق منوی File و گزینه New بر روی Project کلیک کرده و پروژه MFC DLL را انتخاب کنید. عنوان آن را "CLRmixed" بنامید و همانند قبل از Add to Solution استفاده کنید.

در Wizard ظاهر شده با کلیک بر روی Next نوع آن را حتما MFC extension DLL انتخاب کنید و Ffinish را کلیک کنید.

۲- جهت سهولت در انجام کار، فایل CLRmixed.def را از پروژه حذف کنید. برای تکمیل کار در حالی که در Solution Explorer، همین پروژه DLL انتخاب شده است از طریق منوی Project گزینه Properties را انتخاب کنید و به مسیر زیر بروید:

Linker -> Input -> Module definition file

سپس محتویات آن فیلد را (.\CLRmixed.def) حذف کنید.

دقت کنید که از بالای همین پنجره در قسمت Configuration این کار را حتما در حالت All

Configurations انجام دهید تا در هر دو حالت Release و Debug انجام شود.

۳- دوباره در همین دیالوگ قبل به مسیر زیر بروید:

General -> Common Language Runtime support

و آن را به /clr تغییر دهید تا این DLL بر اساس CLR کامپایل شود و قادر به استفاده از C++/CLI باشیم.

۴- کد زیر را بعد از آخرین include اما قبل از آخرین endif در فایل stdafx.h اضافه کنید:

```
#include <afxwinforms.h>
```

تذکر: برای استفاده از کلاس CWinFormsDialog به این هدر فایل نیازمند هستیم.

۵- در Solution Explorer بر روی این پروژه CLRmixed راست کلیک کنید و References را انتخاب کنید، در پنجره ظاهر شده با Add New Reference به قسمت Project بروید و پروژه DatabaseControlLibrary را انتخاب کنید تا این DLL به User Control دسترسی داشته باشد.

تذکر: شما می توانید به جای reference ، مستقیم در فایل هایی که می خواهید از آن استفاده کنید کد زیر را اضافه کنید:

```
#using <DatabaseControlLibrary.DLL>
```

۶- برای ساخت دیالوگ اول، از طریق منوی Project و گزینه Add class به MFC class بروید. در wizard ظاهر شده نام کلاس را "CDialogOpen" بنامید و base class یا همان کلاس پایه را CDialog انتخاب کنید و کلاس را بسازید.

۷- از طریق Resource View به نمای طراحی دیالوگ ساخته شده بروید و دکمه OK را به Open تغییر نام دهید و ID آن را نیز به ID_OPEN_NET تغییر دهید. با راست کلیک کردن بر روی این دکمه و Add event handler رویداد BN_CLICKED را بسازید و نام آن را "OnBnClickedOpenNet" قرار دهید.

تذکر: دستور مورد نیاز، در مراحل بعدی در داخل این رویداد نوشته می شود.

دقت کنید که برای این دیالوگ، آن را از نوع Tool Window انتخاب کردیم.


**شکل ۲**

۸- اکنون وقت نوشتن تابعی است که DLL ما صادر خواهد کرد و از آن برای فراخوانی این دیالوگ به صورت modal در برنامه MFC بومی خود استفاده خواهد شد. پس فایل CLRmixed.cpp را به شکل زیر تعریف کنید:

```
#include "stdafx.h"
#include "DialogNetControl.h"
#include "DialogOpen.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#endif
extern "C"    AFX_EXT_API    void WINAPI
OpenNetDialog()
{
   CDialogOpen  openDlg;
    openDlg.DoModal();
}
```

۹- قالب تابع فوق را در بالای فایل MainApp.cpp در برنامه native خود به شکل زیر تعریف کنید:

```
extern       "C"       void       WINAPI
OpenNetDialog();
```

و رویداد OnFileOpenaform را که در پروژه اول ایجاد کرده بودید به شکل زیر تغییر دهید:

```
void CMainAppApp::OnFileOpenaform()
{
   // Open my Windows Form from my
Native MFC application
  OpenNetDialog();
}
```

تذکر: دقت کنید که برای فراخوانی و استفاده از این DLL نیاز نیست پروژه MainApp در حالت CLR کامپایل شود. بلکه فقط باید CLRmixed.lib را به لینکر اضافه کنید. به این شکل که ابتدا به مسیر زیر رفته:

Project->Properties->Linker -> Input -> Additional dependencies

و در آن فیلد، عبارت زیر را بنویسید تا بتوانید از DLL در برنامه استفاده کنید:

"..\release\ CLRmixed.lib"

۱۰- برای ساخت دیالوگ دوم به همان شکل مرحله ۶ عمل کنید و کلاسی با نام CDialogNetControl از CDialog به ارث ببرید.

۱۱- حال برای این دیالوگ دوم در دو فایل CDialogNetControl.cpp و CDialogNetControl.h ساخته شده، هر جایی که عبارت CDialog وجود دارد آن را با عبارت زیر جایگزین کنید تا از آن به ارث ببرد:

CWinFormsDialog< DatabaseControlLibrary:: DatabaseControl>

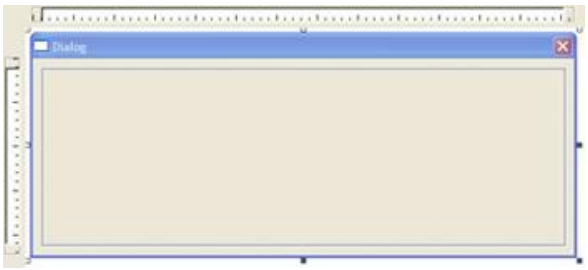
به این شکل DatabaseControl محتویات داخلی دیالوگ را تشکیل می دهد.

۱۲- CWinFormsDialog از CDialog مشتق شده است و این امکان را فراهم می کند که به طور خودکار دیالوگی ساخته و DatabaseControl را در خود جای دهد، اما از آنجایی که اندازه آن خودکار با اندازه DatabaseControl ما تنظیم نمی شود، ما دیالوگ ساخته شده کنونی را خالی می کنیم و به اندازه دلخواه به عنوان یک قالب استفاده می کنیم.

پس برای این کار دکمه های موجود در دیالوگ دوم ساخته شده توسط این کلاس را حذف کنید و ID آن را به IDD_DIALOGNETCONTROL تغییر دهید.

از آنجایی که به منابع دیالوگ ساخته شده احتیاج نداریم قطعه کد زیر را از فایل h. آن حذف کنید:

```
// Dialog Data
        enum        {        IDD        =
IDD_DIALOGNETCONTROL }; // حذف گردد
```



۱۳ - سازنده کلاس را به شکل زیر تغییر دهید تا از دیالوگ ما به عنوان قالب استفاده کند :

```
#include "Resource.h"
CDialogNetControl::CDialogNetControl(CWnd* pParent /*=NULL*/)
  :CWinFormsDialog< DatabaseControlLibrary:: DatabaseControl>(
IDD_DIALOGNETCONTROL, pParent)
{

}
```

۱۴- هدر فایل DialogNetControl.h را در فایل DialogOpen.cpp تعریف کنید:

```
#include "DialogNetControl.h"
```

و قطعه کد زیر را در رویداد OnBnClickedOpenNet که در مرحله ۷ ساخته بودید بنویسید:

```
void CDialogOpen::OnBnClickedOpenNet()
{
    CDialogNetControl  dlg;
    dlg.DoModal();
}
```

۱۵- در Solution Explorer بر روی پروژه mainMFC راست کلیک کنید و گزینه Set as StartUp Project را انتخاب کنید تا بعد از کامپایل، فایل اجرایی شروع شود.

۱۶- موارد زیر را برای تعیین وابستگی ها و ترتیب کامپایل انجام دهید:

- بر روی پروژه mainMFC راست کلیک کرده و Add Dependencies را انتخاب کنید، در دیالوگ مربوطه CLRmixed را انتخاب کنید.
- به همین شکل بر روی CLRmixed نیز راست کلیک کرده و Add Dependencies را انتخاب کنید و وابستگی را برای DatabaseControlLibrary انتخاب کنید.

۱۷- کل پروژه ها را از طریق Build Solution کامپایل کنید و با استفاده از Run اجرا کنید.

به این شکل با کلیک کردن بر روی دکمه Open یک فرم نمایش داده می شود که کنترل های ساخته شده در C# را نشان می دهد.

- نحوه استفاده و ارتباط با User Control ساخته شده :

اکنون با انجام مراحل زیر یاد خواهید گرفت که چگونه محتویات این کنترل ها را با C++/CLI بخوانیم و بر روی دکمه کلیک کنیم.

۱- از طریق Class view بر روی کلاس CDialogNetControl بروید و در پنجره Properties ، با انتخاب Overrides متد OnInitDialog را بارگذاری کنید. تا در هنگام شروع به کار دیالوگ بتوانیم رویدادهایی را آماده کنیم.

تذکر: در صورتی که این امکان قابل مشاهده نیست، شما می توانید به صورت دستی کد زیر را در قسمت public کلاس DialogNetControl اضافه کنید:

```
virtual BOOL OnInitDialog();
```

سپس تعریف زیر را نیز در فایل DialogNetControl.cpp انجام دهید:

```
BOOL CDialogNetControl::OnInitDialog()
{
CWinFormsDialog< DatabaseControlLibrary:: DatabaseControl>::OnInitDialog();

 GetControl()->okButton->Click += MAKE_DELEGATE(System::EventHandler,
     OnOkButton);
   return TRUE;
}
```

تذکر: GetControl متدی از کلاس CWinFormsDialog است که مرجعی به کنترل ساخته شده ما در C# بر می گرداند، لذا کافیست همواره به شکل فوق عمل کنیم و با فراخوانی MAKE_DELEGATE در طرف دوم، یک رویداد را به آن کنترل اختصاص دهیم. در اینجا رویداد OnOkButton با کلیک کردن بر روی okButton در کلاس کنونی فراخوانی می شود.

۲- از انجایی که OnOkButton برای رویداد Windows Forms هست، باید از C++/CLI استفاده کرد. پس آن را در فایل DialogNetControl.h به شکل زیر تعریف می کنیم:

```
virtual void OnOkButton (
System::Object^ sender,
System::EventArgs^ e );
BEGIN_DELEGATE_MAP( CHostForWinForm )
EVENT_DELEGATE_ENTRY( OnOkButton,
System::Object^, System::EventArgs^ );
END_DELEGATE_MAP()
```

در فایل DialogNetControl.cpp نیز به صورت زیر تعریف OnOkButton را انجام دهید:

```
void CDialogNetControl::OnOkButton (
System::Object^ sender,
System::EventArgs^ e )
{

System::Windows::Forms::MessageBox::Sh
ow( GetControl()->msgTextBox->Text );
}
```

همانطور که مشاهده می کنیم، با کلیک بر روی okButton رشته موجود در msgTextBox را از طریق GetControl دریافت کرده و آن را در یک MessageBox از Class Library که در .NET موجود است نمایش می دهیم.

۳- برنامه را به طور کامل کامپایل و اجرا کنید، سپس متنی در فیلد بنویسید و دکمه را کلیک کنید.

- نحوه استفاده و ارتباط یک پایگاه داده SQL server با DataGridView:

با این مثال آخر به قدرت این ترکیب و راحتی استفاده از امکانات .NET در MFC پی خواهید برد. لذا به منظور کاربردی کردن این آموزش و فقط جهت یک نمونه، کد زیر را در رویداد OnOkButton بنویسید:

```
#using <System.Data.DLL>
using namespace System;
using namespace System::Data;
using namespace System::Data::SqlClient;
void CDialogNetControl::OnOkButton ( System::Object^ sender, System::EventArgs^ e )
{
     System::String^ connectionString = "Data Source=.\\SQLEXPRESS;"
                                        "AttachDbFilename=C:\\myDatabase.mdf;"
                                        "Integrated Security=True;Connect Timeout=30;User
Instance=True";
```

```
  SqlConnection^ myConnection = gcnew SqlConnection( connectionString );
  SqlCommand^ command = gcnew SqlCommand( "Select * From STD", myConnection );
  SqlDataAdapter^ adapter = gcnew SqlDataAdapter;
  adapter->SelectCommand = command;
  DataTable^ table = gcnew DataTable;
  adapter->Fill( table );
  GetControl()->dataGridView_STD->AutoGenerateColumns = true;
  GetControl()->bindingSource_STD->DataSource = table;
  GetControl()->dataGridView_STD->DataSource =
       GetControl()->bindingSource_STD;
}
```

در مثال فوق پایگاه داده ای با نام myDatabase.mdf در درایو C قرار داده شده است و از SQL Server Express استفاده شده است. در SqlCommand نیز فقط جهت نمونه یک SELECT ساده از جدولی با نام STD گرفته شده است. در انتها از طریق DataTable اطلاعات را گرفته و به bindingSource_STD که همراه dataGridView_STD اضافه کرده بودیم متصل می کنیم تا نمایش داده شود.

تذکرات:

- شما به همین شکل می توانید از سایر پایگاه داده ها و ارتباط ها مانند OleDbDataAdapte، OdbcDataAdapter، SqlDataAdapter، OracleDataAdapter استفاده کنید.
- در C++/CLI مواقعی که با ^ هندلی به یک شئی اختصاص می دهیم و از gcnew برای تخصیص انواع بر روی garbage-collected heap استفاده می کنیم، پس از پایان عملیات به طور خودکار Garbage Collector فراخوانی مخرب و آزاد کردن حافظه را انجام می دهد. اما با این حال در مواردی برای اطمینان از فراخوانی مخرب و جلوگیری از memory leak مخصوصا برای انواع غیر مدیریت شده در کلاس های مدیریت شده خود، می توانید پس از اتمام استفاده آن ها، از delete استفاده کنید.

مثلا در پایان تابع، یا کلا با انجام عمل override متد مجازی بستن دیالوگ با نام DestroyWindow و فراخوانی delete برای کلاس در داخل آن، و یا با استفاده از try/finally به شکل زیر:

```
MyClass^ cls = gcnew MyClass();
try
{
      این در را کلاس قسمت استفاده کنید
//
}
finally
{
   delete  cls;
}
```

دربارهٔ نویسنده: چندین سال تجربه ام در زمینه گرافیک باعث شد به توسعه ابزارهای گرافیکی، طراحی بازی و کلا برنامه نویسی علاقه مند شوم، به همین منظور برنامه نویسی را با C/C++ آغاز کرده و به صورت تخصصی win32 API را فراگرفتم. از دانشگاه با لیسانس نرم افزار فارغ التحصیل شدم. اولین برنامه ویندوزی که نوشتم یک بازی دو بعدی بود، و بعد از آن دانش خود را در این حوزه با برنامه نویسی سه بعدی در DirectX و HLSL کامل تر کردم. بیش از یک سال است که مدیریت تالارهای بخش C در سایت برنامه نویس را بر عهده دارم.

## Guy Steele

آقای Steele یکی از افراد خبره و سرشناس جهانی در زمینه ی زبان های برنامه نویسی هستند. اگر تا حالا اسم زبان Scheme به گوشتان خورده باشد ، اسم ایشان را هم حتما شنیده اید.

وی در میسوری امریکا متولد شده و اولین مدرک دانشگاهی اش را در رشته ی ریاضیات کاربردی از هاوارد در سال ۱۹۷۵ گرفت؛ در سال های ۱۹۷۷ و ۱۹۸۰ هم به ترتیب مدارک کارشناسی ارشد و Ph.D را در زمینه ی علوم کامپیوتر از MIT دریافت کرد، مدتی هم در دانشگاه کارنگی ملون TA بود. در سال ۱۹۹۴ به شرکت Sun پیوسته و جزئی از تیم جاوا شد. دوران طلایی عمر ایشان دورانی بود که همراه با Gerald Sussman (و بعد ها با Richard Gabriel) روی زبان لیسپ کار کرده و مقالات مهمی در این زمینه نوشتند. ایشان یکی از طراحان زبان Scheme هستند که یکی از دیالکت های زبان لیسپ است. همچنین طراح مجموعه دستورات Emacs بودند. آقای Steele عضو کمیته های استاندارد سازی X3J11 (زبان سی ) و X3J3 (فورترن) بوده و الان مسئول X3J13 (Common LISP) هستند. همچنین جزو تیم IEEE برای استاندارد زبان Scheme بودند. ایشان در شرکت سان specification زبان جاوا را نوشتند. (همینطور CLTL2 که specification زبان Common LISP محسوب می شود کار ایشان است) در سال ۲۰۰۵ سرپرستی تیمی از محققین را در شرکت سان بر عهده گرفتند تا یک زبان جدید با نام Fortress را به وجود بیاورند (با هدف جایگزین کردن فرترن)

از دیگر کارهای ایشان می شود به کتاب های C: A Reference Manual و High Performance Fortran Handbook اشاره کرد.

### از افتخارات ایشان:

جایزهٔ ACM Grace Murray Hopper در سال ۱۹۹۸

ACM Fellow در سال ۱۹۹۴

عضو آکادمی ملی مهندسی ایالات متحده در سال ۲۰۰۱

Sun Fellow در ۲۰۰۳

جایزهٔ Dr. Dobb's Excellence in Programming در سال ۲۰۰۵

### مقالات مهم Guy Steele:

Scheme: An Interpreter For Extended Lambda Calculus (همراه با Sussman)

Debunking the "Expensive Procedure Call" Myth or, LAMBDA: The Ultimate GOTO

تز Ph.D : The Definition and Implementation of a Computer Programming Language

The Art of the Interpreter (همراه با Sussman)

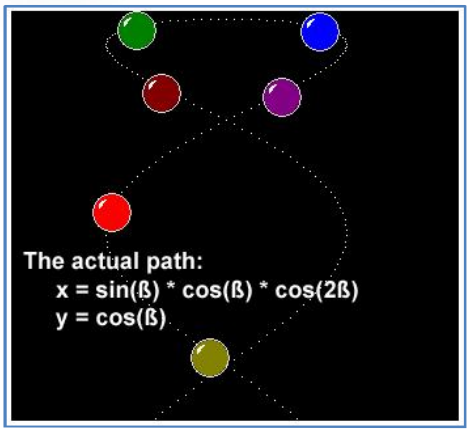
The Evolution of Lisp (همراه با Richard Gabriel)

آقای Steele عضو دائمی فدراسیون شطرنج ایالات متحده هم هستند؛ از ویژگی های بارز ایشان شیوه ی نگارش دقیق و روش توضیحشان در مسائل مربوط به جزئیات زبان های برنامه نویسی است.

منابع:

http://en.wikipedia.org/wiki/Guy_L._Steele,_Jr.
h__p://www.lisp50.org/schedule/schedule/steele.html

# ساخت محافظ صفحه نمایش




نویسنده: مهدی موسوی

برای دریافت نمونه کد مقاله، می توانید به آدرس زیر رجوع کنید:

http://www.codeproject.com/KB/graphics/ballfusion.aspx

سطح : متوسط

چکیده :در این مقاله، با اصول اولیه ساخت محافظ صفحه نمایش در ویندوز ، با استفاده از VC++ در Win32 آشنا خواهید شد. این محافظ متشکل است از ۷ توپ رنگی که بر روی مسیری سینوسی شکل به دنبال یکدیگر در حرکت خواهند بود. از آنجایی که مقاله و کد متصل شده به آن ۸ سال پیش نوشته شده بود، از کامپایلر VC++ 6.0 استفاده شده است. بنابراین اگر قصد دارید آن را در VS2005 یا VS2008 کامپایل کنید، از اضافه کردن کتابخانه comctl32.lib به لیست کتابخانه های پروژه اطمینان حاصل کنید تا هنگام کامپایل کردن کد، با خطا مواجه نشوید.

## صفحه نمایش

این مقاله در مورد محافظ های صفحه نمایش در ویندوز صحبت می کند و به شما نشان می دهد چگونه محافظ خود را بسازید.

## تشکر و قدردانی

مایلم تا این مقاله را با تشکر از کلیه افرادی که ایمیل های بسیاری برای قدردانی از مقالات قبلی من، بخصوص مقاله ISAPI Extension برایم ارسال کردند، آغاز نمایم. نمی توانید تصور کنید که آن ایمیل ها چقدر الهام بخش بود. من برای زمان شما ارزش قائل هستم و امیدوارم تا این مقاله نیز پاسخگوی نیازهای شما باشد.

## مقدمه

بر خلاف زندگی انسانها که امروزه به سختی می توان از آن در برابر حملات دیگر به اصطلاح انسان ها محافظت نمود، می توان به راحتی و توسط یک محافظ از فسفر سوزی صفحه نمایش حفاظت کرد. امروز، قصد دارم تا درباره محافظین صفحه نمایش در ویندوز صحبت کنم، مقوله ای که بیشتر برنامه نویسان مایلند تا درباره آن بدانند.

## اساس

محافظ صفحه نمایش، نرم افزاری از نوع Win32 است که پس از گذشت مدت زمانی که موشواره و صفحه کلید بی استفاده باقی ماندند، به صورت خودکار به اجرا در می آید. هنگامیکه این زمان به سر رسد، کاربر ممکن است با صفحه ای خالی یا انیمیشنی پیچیده روبرو شود. پس از آن، اگر کاربر کلیدی را

روی صفحه کلید فشار دهد یا موشواره را به حرکت در آورد، صفحه به وضعیت قبلی خود که با شروع اجرای محافظ غیب شده بود، باز خواهد گشت.

برای ایجاد محافظ صفحه، شما دو روش در پیش روی دارید: در روش اول، شما می توانید یک نرم افزار از نوع Win32 ایجاد کرده و پیام های مورد نیازی که به سوی پنجره شما ارسال می شود را، پردازش کنید. در این روش، برنامه شما علاوه بر ایجاد صفحه ای Full Screen، باید توسط window procedure خود اقدام به پردازش پیام های ارسال شده به پنجره شما نماید. در window procedure مزبور، شما کدی مشابه کد زیر می توانید داشته باشید:

```
switch(message)
{
    case WM_SYSCOMMAND:
        if(wParam == SC_SCREENSAVE ||
wParam == SC_CLOSE)
            return FALSE;
    break;

    case WM_LBUTTONDOWN:
    case WM_MBUTTONDOWN:
    case WM_RBUTTONDOWN:
    case WM_KEYDOWN:
    case WM_KEYUP:
    case WM_MOUSEMOVE:
        //Quit the application here

    break;
}
```

به بیان دیگر، شما باید تکه های کد مورد نیاز محافظ صفحه نمایش را Hard Code کرده و منتظر پیام های عمومی محافظین صفحه نمایش در ویندوز باشید.

اما روش دوم، استفاده از کتابخانه ای است که همراه کامپایلر شماست و کلیه Hard Coding های فوق الذکر (و حتی بیشتر) را انجام داده است. در این روش، شما تنها بر روی مساله واقعی متمرکز می شوید که همان ایجاد افکت های بصری است. بدیهی است که در این مقاله برای اجتناب از اختراع مجدد چرخ از این کتابخانه استفاده خواهیم کرد.

کتابخانه محافظ صفحه نمایش (scrnsave.lib) حاوی تابع WinMain است. این تابع همراه با مخلفات دیگر، کلاس پنجره ای را ثبت می کند که چیزی شبیه کد زیر است:

```
WNDCLASS cls;

cls.hCursor = NULL;
cls.hIcon = LoadIcon(hInst,
MAKEINTATOM(ID_APP));
cls.lpszMenuName = NULL;
cls.lpszClassName =
"WindowsScreenSaverClass";
cls.hbrBackground =
GetStockObject(BLACK_BRUSH);
cls.hInstance = hInst;
cls.style = CS_VREDRAW | CS_HREDRAW |
CS_SAVEBITS | CS_DBLCLKS;
cls.lpfnWndProc = (WNDPROC)
ScreenSaverProc;
cls.cbWndExtra = 0;
cls.cbClsExtra = 0;
```

به عبارت دیگر، این کد یک پنجره با پس زمینه سیاه رنگ، بدون اشاره گر موشواره و همراه با آیکونی که با ID_APP مشخص شده است، ثبت می کند. علاوه بر آن، نام window procedure خود را ScreenSaverProc تعیین می کند. این تابع قلب هر محافظ صفحه نمایشی است و تقریبا کلیه کدها در درون این تابع نوشته می شوند.

این تابع، همانند دیگر window procedure ها، بصورت زیر تعریف شده است:

```
LONG ScreenSaverProc(HWND hWnd, UINT
message, WPARAM wParam, LPARAM
lParam);
```

که در آن hWnd هندل پنجره Desktop، message پیام ارسالی به پنجره محافظ و wParam و lParam اطلاعات پنجره-محور مضاعف است. اگر سری مقالات قبلی "MFC در برابر Win32" مرا به یاد بیاورید، به خاطر خواهید آورد که هنگام عدم نیاز به پردازش یک پیام، باید پیام را به تابع DefWindowProc پاس کنیم. در مقابل، هنگام توسعه محافظ صفحه، باید پیام را به تابع DefScreenSaverProc پاس دهیم. این تابع همانند DefWindowProc است با این تفاوت که مراقب دیگر پیام های مورد نیاز برای اجرای محافظ نیز می باشد.

در هر حال، نام فایل header کتابخانه، scrnsave.h است که در شاخه include کامپایلر قرار گرفته و حاوی پیش الگوهایی برای پشتیبانی از کتابخانه محافظ صفحه نمایش است. چند متغیر سراسری در این کتابخانه تعریف شده که نسخه extern آنها در فایل header فوق الذکر قرار گرفته است. دو تا از آن متغیرهای مهم بدین شکل تعریف شده اند:

```
extern HINSTANCE hMainInstance;
extern BOOL fChildPreview;
```

که در آن، hMainInstance هندل به محافظ بوده و fChildPreview نیز بیانگر آنست که آیا محافظ در حالت

پیش نمایش در حال اجراست یا خیر. نگران نباشید، به زودی هر یک را بررسی خواهیم کرد.

## اهداف

ما قصد داریم تا محافظ صفحه نمایشی تحت عنوان "ترکیب توپ ها" بسازیم که متشکل از ۷ توپ متحرک روی مسیری سینوسی شکل است:

```
x = sin(ß) * cos(ß) * cos(2ß)
y = cos(ß)
```

که در آن ß عددی مثبت در بازه ۰ و ۳۶۰ می باشد. شما می توانید این فرمول را در صورت تمایل تغییر دهید.

## دیگر ملاحظات

محافظ "ترکیب توپ ها" فاقد پنجره پیکربندی است بدین معنا که اگر شما کلید Settings را در Control Panel مربوط به محافظ های صفحه نمایش فشار دهید، پاسخی دریافت نخواهید کرد. این مساله فقط به دلیل کمبود وقت است. در واقع، مدیرم پس از مشاهده این محافظ بر روی صفحه نمایشگر در شرکت، به من اعتراض کرد. در نتیجه، از توسعه آن خودداری کردم و بدین دلیل است که این محافظ پنجره پیکربندی ندارد.

## آغاز بکار

برای شروع، MSVC++ 6.0 را اجرا کنید. از منوی فایل، گزینه New را انتخاب کنید. در حالیکه گزینه Projects را انتخاب کرده اید، Win32 Applica on را روشن کرده و در محل Project Name، کلمه BallFusion را نوشته و کلید OK را فشار دهید. اکنون گزینه "An empty project" را انتخاب کرده و به ترتیب کلیدهای Finish و OK را فشار دهید. اکنون، زمان افزودن BallFusion.cpp به پروژه به منظور نوشتن کد رسیده است. بدین منظور، از منوی فایل، گزینه New را انتخاب کنید. در حالیکه بخش C++ Source File در قسمت Files انتخاب شده است، کلمه BallFusion را بعنوان نام فایل وارد کرده و کلید OK را فشار دهید. بدین ترتیب ما چهارچوب مورد نظرمان را مهیا کرده ایم.

فایل BallFusion.cpp را برای اضافه کردن تکه کد زیر باز کنید:

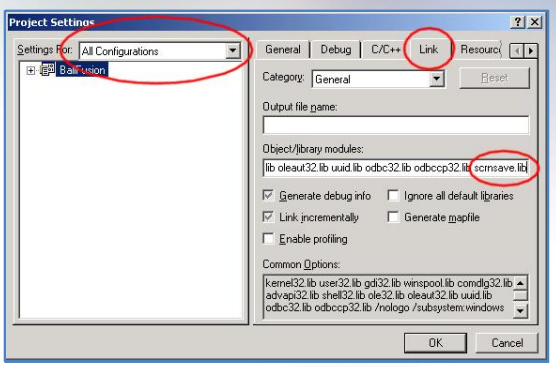
```
#include "windows.h"
#include "scrnsave.h"
```

```
LRESULT WINAPI ScreenSaverProc(HWND
hWnd,
     UINT message, WPARAM wParam,
LPARAM lParam)
{
    return 0;
}
```

اضافه نمودن دو تابع دیگر نیز مورد درخواست کتابخانه محافظ است: ScreenSaverConfigureDialog و RegisterDialogClasses. تابع اول پیام های ارسالی به پنجره پیکربندی محافظ را دریافت نموده و تابع دوم به منظور ثبت "کلاس های پنجره ای" مورد نیاز پنجره پیکربندی به کار برده می شود. از آنجایی که محافظ ما فاقد پنجره پیکربندی است، می توانیم به راحتی پیاده سازی این توابع را نادیده بگیریم. اما با این حال، باید آنها را در برنامه بیاوریم:

```
BOOL WINAPI
ScreenSaverConfigureDialog(HWND hDlg,
           UINT message, WPARAM
wParam, LPARAM lParam)
{
    //We simply return FALSE to
      indicate that
    //we do not use the configuration
      dialog box

    return FALSE;
}

BOOL WINAPI
RegisterDialogClasses(HANDLE hInst)
{
    //Since we do not register any
      special window class
    //for the configuration dialog
      box, we must return TRUE

    return TRUE;
}
```

قدم بعد، افزودن کتابخانه scrnsave.lib به ماژول های کتابخانه ای است که می خواهیم از آن ها در برنامه خود استفاده کنیم. برای این منظور، از منوی Project، گزینه Settings را انتخاب کرده و طبق شکل زیر تغییرات لازم را اعمال نمایید:




توضیحات

**پس از اعمال تغییرات مورد نیاز در پنجره اختیارات کامپایلر، زمان آن فرا رسیده است که بر روی مساله اصلی متمرکز شویم. من، چند متغیر سراسری در برنامه تعریف کرده ام:**

```
#define MAX_BALLS 7
#define pi 3.141592625

#define szAppName "BallFusion 1.0"
#define szAuthor "Written by Mehdi Mousavi, © 2001"
#define szPreview "Ball Fusion 1.0"

typedef struct _BALLS
{
    UINT uBallID;
    HICON hIcon;
    int x;
    int y;
    int angle;
}BALLS;

BALLS gBalls[] = {IDI_REDBALL, NULL, 0, 0, 0,
          IDI_GREENBALL, NULL, 0, 0, 25,
          IDI_BLUEBALL, NULL, 0, 0, 50,
          IDI_PURPLEBALL, NULL, 0, 0, 75,
          IDI_LIGHTREDBALL, NULL, 0, 0, 100,
          IDI_YELLOWBALL, NULL, 0, 0, 125,
          IDI_BLACKBALL, NULL, 0, 0, 150};
```

که در آن MAX_BALLS حداکثر تعداد توپ ها است، szAppName نام برنامه است که همراه نام مولف (szAuthor) در قسمت پایین و چپ صفحه نمایش داده می شود، szPreview رشته ای است که هنگام پیش نمایش محافظ در Control Panel به کاربر نشان داده شده و BALLS ساختاری است که مشخصات توپ در آن نگهداری می شود. این مشخصات حاوی شناسه آیکون (uBallID)، هندل به آیکون پس از بارگزاری (hIcon)، مختصات x و y توپ و زاویه (angle) شروع حرکت توپ است. در انتها به gBalls میرسیم، که آرایه ای از ساختار فوق الذکر می باشد.

هنگامی که تابع محافظ پیام WM_CREATE را دریافت می کند، برنامه، آیکون توپ ها را بر اساس شناسه هر یک به ترتیبی که در متغیر gBalls آورده شده است بارگزاری کرده و این آیکون های بارگزاری شده را در متغیر hIcon ساختار BALL نگهداری می کند. لطفا توجه کنید که برای بارگزاری آیکونها، handle به instance فعلی برنامه (hMainInstance) به تابع LoadImage داده می شود:

```
for(i = 0; i < MAX_BALLS; i++)
    gBalls[i].hIcon =
(HICON)LoadImage(hMainInstance,

MAKEINTRESOURCE(gBalls[i].uBallID),
                    IMAGE_ICON,
                    48,
                    48,
                    LR_DEFAULTSIZE);
```

سپس، نقطه شروع هر توپ (اعضای x و y ساختار BALL) بازای هر توپ بر اساس فرمول ذیل محاسبه می شود:

```
x = sin(ß) * cos(ß) * cos(2ß)
y = cos(ß)
```

**به بیان دیگر:**

```
xpos = GetSystemMetrics(SM_CXSCREEN) / 2;
ypos = GetSystemMetrics(SM_CYSCREEN) / 2;

for(i = 0; i < MAX_BALLS; i++)
{
    double alpha = gBalls[i].angle * pi / 180;
    gBalls[i].x = xpos +
      int((xpos - 30) * sin(alpha) * cos(alpha) * cos(2 * alpha));
    gBalls[i].y = ypos - 30 + int(265 * cos(alpha));
}
```

همچنین یک قلم سیاه رنگ هنگام قرار دادن آیکون ها بر روی صفحه ایجاد شده و یک تایمر نیز آغاز به کار می کند:

```
hBrush = CreateSolidBrush(RGB(0, 0,
0));
uTimer = SetTimer(hWnd, 1, 1, NULL);
```

هنگامی که تایمر شروع به کار کرد، تابع محافظ صفحه پیام WM_TIMER را دریافت می کند. برای این پیام، تمام کاری که باید انجام دهیم محاسبه x و y هر توپ است در حالیکه متغیر زاویه توپ (angle) را نیز افزایش می دهیم (0<=angle<=360). سپس ناحیه مربعی شکلی که توپ در آن به نمایش در خواهد آمد محاسبه و در متغیر rc نگهداری می شود.

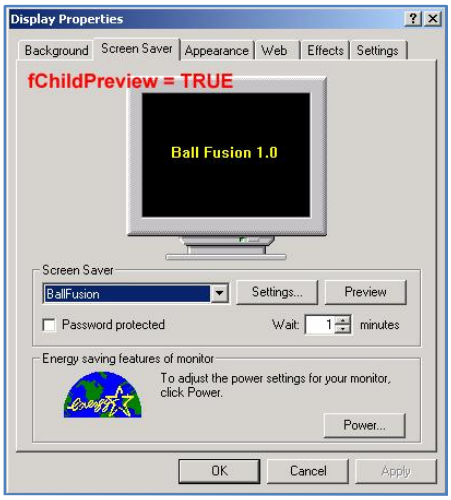
```
for(i = 0; i < MAX_BALLS; i++)
{
    double alpha = gBalls[i].angle * pi / 180;
    gBalls[i].x = xpos +
      int((xpos - 30) * sin(alpha) * cos(alpha) * cos(2 * alpha));
    gBalls[i].y = ypos - 30 + int(265 * cos(alpha));
    gBalls[i].angle = (gBalls[i].angle >= 360) ? 0 : gBalls[i].angle + 1;

    rc.left = gBalls[i].x;
    rc.right = gBalls[i].x + 48;
    rc.top = gBalls[i].y;
    rc.bottom = gBalls[i].y + 48;

    InvalidateRect(hWnd, &rc, FALSE);
}
```

هنگامی که تابع Invalidate فراخوانی شود، برنامه پیام WM_PAINT را دریافت می کند. در این شرایط کافیست تا به محض دریافت پیام، هر توپ را در جایگاه خودش که با x و y مشخص شده است، قرار دهیم:

```
if(fChildPreview)
{
    SetBkColor(hDC, RGB(0, 0, 0));
    SetTextColor(hDC, RGB(255, 255, 0));
    TextOut(hDC, 25, 45, szPreview, strlen(szPreview));
}
else
{
    SetBkColor(hDC, RGB(0, 0, 0));
    SetTextColor(hDC, RGB(120, 120, 120));
    TextOut(hDC, 0, ypos * 2 - 40, szAppName, strlen(szAppName));
    TextOut(hDC, 0, ypos * 2 - 25, szAuthor, strlen(szAuthor));

    for(i = 0; i < MAX_BALLS; i++)
```

```
        DrawIconEx(hDC,
        gBalls[i].x,
        gBalls[i].y,
        gBalls[i].hIcon,
        48,
        48,
        0,
        (HBRUSH)hBrush,
        DI_IMAGE);
}
```

و fChildPreview چیست؟ همانطور که پیشتر گفتم، این متغیر بیانگر آن است که آیا محافظ در وضعیت پیش نمایش در حال اجراست یا خیر. به بیان دیگر، اگر کاربر محافظ را در پنل کنترلی مربوطه مشاهده کند، fChildPreview برابر true و در غیر اینصورت، false خواهد بود.




## کلام آخر

کامپایلر برای این پروژه یک فایل اجرایی (.exe) تولید می کند. اما شما باید پسوند آن را به .scr تغییر دهید تا پنل کنترلی محافظ بتواند آنرا در ComboBox مربوطه بارگزاری نماید (همانظور که در شکل فوق نمایش داده شده است). نکته مهم دیگری که باید به خاطر داشت این است که آیکونی با شناسه ID_APP در برنامه باید قرار بگیرد. این، آیکون محافظ خواهد بود زیرا کتابخانه مربوطه این شناسه را به عنوان شناسه برنامه هنگام ثبت window class معرفی نموده است:

```
cls.hIcon = LoadIcon(hInst,
MAKEINTATOM(ID_APP));
```

## توسعه بیشتر

من ایده ای برای توسعه بیشتر این محافظ دارم و آن عبارت است از نگهداری تصویر فعلی پس زمینه بعنوان پس زمینه محافظ. برای آنکه متوجه منظورم شوید، پیام WM_ERASEBKGND را به window procedure خود اضافه کنید و برنامه را مجددا کامپایل کنید. البته، این توسعه میسر خواهد بود اگر و تنها اگر بتوانم رئیسم را متقاعد سازم. هرگونه پیشنهاد، سوال یا انتقاد را با کمال میل پذیرا هستم.

## پروانه

این مقاله همراه با کلیه کد های ضمیمه، تحت پروانه CPOL عرضه شده است.

درباره مولف :مهدی موسوی طراح حرفه ای نرم افزار است. او ۱۸ سال است که نرم افزارهای زیادی مبتنی بر تکنولوژی های مایکروسافت برای شرکت های بزرگ داخلی و خارجی طراحی کرده و هم اکنون در سمت مدیر پروژه یک شرکت کانادایی مشغول به کار است. برای تماس با وی می توانید به آدرس http://mehdi.biz مراجعه نمایید

# Interfaces (رابط ها) در Delphi

**نویسنده:** علی کشاورز

در این مقاله تعریف مختصری از مفهوم Interface و Abstract Class در طراحی نرم افزار ارائه می کنیم، و سپس به قابلیت های نوع داده Interface در دلفی و برخی کاربردهای آن، با ذکر مثال، می پردازیم.

## مقدمه

نوع داده Interface از نسخه ۳ به دلفی اضافه شد تا پشتیبانی از تکنولوژی COM شرکت مایکروسافت در دلفی فراهم شود. Interface علاوه بر پشتیبانی از COM، قابلیت های جالبی برای برنامه نویسان دلفی به ارمغان آورد که کمتر از آن استفاده می شود. متاسفانه به دلیل آنکه چارچوب اصلی VCL و بخصوص کامپوننت های کار با بانک اطلاعاتی دلفی مدت ها قبل از آن (از نسخه اول دلفی) ایجاد شده بودند، استفاده چندانی از Interfaceها در VCL صورت نگرفت.

برای درک بهتر interface ها، و قبل از آنکه به تعریف Interface و کاربردهای آن بپردازیم، ابتدا به مفهوم کلاس های Abstract و تعریف آنها اشاره ایی خواهیم داشت؛ زیرا مشابهت هایی در کاربرد این دو مفهوم وجود دارد، و شناخت مفهوم کلاس های Abstract درک مفهوم Interfaceها را ساده تر می کند.

## کلاس های Abstract

در دلفی، کلاسی که حداقل یکی از متدهای آن با استفاده از رهنمود abstract تعریف شده باشد، کلاس Abstract نامیده می شود.[۲۹]

متدهای abstract متدهایی هستند که توسط یک کلاس والد (parent) تعریف می شوند، اما پیاده سازی نمی شوند. پیاده سازی اینگونه متدها بر عهده کلاس های فرزند (child) می باشد.

در صورتی که یک نمونه (instance) [ یک شی (object) ] از یک کلاس abstract ایجاد کنید، با هشدار کامپایل مواجه خواهید شد و در صورتی که یکی از متدهای abstract آن شی را فراخوانی کنید، برنامه شما با یک استثنای "EAbstractError" متوقف خواهد شد.در اینجا این سوال ممکن است مطرح شود که اصلا کلاس یا متدی که هیچ پیاده سازیی ندارد و نمی توان از آن یک نمونه شی ایجاد کرد، چه استفاده ایی ممکن است داشته باشد؟

کلاس های Abstract در ساختار یک نرم افزار موجب می شوند که برنامه نویسی که پیاده سازی کلاس های فرزند را برعهده دارد، ملزم به پیاده سازی یکسری رفتارهای معین در هر یک از کلاس های خود شود؛ در غیر این صورت، با هشدارهای کامپایلر مواجه می شود. به این ترتیب احتمال اشتباه برنامه نویس (بخصوص در پروژه های بزرگ) کاهش می یابد.

مثال :

فرض کنید قصد داریم دو نوع پنجره طراحی کنیم (پنجره بیضوی و پنجره مستطیلی). هر دو این پنجره رفتارهای استانداردی مثل Minimize, Restore و غیره دارند. هر دو این پنجره ها باید ترسیم شوند، اما نحوه ترسیم هر یک از آنها فرق می کند. برای طراحی آنها می توانیم رفتارهای مشترک آنها را در کلاس TBaseWindow قرار دهیم و کلاس های دیگر از همین کلاس پایه، مشتق شوند. برای تعریف کلاس TBaseWindow دو روش می توانیم بکار بریم.

---

[۲۹] در دلفی ۲۰۰۶ و نسخه های بعد از آن می توان با اضافه کردن رهنمود abstract به تعریف کلاس، کل متدهای آن را بصورت abstract تعریف کرد، و لزومی ندارد عبارت abstract را به تعریف تک تک متدها اضافه کنیم:

```
TPureAbstract = class abstract
```

روش اول:

```
TBaseWindow = class
private
  AField  : TSomeType;
public
  procedure Draw; virtual;
  procedure Minimize;
  procedure Maximize;
  procedure Restore;
end;
```

متد Draw در کلاس پایه به صورت virtual تعریف شده است، در نتیجه کلاس های TOvalWindow و TRectangularWindow (دو کلاس فرضی برای رسم پنجره بیضوی و مستطیلی، مشتق شده از کلاس پایه TBaseWindow) می توانند این متد را override کرده و پیاده سازی خود را اعمال کنند. اگر برنامه نویسان این کلاس ها فراموش کنند که متد Draw را Override کنند، تا زمان اجرای برنامه مشکلی بوجود نخواهد آمد؛ اما در زمان اجرا با فراخوانی متد Draw پنجره ایی رسم نخواهد شد!

روش دوم:

```
TBaseWindow = class
private
  AField  : TSomeType;
public
  procedure Draw; virtual; abstract;
  procedure Minimize;
  procedure Maximize;
  procedure Restore;
end;
```

متد Draw در کلاس پایه بصورت abstract تعریف شده است. اگر برنامه نویسی که کلاس های TOvalWindow و TRectangularWindow را ایجاد می کند، فراموش کند که متد Draw را پیاده سازی کند؛ با فراخوانی متد Draw هشدار زیر را دریافت می کند:

```
OvalWindow.Draw;
```

*Constructing instance of TOvalWindow contains abstract method TBaseWindow.Draw*

Interfaces

اینترفیس ها نوعی قرارداد بین دو یا چند کلاس محسوب می شوند که در آن یک کلاس پیاده سازی **تمامی** متدهای مشخص شده در قرار داد را برای استفاده سایر کلاس ها یا کاربر تضمین می کند. پیاده سازی این متدها از دید کلاس های استفاده کننده و کاربر پنهان است. اینترفیس ها شبیه به کلاسی هستند که تمامی متد های آن abstract باشد و هیچ فیلدی برای ذخیره اطلاعات نداشته باشد.

اما اینترفیس ها ویژگی هایی دارند که آنها را از کلاس های Abstract جدا می کند:

اینترفیس ها کلاس نیستند، بلکه یک نوع (Type) مستقل در زبان مربوطه (در اینجا Object Pascal) محسوب می شوند.

اینترفیس ها از TObject مشتق نمی شوند. IUnknown در نسخه های قبلی دلفی (قبل از دلفی ۵) بعنوان والد تمام اینترفیس ها محسوب می شد، اما از نسخه ۵ به بعد، همه اینترفیس ها از IInterface مشتق می شوند. IUnknown و IInterface یکسان هستند. از IUnknown در برنامه های مبتنی بر تکنولوژی COM مایکروسافت استفاده می شود.

اگر پیدا سازی نکردن یک متد از کلاس abstract ،در کلاس های فرزند، موجب ایجاد هشدار کامپایلر می شد؛ پیاده سازی نکردن هر یک از متدهای یک اینترفیس، توسط کلاسی که آن اینترفیس را پشتیبانی می کند، موجب کامپایل نشدن برنامه می شود.

یک کلاس می تواند از چندین اینترفیس بطور همزمان پشتیبانی کند.

اینترفیس ها در دلفی قابلیت Reference-counting (شمارش مرجع) دارند.

*نکته: از اینترفیس ها هم مثل کلاس های abstract نمی توان یک نمونه (شی) ایجاد کرد.*

اینترفیس ها این امکان را به شما می دهند که خارج از ساختار سلسله مراتبی کلاس ها، قابلیت هایی را به برخی از کلاس های خود اضافه کنید. برای مثال فرض کنید شما کلاسی بنام TCar (برای تعریف یک اتومبیل) دارید که کلاس هایی مثل TSUV, TTruck, TSedan, TVan و غیره (برای تعریف انواع اتومبیل) از آن مشتق شده اند. اگر شما قصد اضافه کردن قابلیت 4WD (تقسیم نیروی محرکه بر روی هر چهار چرخ) به ماشین های SUV و Van داشته باشید، نمی توانید این قابلیت را در کلاس TCar پیاده سازی کنید، چون در این صورت تمامی ماشین ها از آن بهره خواهند برد. البته می توان این قابلیت را در هر یک ار این کلاس ها بصورت مستقل تعریف و پیاده سازی کرد، اما در صورت استفاده از اینترفیس (در اینجا I4wd)، هم می توانید از یک واسط مشترک استفاده کنید و هم از قابلیت های Polymorphic (چند ریختی) اینترفیس ها بهره ببرید:

```
I4wd = interface
  ['{C2816773-A5DF-4136-AC86-27E6231DB4A9}']
  procedure Initialize4WD;
end;

TCar = class(TInterfacedObject)
public
  procedure SomeCommonBehavior;
  procedure StartEngine; virtual;
end;

TSedan = class(TCar)
public
  procedure SomeSpecificBehavior;
  procedure StartEngine; override;
end;

TSUV = class(TCar, I4wd)
public
  procedure SomeSpecificBehavior;
  procedure StartEngine; override;
  procedure Initialize4WD;
end;

TVan = class(TCar, I4wd)
public
  procedure SomeSpecificBehavior;
  procedure Initialize4WD;
end;

{نحوه استفاده}

procedure Drive4WD(A4WDCar : I4WD);
begin
  if A4WDCar <> nil then
    A4WDCar.Initialize4WD;
    {Add Code for driving the car here}
end;
```

با استفاده از رویه Drive4WD شما می توانید هر ماشینی را که از اینترفیس I4WD پشتیبانی می کند، برانید. البته هر ماشین متد Ini alize4WD مربوط به خود را فراخوانی می کند. برای مثال:

```
Drive4WD(TVan.Create);
```

نگران آزاد کردن حافظه اشغال شده توسط TVan.Create نباشید، شی ایجاد شده بصورت خودکار آزاد خواهد شد.

### Globally Unique Identifier (GUID)

عبارت {C2816773-A5DF-4136-AC86-27E6231DB4A9} در تعریف اینترفیس بالا، یک GUID نامیده می شود. GUID عددی ۱۲۸ بیتی است (در مبنای ۱۶) که با تقریب بالایی منحصر به فرد می باشد [تعداد کل GUID هایی که می توان ساخت ۲ به توان ۱۲۸ است و عملا احتمال اینکه دو GUID یکسان تولید شود بسیار کم است.]

استفاده از GUID در تعریف یک اینترفیس موجب منحصر به فرد شدن آن اینترفیس می شود، به عبارت دیگر، حتی اگر چند اینترفیس در یک سیستم با نام مشابه وجود داشته باشند، مشکلی در استفاده از آنها بوجود نمی آید. استفاده از GUID در یک اینترفیس اجباری نیست، اما وجود آن برای استفاده از عملگر as، متد GetInterface، تابع Supports و متد QueryInterface (بطور کلی Interface Querying) الزامی است. اشیاء COM، اینترفیس ها و Type Library آنها نیز باید هر کدام دارای GUID باشند.

برای ساخت GUID در Editor دلفی می توانید از ترکیب سه کلید Ctrl+Shift+G استفاده کنید تا دلفی برای شما یک GUID ایجاد کند.

نکته: هیچگاه GUID یک اینترفیس را در یک اینترفیس دیگر استفاده نکنید.

## استخراج یک Interface از داخل یک Class

فرد متاهلی را در نظر بگرید که ریس یک شرکت می باشد. این فرد در رابطه با کارمندان خود نقش ریس، در رابطه با همسر خود نقش شوهر و در رابطه با فرزندان خود نقش پدر را ایفا می کند. در واقع این فرد در رابطه با هر گروه از افراد مذکور طبق یک رسم یا توافق مشترک، عمل می کند. هر یک از این گروه ها نیز بر اساس همان توافق یا رسم با این فرد ارتباط برقرار می کند و فقط به نقش مربوط به خود نیاز دارد (مثلا یک کارمند نیازی به دانستن رابطه ریس خود با همسر و فرزندانش ندارد). در دنیای نرم افزار هم یک کلاس می تواند با پیاده سازی اینترفیس های مختلف، نقش های مختلفی را در ارتباط با دنیای خارج بر عهده بگیرد.

حال، فرد فوق را بعنوان یک کلاس و توافق او با هر یک از گروه های فوق را به عنوان یک اینترفیس در نظر می گیریم:

هر گروه (**درخواست کننده؛ هر شی در دنیای خارج از آن کلاس**) باید ابتدا بررسی کند که آیا فرد (**کلاس پیاده سازی کننده**) مذکور به رسومات و توافق مشخص شده (**اینترفیس**) پایبند هست یا نه، اگر آن کلاس از اینترفیس مشخص شده پشتیبانی کند، درخواست کننده می تواند، بر اساس همان اینترفیس، درخواست خود را به آن کلاس ارائه دهد و جواب دریافت کند (**استخراج یک اینترفیس از کلاسی که آن را پیاده سازی می کند**). در دلفی برای بررسی پشتیبانی کردن یک کلاس از یک اینترفیس خاص و استخراج آن اینترفیس می توان از تابع Supports در یونیت SysUtils، متد QueryInterface از هر کلاسی که IInterface را پشتیبانی می کند (مثل TInterfaceObject) ، عملگر as، یا متد GetInterface در کلاس TObject استفاده کرد.[۳۰] به روش های مختلف استخراج اینترفیس I4WD از کلاس TSUV در مثال زیر توجه کنید:

[۳۰] در واقع تمامی روش های ذکر شده در داخل خود به نوعی از متد GetInterface استفاده می کنند.

```
var
  SUV : TSUV;
  A4wdInterface : I4WD;
begin
  {If TSuv implements I4WD (supports it), extract the interface.}
  if Supports(TSUV.Create,I4WD,A4wdInterface) then
    {Invoke Initialize4WD from returned interface.}
    A4wdInterface.Initialize4WD;

//----------------------------------
  {Just check if TSuv supports I4WD, no interface is returned.}
  if Supports(TSuv,I4WD) then
    ShowMessage('TSuv supports I4WD interface');

//----------------------------------
  SUV := TSUV.Create;
  {If you are sure than SUV supports I4WD, use as operator to extract
   the interface and invoke Initialize4WD.
   If SUV does not support I4WD, an exception will be thrown.}
  (SUV as I4wd).Initialize4WD;

//----------------------------------
  SUV := TSUV.Create;
  {If Suv implements I4WD (supports it), extract the interface.}
  if SUV.GetInterface(I4WD,A4wdInterface) then
    A4wdInterface.Initialize4WD;

//----------------------------------
  {TSUV supports I4WD, so you can assign an instance of it to a
   variable of type I4WD.}
  A4wdInterface := TSuv.Create;
  A4wdInterface.Initialize4WD;
end;
```

* تمامی توابع و عملگرهای فوق در Help دلفی توضیح داده شده اند.

### Multiple Inheritance (وراثت چندگانه)

در زبانهایی مثل دلفی، C#، و جاوا وراثت چندگانه امکان پذیر نیست - یعنی یک کلاس نمی تواند مثل C++ از چند کلاس والد مشتق شود. اما در این زبان ها امکان پیاده سازی چند اینترفیس توسط یک کلاس وجود دارد و می تواند به نوعی این خلاء را پر کند. مثلا کلاس TMyMobile (یک کلاس فرضی مربوط به گوشی موبایل) می تواند بطور همزمان اینترفیس های IInfraRed و IBlueTooth را پیاده سازی کند:

```
TMyMobile = class(TInterfaceObject, IInfraRed, IBlueTooth)
  {TMobile's fields and methods + IInfraRed's methods + IBlueThoot's methods}
end;
```

### Reference counting (شمارش مرجع)

یکی از ویژگی های مهم اینترفیس ها در دلفی (همچنین در COM) خصوصیت Reference Counting آنها ست. با استفاده از این قابلیت اشیاء ایجاد شده بصورت خودکار مدیریت می شوند و در صورتی که نیازی به آنها نباشد، بطور خودکار آزاد می شوند. متدهای مربوط به این قابلیت در اینترفیس IInterface تعریف شده اند.

هر بار که اینترفیسی از یک کلاس استخراج می شود یا نمونه ایی از یک کلاس به متغیری از یک اینترفیس اختصاص داده می شود، دلفی متد _AddRef از اینترفیس IInterface را

فراخوانی می کند. هر زمان که آن متغیر نا معتبر شود (با اختصاص مقدار nil به آن یا خارج شدن از محدوده تعریف آن متغیر)، دلفی متد Release_ را فراخوانی می کند. شما به عنوان برنامه نویس وظیفه دارید که متغیری در کلاس خود تعریف کرده (برای ذخیره شماره مرجع) و این متد ها را پیاده سازی کنید. با اجرای AddRef_ باید یک واحد به شماره مرجع اضافه شود. با اجرای Release_ باید یک واحد از شماره مرجع کم شود، اگر شماره مرجع به صفر رسید، Release_ وظیفه آزاد کردن شی ایجاد شده را بر عهده دارد.

در صورتی که تمایلی به پیاده سازی متدهای فوق ندارید، می توانید کلاس خود را از کلاس TInterfacedObject مشتق بگیرید. در این صورت کلاس TInterfacedObject قابلیت شمارش مرجع را برای کلاس شما پیاده سازی خواهد کرد. TInterfacedObject در یونیت System تعریف شده است. بهتر است کد آن را مطالعه نمایید.

به مثال های زیر در رابطه با کاربرد Reference Counting توجه کنید:

```
procedure TForm1.Button3Click(Sender: TObject);
var
  SUV : TSUV;
  Van : TVan;
  AnInterface,
  AnotherInterface : I4WD;
begin
  Memo1.Clear;
  SUV := TSuv.Create;
  //SUV.RefCount = 0
  AnInterface := SUV;
  //SUV.RefCount = 1
  AnotherInterface := SUV;
  //SUV.RefCount = 2

  Van := TVan.Create;
  //Van.RefCount = 0;
  AnInterface := Van;
  //SUV.RefCount = 1
  //Van.RefCount = 1
end;
//" AnInterface" is out of scope: SUV.RefCount = 0
//SUV is destroyed
//AnotherInterface is out of scope: Van.RefCount = 0
//Van is destroyed
```

در مثال بالا به تغییرات RefCount توجه کنید:

زمانی که SUV به یک اینترفیس اختصاص داده می شود، یک واحد به شمارشگر مرجع آن افزوده می شود.

با اختصاص SUV به اینترفیس بعدی (AnotherInterface)، شمارشگر مرجع آن به ۲ می رسد.

با اختصاص Van به اینترفیس AnInterface ، شمارشگر مرجع این شی یک واحد افزایش پیدا می کند، ولی شمارشگر مرجع شی SUV که قبلا به AnInterface اختصاص داده شده بود، یک واحد کاهش پیدا می کند.

در پایان این متد، هیچ کدی برای آزاد سازی متغیرهای SUV و Van وجود ندارد؛ اما با پایان یافتن کد Form1.Bu on3Click، متغیرهای AnInterface و AnotherInterface که بصورت محلی تعریف شده اند، نامعتبر می شوند و با نامعتبر شدن هر یک از آنها، یک واحد از شمارشگر مرجع هر یک از اشیاء اختصاص یافته به آنها کاهش میابد. با صفر شدن شمارشگر مرجع هر شی، آن شی آزاد می شود.Malcome Groves در مقاله Interfaces: Off the Beaten Track مثالی از پیاده سازی قابلیت Garbage Collection با استفاده از اینترفیس ارائه کرده است[۳۱]:

```
unit mtReaper;
```

[۳۱] Delphi Magazine, شماره ۸۰

```
interface

type
  ImtReaper = interface
  ['{1B321324-975F-4026-B742-E7B3AD486BB5}']
  end;

  TmtReaper = class(TInterfacedObject, ImtReaper)
  private
    FObject : TObject;
  public
    constructor Create(AObject : TObject);
    destructor Destroy; override;
  end;

implementation

uses SysUtils;

constructor TmtReaper.Create(AObject: TObject);
begin
  FObject := AObject;
end;

destructor TmtReaper.Destroy;
begin
  FreeAndNil(FObject);
  inherited;
end;

end.
```

در این کد کلاس ساده ایی (TmtReaper) تعریف شده که Garbage Collection را برای شما انجام می دهد. با ارسال یک شی به متد Create این کلاس، شی مورد نظر در زمان مقتضی آزاد خواهد شد و نیازی نیست که شما شخصا آن شی را آزاد کنید:

```
var
  List : TStringList;
  mtReaper : ImtReaper;
begin
  List := TStringList.Create;
  mtReaper := TmtReaper.Create(List);

  //Use List in your code
  List.LoadFromFile('A File Name');
  List.Sort;
end;
//List will be destroyed automatically by mtReaper
```

در دلفی ۲۰۰۹ می توان این قابلیت را با استفاده از مفهوم Smart Pointers به شکل مناسب تری پیاده سازی کرد.

***

# آشنایی با زبان برنامه نویسی F#

نویسنده: مهدی عسگری

سطح : متوسط ، پیشرفته

چکیده: F# یکی از جدیدترین زبان هایی است که توسط شرکت مایکروسافت ارائه شده است. چرا یک زبان دیگر ؟ آیا زبان های برنامه نویسی موجود کافی نیستند ؟ در این مقاله علاوه بر معرفی اجمالی زبان F# ، نگاهی هم به کاربردهای ان و تفاوت ها و شباهت هایش با زبان های موجود می اندازم.

از روزی که جان مک کارتی در سال ۱۹۵۸ زبان LISP را ایجاد کرد ، تا همین چند سال پیش شاهد پیشرفت های خیلی بزرگ و بنیادی در زبان های برنامه نویسی نبودیم. یعنی امروزه هم ما برای برنامه نویسی کد را در فایلی می نویسیم و تحویل کامپایلر می دهیم (و معمولا واحد جداکننده ، خط است) و هنوز از مفاهیمی که زبان LISP و زبان های دهه ی ۶۰ میلادی معرفی کردند استفاده می کنیم (ارایه ، اشاره گر متغیر ، حلقه و ...) و بیشترین تغییرات را در ابزار (IDE ، دیباگر ، پروفایلر و ...) شاهد بودیم. (حتی مفاهیمی مثل GC هم از نسخه های اولیه ی لیسپ موجود بودند) شیوه ی برنامه نویسی ای که غالب برنامه نویسان به ان عادت کرده و با آن کد می نویسند (و بیشتر برنامه نویسان قادر به تصور برنامه نویسی به شکلی دیگر نیستند) ، روش دستوری (imperative) است (و نیز شی گرا)

زبان های برنامه نویسی ای که می شناسیم (سی ، پاسکال جاوا ، ...) بر مبنای مدل تورینگ هستند[1]. منتها مدل دیگری هم برای زبان های برنامه نویسی وجود دارد که شاید کمتر به گوشمان خورده باشد: حساب لامبدا. آقای Alonzo Church (پدر لامبدا!) و Allen Turing همزمان به این دو مدل دست یافتند و بعدا هم ثابت کردند که این دو زبان از نظر قدرت محاسباتی با هم برابرند. زبان های مبتنی بر مدل تورینگ ، معمولا دستوری هستند. نقطه ی مقابل زبان های دستوری ، زبان های بیانی (declarative) است که در ان ها به جای تاکید بر چگونگی انجام یک عمل (how) روی این که چه کاری را می خواهیم انجام دهیم (what) تاکید می کنیم.

(مثل SQL که فقط با query ها می گوییم که چه می خواهیم ، ولی چگونگی انجامش را واگذار می کنیم به کامپایلر/مفسر)

زبان های تابعی هم به نوعی declarative محسوب می شوند (اکثر اوقات ما روی بخش what تاکید داریم تا how. در ادامه مثال خواهم آورد)

زبان های محبوب و رایج امروزی (سی پلاس پلاس ، جاوا دلفی ، سی شارپ ، روبی و ...) زبان های شی گرا هستند (مثال آخر بر خلاف بقیه یک زبان داینامیک است) (البته درست است که در سال های اخیر ویژگی های دیگری هم به این زبان ها اضافه شده که برنامه نویسی برای مدل های دیگر همچون تابعی را پشتیبانی می کند ، ولی باز هم هسته ی این زبان ها دستوری و شی گراست). زبان های تابعی به اندازه ی زبان های دستوری قدمت دارند (از LISP در سال ۱۹۵۹و Scheme در دهه ی ۷۰ و Haskell و ML در دهه ی ۸۰) منتها از عللی که باعث شده این زبان ها بیشتر در دنیای آکادمیک کاربرد و محبوبیت داشته باشن ، می شود موارد زیر را برشمرد:

- نداشتن کتابخانه های قدرتمند و وسیع
- نداشتن حمایت مالی شرکت های نرم افزاری
- عدم توانایی برقراری ارتباط با زبان های مهمی مثل سی برای ارتباط با سیستم عامل میزبان
- سرعت اجرای پایین
- شیب زیاد منحنی یادگیری
- و ....

البته تمام موارد فوق امروزه غلط هستند و مخصوصا در مورد سرعت اجرا زبان هایی مثل OCaml در موارد زیادی از سی

[1] http://en.wikipedia.org/wiki/Alan_Turing

هم سریع ترند (به دلیل نامرتبط بودن با این بحث ، محولش می کنم به یک مقاله ی دیگر ؛ فقط این نکته را اضافه کنم که علت سرعت پایین تر این زبان ها در گذشته محدودیت حافظه و قدرت پردازش کامپیوتر های ان زمان و نیز استفاده ی این زبان ها از GC و پویا بودن و یا انجام type checking های پیچیده (در مورد استاتیک ها) است که با توجه به سخت افزار های ضعیف ان روز ها سرعت اجرای پایین و حافظه ی مصرفی بالا داشتند.)

زبانی مثل Haskell کاملا شی گراست (pure functional) و برنامه نویسی به مدل های دیگر را پشتیبانی نمی کند. از دلایل موفقیت زبان OCaml این است که این زبان هم شی گراست و هم تابعی و به نظر من زبان هایی موفق ترند که چندین پارادایم را پشتیبانی کرده و دست برنامه نویس را باز می گذارند (حرکت اخیر مایکروسافت برای افزودن ویژگی های تابعی ( LINQ, anonymous methods, expression trees,...) و پویا به زبان سی شارپ موید این نظر است)

F# محصول تیم تحقیقاتی کمبریج مایکروسافت به سرپرستی Don Syme ، پورتی از زبان OCaml برای .NET framework است. این زبان شی گرا ، تابعی و استاتیک است. خود OCaml از Caml و ان هم از ML مشتق شده اند. خانواده ی ML شامل دو زبر شاخه ی SML و Caml است که موفق ترینشان تا به امروز OCaml بوده. زبان های خانواده ی ML و همچنین زبان Haskell به دلیل استفاده از Type inference[1] ، کد موجز و کوتاه تری در مقایسه با دیگر زبان های استاتیک دارند. (گرچه می شود برای خوانایی بیشتر یا کمک به ابزار مستندسازی یا ... ، نوع داده ها را هم در کد آورد. در نسخه ی ۳ زبان سی شارپ هم ویژگی استنباط نوع از طریق کلمه ی کلیدی var اضافه شد).

F# در اصل یک زبان استاتیک است ، منتها با استفاده از محیط تعاملی این زبان (F# Interactive) می شود مثل زبان های پایتون و روبی ، به شکل داینامیک کد نوشت. در ضمن این زبان بر اساس سینتکس و مفاهیم زبان OCaml پیاده سازی شده (و کتابخانه های OCaml را جز تعداد اندکی ساپورت نمی کند) ، منتها مدل شی (object model) اش همانند سی شارپ است (به خاطر این که زبانی تحت دات نت است) و همچنین ویژگی indentation یا تورفتگی کد های آن ، یاداور پایتون است. در ضمن از Haskell و Erlang هم ویژگی هایی به ارث برده.

آخرین نسخه ی این زبان نسخه ی CTP 1.9.6.2 است که از وبسایت رسمیF# قابل دانلود است. در نسخه ی بعدی Visual Studio (نسخه ی ۲۰۱۰) این زبان در کنار Visual C++, C# , VB جزو زبان های رسمی مایکروسافت خواهد بود. (مایکروسافت زبان های دیگری را هم به طور مستقیم و غیر مستقیم پشتیبانی می کند ؛ مثل SmallBasic ، Haskell OCaml و ... منتها زبان های رسمی مایکروسافت نیستند)

F# مزایای زیادی دارد که ان را از دیگر زبان های تابعی متمایز می کند. هم تابعیست هم شی گرا. زبانی استاتیک است ، منتها اجازه می دهد که مثل زبان های داینامیک با آن کار کنیم. سینتکسی دارد که نزدیک به ۳۰ سال است وجود دارد و نخواسته انقلابی ارائه کند. (به قول طراحش ، evolutionary است نه revolutionary). مشکل کتابخانه را حل کرده (استفاده از دات نت). حمایت شرکت مایکروسافت را دارد. هنوز که نسخه ی نهایی اش ارائه نشده ، ۳ کتاب داشته (و ۳ کتاب هم در راهند) و شرکت های زیادی دارند در محصولاتشان از آن استفاده می کنند. مثل Haskell کاملا lazy نیست ، منتها برای این کار کلمه ی کلیدی مخصوص دارد. مثل Erlang کاملا بر اساس مدل message-passing کار نمی کند منتها هم مدل shared memory و هم مدل MP را پشتیبانی می کند. کد موجزی دارد که باعث می شود productivity برنامه نویس بالا رفته و درصد سیگنال به نویز (نقطه ویرگول ، آکولاد ، بلاک ها ، type identifier و ...) بالاتر رود (این رامدیون استفاده از Hindley–Milner[2] است) همانطور که خالق این زبان در مصاحبه هاش گفته ، قرار نیست این زبان جایگزین وی بی یا سی شارپ بشود. در واقع هنوز هم برای برنامه های تجاری[3] و GUI و برنامه های تحت وب و ... زبان های فعلی مناسب ترند (نه این که F# قادر به انجام این کارها نباشد)

## موارد کاربرد F# :

[1] استنباط نوع داده ی یک عبارت یا متغیر بر اساس نوع استفاده از ان

[2] http://en.wikipedia.org/wiki/Type_inference#Hindley.E2.80.93Milner_type_inference_algorithm

[3] Line Of Business

- برنامه هایی که قرار است به صورت موازی روی چند هسته (یا سی پی یو) اجرا شوند
- برنامه هایی که با حجم وسیعی از داده ها سر و کار دارند (data analysis)
- برنامه های محاسباتی و ریاضی و آماری و ...

و اما در مورد concurrency و عصر جدید (پردازنده های چند هسته ای) که به ما این اجازه را می دهد به صورت واقعی (نه شبیه سازی با thread ها) برنامه هایی بنویسیم که موازی اجرا شوند و اهمیت زبان های تابعی برای این کار (و اصلا این که چطور شد پس از ۵۰ سال توجه عمومی شرکت ها و برنامه نویسا و ... به این زبان ها جلب شده):

دو مدل غالب برای برنامه نویسی موازی داریم: مدل حافظه ی مشترک[۱] و ارسال پیام[۲]. در مدل حافظه ی مشترک (مدل غالب تر!) به خاطر دسترسی همزمان چند ریسمان/فرایند/پردازنده به قسمتی از حافظه ، مشکلات زیادی مثل race condition و deadlock و synchronization و ... به وجود می آید و همواره باید مراقب کدمان باشیم که همین امر ، برنامه نویسی در این مدل را سخت می کند. در زبان های تابعی به طور پیش فرض ما متغیر نداریم و همه چیز تغییرناپذیر (immutable) است، بنابراین تغییر مقدار بی معنی است. از طرفی در این زبان ها از ساختار هایی مثل حلقه (که تقدم و تاخر را به وجود می آورند) کمتر استفاده شده و مدل برنامه نویسی declarative است؛ بنابراین امکان موازی شدن به صورت خودکار[۳] (توسط کامپایلر و رانتایم. مثل Parallel LINQ[۴]) بیشتر شده و کدنویسی به این منظور هم راحت تر می شود. (در زبانی مثل C++ هم این کار قابل انجام است ، منتها زمان زیادی صرف کارهای دستی و مراقب کد بودن و ... می شود که در نهایت هم باگ هایی در برنامه وجود خواهد داشت که پیدا کردنشان خیلی سخت است). این مدل مثل SQL است. شما فقط کوئری را می نویسید و موازی سازی و ... به عهده ی SQL engine هست. ویژگی immutability بزرگ ترین وجه تمایز زبان های تابعی از دیگر زبان هاست. البته باز هم در اینجا F# با استفاده از کلمه ی کلیدی mutable اجازه می دهد که بتوانیم مقدار یک شی را عوض کنیم.

در زیر قطعه کدهایی از زبان F# را همراه با مشابهش در سی شارپ (یا سی پلاس پلاس) قرار می دهم: (هدف نشان دادن برتری زبانی به دیگری نیست ، فقط می خواهم مقایسه ای بین ظاهر و مدل برنامه نویسی صورت بگیرد )

محاسبه ی فاکتوریل یک عدد:

سی شارپ:

```
class Program
{
    public int factorial(int n)
    {
        if (n < 2)
            return 1;
        else
            return n * factorial(n -
1);
    }
}
```

F#:

```
let rec factorial n = if n < 2 then 1
else n * factorial (n-1)
```

[۱] Shared memory
[۲] Message passing
[۳] Automatic Parallelization
[۴] http://en.wikipedia.org/wiki/Parallel_FX_Library#PLINQ

Currying:

```
using System;
using System.Collections.Generic;
using System.Linq;

class Program
{
    delegate int myAddDelegate(int a, int b);
    delegate int mySecondDelegate(int a);

    static void Main()
    {
        myAddDelegate add = delegate(int a, int b) { return a + b; };
        int x = add(10, 20);
        mySecondDelegate add10 = delegate(int a) { return add(a, 10); };
        int y = add10(20);
    }
}
```

F#:

```
let add x y = x + y
let add10 x = add x 10
let x = add 10 20
let y = add10 20
```

ایجاد یک Windows Application ساده:

سی شارپ:

```
using System.Windows.Forms;

class MyForm : Form
{
    private Button btn;

    public MyForm()
    {
        btn = new Button();
        btn.Text = "Click me";
        btn.BackColor = System.Drawing.Color.Red;
        this.Controls.Add(btn);
        this.Text = "Hello";
        btn.Location = new System.Drawing.Point(50, 50);
        this.TopMost = true;
        btn.Click += delegate(object sender, System.EventArgs e) {
MessageBox.Show("Hello World"); };
    }

    static void Main()
    {
        Application.Run(new MyForm());
    }
}
```

F#:

```
#light

open System.Windows.Forms
let frm = new Form(Text = "Hello")
let btn = new Button(Text = "Click me", BackColor = System.Drawing.Color.Red)
frm.Controls.Add(btn)
btn.Location <- new System.Drawing.Point(50,50)
btn.Click.Add(fun args -> MessageBox.Show("Hello World") |> ignore)
frm.Show()
```

```
frm.TopMost <- true
```

تولید و چاپ اعداد بین ۱۰۰۰۰۰۰ تا ۱۰۰۲۰۰۰ به طور موازی (با استفاده از asynchronous workflow ها که از ویژگی های منحصر به فرد زبان F# هست – کلمه ی کلیدی async) (مثال از ویکی پدیا)

```
let is_prime n =
    {2 .. n-1} |> Seq.exists (fun x -> n % x = 0) |> not

let primeAsync n =
    async { return (n, is_prime n) }

let primes m n =
    {m .. n}
        |> Seq.map primeAsync
        |> Async.Parallel
        |> Async.Run
        |> Array.filter snd
        |> Array.map fst

primes 1000000 1002000
    |> Array.iter (printfn "%d")
```

نتیجه و نظر شخصی: به نظر من موفقیت از آن زبان هایی بوده و خواهد بود که پارادایم های مختلفی را پشتیبانی کرده و به برنامه نویس اجازه بدهند روی مفهوم و الگوریتم و حل مسئله کار کرده تا این که بخواهد خودش را درگیر "ریز بهینه سازی" ها و کمک به کامپایلرکند (در واقع باید روز به روز میزان کمک کامپیوتر و نرم افزار به انسان بیشتر شود تا بتوانیم از سطح بالاتر و متفاوتی به مسائل و نحوه ی حلشان نگاه کنیم. زبان F# از زبان هایی است که توانسته راه درست رو انتخاب کنه (یعنی گلچینی از ویژگی های خوب و محبوب زبان های پیشین) و به برنامه نویس اجازه می دهد از سطحی بالاتر به مسائل نگاه کرده و برنامه بنویسد که در نهایت منجر به کارایی بیشتر برنامه نویس و صرفه جویی در وقت (که مهم ترین و گران ترین منبع است) خواهد شد.)

## منابع و مطالعه بیشتر:

این مقاله را به صورت مستقیم از جایی ترجمه نکردم و حاصل تجربه و مطالعات پراکنده ام است.

وبسایت رسمی F# :

http://research.microsoft.com/fsharp/fsharp.aspx

فروم F# : http://cs.hubfs.net/forums

وبلاگ Don Syme :

http://blogs.msdn.com/dsyme

## مدخل ویکی پدیا برای این زبان:

http://en.wikipedia.org/wiki/F_Sharp_programming_language

## کتب منتشر شده برای این زبان:

Foundations of F# نوشته ی Robert Pickering انتشارات Apress

Expert F# نوشته ی Don Syme ، Adam Granicz و Antonio Cisternino انتشارات Apress

F# for Scientists نوشته ی Jon Harrop انتشارات Wiley

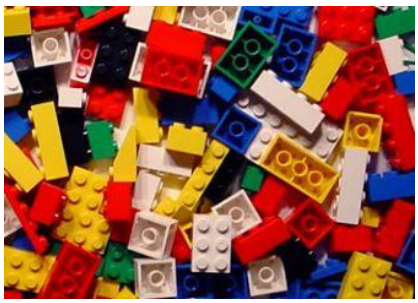
(کتاب های Programming F# ، Real World Functional Programming و F# in a Nutshell هم قرار است امسال عرضه شوند)

دربارهٔ نویسنده: مهندس نرم افزار. حدود ۶ ساله که برنامه نویسی می کنم و خارج از وقت کاری ام (که مربوط به امنیت نرم افزار و برنامه نویسی سیستمی میشه) به علایقم (یادگیری زبان های جدید ، زبان های تابعی ، پردازش موازی ، شنا و تماشای فیلم) می پردازم. در حال حاضر بیشترین وقتم صرف مطالعه در مورد Concurrency (Erlang, Parallel FX, Cilk++, ...) و زبان های تابعی (F#, Haskell, Scheme/LISP) میشه. آدرس الکترونیکی:

mehdi.asgari@yahoo.com

# آنچه توسعه دهندگان ASP.NET باید از فناوری SharePoint بدانند (قسمت اول)

نو سنده: حمدر ضا متقیان



سطح : مبتدی ، متوسط

**مقدمه:**

فناوری SharePoint اولین بار در سال ۲۰۰۱ میلادی توسط شرکت مایکروسافت به منظور فراهم آمدن بستری مناسب برای انجام تعاملات درون سازمانی ارائه شد. این فناوری در سال ۲۰۰۳ با امکانات گسترده تری وارد بازار پورتال های تجاری شد.

SharePoint یک پلتفرم قابل مدیریت و توسعه پذیر می باشد که با بکارگیری امکانات و قدرت Microsoft Office به تولید برنامه های تحتِ وب می پردازد.

MS Office برنامه ای قدرتمند در حوزه ی مدیریت مستندات و اطلاعات نا ساخت یافته می باشد. برنامه نویسانِ .NET نیز با استفاده از پلتفرم ASP.NET قادر به ایجاد برنامه های تحت وب هستند. در واقع مایکروسافت با پیوند دادن این دو برنامه، بستر جدیدی به نام SharePoint را به توسعه دهندگان وب معرفی کرد تا برنامه نویسانِ .NET بتوانند با اتکاء به دانشِ ASP.NET خود و استفاده از توانایی برنامه های MS Office مجموعه ای از برنامه های تحت وب را با استفاده از محصولات SharePoint (WSS 3.0 و MOSS 2007) تولید کنند.

لازم به ذکر است که جامعیت پلتفرم SharePoint بسیار فراتر از پیوند این دو برنامه هست ولی در کل، ایده ی شکل گیری این پلتفرم در بر آورده ساختن نیاز به پیوند دادنِ MS Office و ASP.NET در یک محیط عملیاتی- سازمانی بوجود آمد.

چیزی که در عموم اذهان متداول هست SharePoint بعنوان یک سیستم مدیریت محتوای صرف در پروژه های بزرگ قابل معرفی است. در این مقاله سعی شده است جوانب کامل این موضوع مورد بررسی قرار گیرد و در ادامه به مسائلی که یک برنامه نویس و یا توسعه دهنده ی .NET در مورد این فناوری باید بداند خواهیم پرداخت.

## محصولات و فناوری های SharePoint:

SharePoint مجموعه ای از امکانات تعاملات سازمانی، آنالیز داده های تجاری، سیستم مدیریت محتوا، سرویس های مردمی و شخصی سازی، جستجوی پیشرفته، پورتال ها و مدیریت فرآیندهای سازمانی می باشد.

SharePoint شامل سه محصول متفاوت زیر است:

### Windows SharePoint Services 3.0 یا WSS 3.0:

این محصول یک برنامه ی رایگان می باشد که به همراه ویندوز سرور 2003 و ۲۰۰۸ ارائه می گردد. شرکت مایکروسافت این برنامه را بعنوان یک استاندارد برای طراحی و توسعه نرم افزارهای تحت وب در محیط .NET معرفی می کند.

### Microso Office SharePoint Server 2007 یا MOSS 2007:

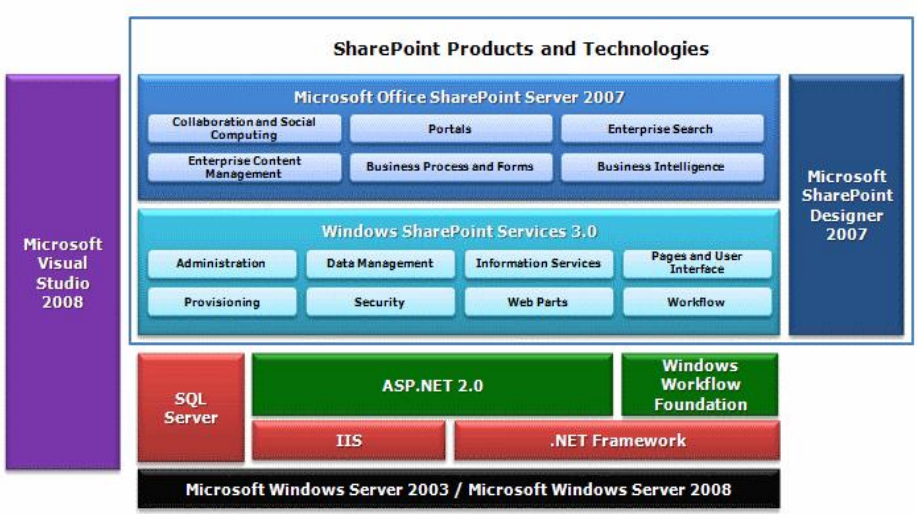
یک محصول تجاری است که مایکروسافت با استفاده از برنامه ی WSS 3.0 تولید کرده است. در واقع MS Content Management Server 2002 و MS SharePoint Portal Server 2003 در این محصول ادغام شده اند. همچنین فناوری های دیگری مانند Business

Intelligence و Business Processes که در ادامه در موردشان بحث خواهیم کرد نیز به آن اضافه شده اند.

این محصول تلاش مایکروسافت برای ورود به بازار تولید Enterprise Applications می باشد.

### Microso Office SharePoint Designer 2007 :

این محصول ابزاریست جهت طراحی و توسعه برنامه های مبتنی بر SharePoint. در واقع این امکان را به شما می دهد تا با سرعت بیشتری برنامه های مورد نیاز خود را طراحی کنید. همانطور که در شکل زیر آمده است WSS 3.0 و MOSS 2007 با استفاده از تکنولوژی ASP.NET 2.0 و .NET 3.0 Windows WorkFlow Foundation تولید شده اند.




بررسی محصول Windows SharePoint Services 3.0 :

WSS 3.0 فن آوری ای شامل مجموعه ای از ابزارها جهت تعاملات سازمانی است. این ابزارها امکان دسترسی آسان به مستندات، محتوا، افراد و اطلاعات را می دهد. WSS 3.0 در بردارنده ی ۸ قابلیت می باشد که با استفاده از این قابلیت ها توسعه دهندگان وب قادر به تولید برنامه های تحت وب حرفه ای و مقیاس پذیر هستند:

### Administration یا مدیریت جامع:

در بردارنده ی مجموعه ای از کلاس ها و خصوصیات است که در فضای نام Microsoft.SharePoint.Administration وجود دارند و قابلیت های مدیریت، اجتماع، توسعه و نگهداری را به سایت هایی که از SharePoint استفاده کرده اند می دهد. اطلاعات بیشتر

### Data Mangenment یا مدیریت داده ها:

WSS 3.0 محیط جامع و قدرتمندی را برای مدیریت تعامل و ذخیره سازی داده ها فراهم می کند که این موضوع شامل روش های مختلفی جهت مدیریت محتوا و فرا داده ای مستندات تجاری می باشد به طوری که انسجام و جامعیت داده ای بین مستندات موجود در WSS و MS Office به طور کامل حفظ می شود.

### Information Services یا خدمات اطلاعاتی:

این خدمات شامل دریافت پست الکترونیک، اعلان دریافت پست الکترونیک و جستجو می باشد.

### Pages and User Interface(UI) یا واسط کاربری:

از آنجایی که نمونه ی صفحات در WSS 3.0 بر پایه ی ASP.NET 2.0 ایجاد شده است شما امکان استفاده از یوزر کنترل ها (User Controls) و مسترپیج ها ( Master Pages) را در طراحی قالب صفحات خود دارید.

همچنین بصورت پویا WSS 3.0 این امکان را به شما می دهد تا بتوانید کنترل های Menu، TreeView و کلا Navigation ها را به سلیقه خود و متناسب با نیازی که دارید تغییر دهید. اطلاعات بیشتر

### Provisioning یا ایجاد خودکار صفحات:

در WSS 3.0 از آنجایی که صفحات قابلیت ایجاد به صورت Run-Time و در لحظه را دارا هستند الزاما نیازی به وجود یک صفحه ی ASPX فیزیکی نیست. WSS این امکان را می دهد که با توجه به نیاز و امکاناتی که شما برای ایجاد یک صفحه در نظر دارید فایل ها و وب پارت های مورد نیاز آن صفحه به صورت خودکار بارگذاری شوند و در طی این مراحل نیازی به تغییرات پایگاه داده ای وجود ندارد. اطلاعات بیشتر

### Security یا امنیت:

با توجه به جامعیت امنیتی موجود در ASP.NET و ویندوز سرور، WSS 3.0 نیز از این سطح امنیتی مستثناء نیست و مکانیزم های تصدیق هویت و اعتبارسنجی موجود در .NET. و امکانات امنیتی IIS در بالا رفتن امنیت برنامه ها ی تولید شده در WSS 3.0 تاثیر مستقیمی گذاشته است.

### Web Parts یا وب پارت ها:

WSS 3.0 بر پایه ی کنترل وب پارت ASP.NET بنا شده است. با استفاده از وب پارت ها شما قادر به ایجاد اجزاء رابط های مختلف کاربری هستید. مدیریت وب پارت ها با استفاده از SharePoint Designer 2007 نیز قابل انجام می باشد.

اطلاعات بیشتر

### Workflow یا گردش کار:

با استفاده از این ابزار شما قادر خواهید بود فرآیند های داخل سازمان که از آن جمله ایجاد اسناد و آیتم های اطلاعاتی می باشند را کنترل نموده و وضعیت پیشرفت کار را پیگیری نمایید.

WSS 3.0 و MOSS 2007 یک مدل مدیریت شده ای را فراهم کرده اند که به شما این امکان را میدهد تا از طریق کدنویسی بتوانید وب ساییتان را توسعه دهید. این مدل قابلیت یکپارچه شدن با ASP.NET را دارد و شما قادر خواهید بود تا به صورت پویا مستندات، محتوا و اطلاعات پروژه ی خود را سفارشی کنید. اطلاعات بیشتر

## بررسی محصول Microsoft Office SharePoint Server 2007 :

این محصول بر اساس WSS 3.0 بنا شده است و در بردارنده ی چارچوب کاری آشنا و سازگار برای کتابخانه های مختلف می باشد. همچنین امکانات زیر را در اختیار شما قرار می دهد:

امکان جستجوی افراد، اسناد و داده ها

قابلیت ایجاد فرم های مختلف جهت فرآیندهای سازمانی

امکان دسترسی و تحلیل مقادیر بزرگ داده های تجاری

مجموعه قابلیت های MOSS 2007 در اشکال زیر معرفی شده است:



همانطور که در شکل مشخص است MOSS 2007 شش قابلیت را جهت توسعه و سفارش کردن برنامه های تحت وب در بر دارد که به بررسی آنها خواهیم پرداخت:

### تعاملات سازمانی (Collaboration)

با استفاده از وب پارت ها می توان امکاناتی نظیر blog، wiki، RSS، email و لیست هماهنگی کارها را به منظور ایجاد یک محیط تعاملی به وجود آورد. اشتراک مستندات، ارجاع وظایف به اعضای تیم، پیگیری اشکالات ثبت شده و به طور کلی همه کارهایی که ماهیت پیگیری(Tracking) دارند در این محیط انجام می شوند.

### پورتال (Portal)

MOSS 2007 این امکان را به شما می دهد که متناسب با نوع تجارتتان پورتال مختلفی را ایجاد و نگهداری کنید. این پورتال ها می توانند کاملا سفارشی شوند و برای هر بخش سازمان شما بصورت مستقل عمل کنند.

### جستجو(Search)

فناوری SharePoint با به کارگیری FTS یا Full-Text-Searchکه بر مبنای SQL می باشد یک موتور جستجوی بسیار قوی را در مجموعه سایت های سازمان شما ایجاد می کند.

### مدیریت محتوا (Content Management)

سیستم مدیریت محتوای موجود در SharePoint امکانی را برای شما فراهم می کند تا بتوانید به راحتی داده های نا ساخت یافته ی خود را مدیریت کنید. این سیستم این امکان را به شما می دهد تا صفحات مورد نیازتان را ایجاد و مدیریت کنید.

### فرآیند های سازمان (Business Processes)

امکان ایجاد فرم های الکترونیکی مختلف و قالب های گردش کار را به شما می دهد. شما می توانید با استفاده از برنامه InfoPath که از سری برنامه ی MS Office 2007 می باشد به راحتی این فرم ها را ایجاد و مدیریت کنید.

### آنالیز داده های تجاری(Business Intelligence)

Excel Service این امکان را می دهد تا صفحات برنامه Excel در محیط SharePoint قابل بارگذاری، تغییر و محاسبه باشند.

Business Data Catalog ابزاریست که پورتال سازمان را با نرم افزارهای تجاری بیرون از پورتال مرتبط می کند.با استفاده از این ابزار به داده های بیرون سازمان متصل می شوید، این داده ها را جستجو می کنید و قادر به استفاده ی از آنها در پورتال خود هستید.

### بررسی محصول Microsoft Office SharePoint Designer 2007 :

این محصول یک ابزار توسعه هست که از طریق آن شما قادر به سفارشی کردن سایت های SharePoint، نمودار های گردش کار و فرم های گزارش گیری هستید. توسعه دهندگان ASP.NET می توانند به راحتی صفحات مختلف را ایجاد کنند و یا تغییر دهند. همچنین با استفاده از ویرایشگر WYSIWYG که در این محصول وجود دارد می توان در کدهای XHTML یا CSS نیز تغییراتی ایجاد نمود و یا حتی قالب صفحات و مسترپیج ها را نیز تغییر داد.

در قسمت دوم به چگونگی ایجاد برنامه های SharePoint با استفاده از VS2008 خواهیم پرداخت.

منابع:

Introduction to SharePoint Products and Technologies for the Professional .NET Developer
WSS 3.0 - Getting Started: Site Provisioning
SharePointSolutions.ir Bloggers

***

# ایجاد حالت "Application Offline" در ASP.NET 2.0

نویسنده : بهروز راد

ممکن است یک وب سایت نیاز داشته باشد که برای مدتی از دسترس کاربرانش خارج شود. به عنوان مثال، در سایت های مختلف حتما با عبارت "در حال حاضر سایت به دلیل انجام عملیات به روز رسانی بر روی سرور در دسترس نمی باشد" مواجه شده اید. در این مقاله به چگونگی انجام این موضوع در ASP.NET خواهیم پرداخت.

از دسترس خارج کردن یک وب سایت دو دلیل دارد:

۱- ایجاد تغییرات اساسی بر روی سایت (مثلا تغییر ساختار دیتابیس)

۲- بررسی قسمت هایی از سایت که در زمان آنلاین بودن سایت امکان پذیر نیست.

در ASP.NET 1.x تنهاه راه برای متوقف کردن سایت، درخواست از مدیر هاستینگ برای توقف آن بود. اما در ASP.NET 2.0 این امکان وجود دارد تا به بهترین نحو این کار را انجام دهید.

اصول کار ساده ست!

موتور پردازشگر ASP.NET در ابتدای هر درخواست، به دنبال فایلی با نام "app_offline.htm"در ریشه ی سایت خواهد گشت و در صورتی که این فایل را پیدا کند به طور خودکار، برنامه را متوقف کرده و محتویات این فایل را به جای محتویات صفحه ی درخواست شده به مرورگر کاربر ارسال خواهد کرد. زمانی که تغییرات مورد نیاز انجام گرفت، کافیست این فایل را حذف کنید تا با دریافت اولین درخواست، برنامه مجددا آنلاین شود!

در فایل "app_offline.htm" معمولا توضیحاتی در مورد دلیل متوقف شدن سایت نوشته می شود.

نکته ی بسیار مهمی که باید بدان توجه داشته باشید این است که:

IE این قابلیت را دارد تا خطاهایی از نوع HTTP را با ذکر جزئیات خطا نمایش دهد. این قابلیت به طور پیش فرض فعال است Tools > Internet Options > Advanced

اما مشکلی که در اینجا وجود دارد این است که در صورت پیدا شدن فایل "app_offline.htm"، سرور ضمن نمایش دادن محتویات این فایل، کد وضعیتی غیر از کد موفقیت پردازش درخواست (۲۰۰) را برگشت خواهد داد. در این حالت، اگر حجم داده های ارسالی به کلاینت کمتر از ۵۱۲ بایت باشد، محتویات فایل "app_offline.htm" نمایش داده نمی شود و به جای آن، خود خطای HTTP با ذکر توضیحاتی ظاهر می شود.

برای رفع این مشکل، در صورتی که حجم فایل "app_offline.htm"شما کمتر از 512بایت شد، مقداری عبارت Comment شده در آن قرار دهید

به همین سادگی!

***

# مفاهیم امنیت نرم افزار و مهندسی معکوس

درس اول : Debugger, Disassembler, Decompiler

نویسنده : (Inprise)

سطح : مبتدی ، متوسط

**دیباگر**: در ادبیات مرسوم مباحث مرتبط با امنیت نرم افزار ، دیباگر نرم افزاری است که امکان بررسی دقیق روند اجرای یک برنامه اجرائی یا کتابخانه یا درایور را فراهم می کند و در این روند ، کنترل کامل و جزئی حافظه ، متغیرها و توابع ، تراکنش های پردازنده و حافظه و دیسک و ... و ارتباط با کتابخانه های اشتراکی در اختیار دیباگر می باشد .

دیباگر برنامه اجرائی یا کتابخانه یا درایور را با ترتیباتی قابل پیکره بندی فراخوانی می کند ، به توسعه گر یا آزمایش کننده این امکان را می دهد که در مواقع به خصوصی روند اجرا را متوقف یا منوط به وقوع اتفاق خاصی نماید ، یا حافظه را برای کشف مقدار یا مقادیر به خصوصی جستجو کند یا تراکنش های برنامه با کتابخانه های همراه ، کتابخانه های سیستم عامل ، سخت افزارهای همراه و ...را کنترل نماید . در واقع وظیفه ی اصلی یک دیباگر کنترل کامل فرآیند اجرا شدن یا فراخوانده شدن یک برنامه یا کتابخانه اشتراکی است . اطلاعاتی که می توان با استفاده از دیباگرها به صورت معمول بدست آورد:

- هر آنچه بین نرم افزار و حافظه اتفاق می افتد. آدرس دهی متغیرها و توابع ، جایگزینی مقادیر ، تخصیص حافظه و آزاد سازی ، محل دستورات زبان ماشین در حافظه ، داده های در حال ارسال به پردازنده و...
- وضعیت لحظه به لحظه ی پردازنده ؛ محتویات رجیسترها و ...
- هر آنچه بین نرم افزار و کتابخانه هائی که از آن تغذیه می کند واقع می گردد . فراخوانی توابع ، کلاس ها یا اشیا ؛ آدرس کتابخانه ها و توابع مربوطه در حافظه و...

عموم دیباگرهای متداول غیر از موارد فوق اطلاعات متنوع دیگری را نیز فراهم می کنند که بسته به مورد و نوع کاربرد متفاوت هستند.

## انواع دیباگر ها

**از دید لایهء کاربرد:**

- Application Mode
- Kernel Mode

دیباگرهای Application Mode فقط به Ring3 و فضای User Modeسیستم عامل دسترسی دارند و امکان Resolve آدرس ها فقط در این محدوده برایشان موجود است اما دیباگرهای Kernel Mode یا اصطلاحا Kernel Debugger ها امکان دسترسی به فضای آدرسی Ring0 یا Kernel Spaceرا نیز دارند و در صورت وجود Symbol file ها ، امکان Resolve آدرس ها در فضای کرنل نیز برایشان وجود دارد.

**از دید نوع عملکرد:**

- Source code Level
- Assembler Level

دیباگرهای Source Code Level با استفاده از سورس کد برنامه ، وظایف مذکور را کنترل و بررسی می کنند و عموما هنگام توسعه ی یک نرم افزار ، به خطا یابی یا ایجاد روندهای کنترل خطای منطقی کمک بسیاری می کنند اما دیباگرهای Assembler Level بدون استفاده یا نیاز به سورس کد و صرفا با داشتن نسخه ی باینری ، وظایفشان را انجام می دهند . امکان کنترل کد ، در سطح Assembler وجود دارد یعنی فرضا می توانید روی دستوری مانند Call یا Test یا Mov برنامه مورد نظر یک Breakpoint بگذارید.

دیباگرها اولین وسیله مورد استفاده نفوذگران نرم افزاری است فلذا روتین های آنتی دیباگ همیشه یکی از عناوین مطرح در روش ها و ابزارهای حفاظت از نرم افزار هستند . این روتین ها عموما از این روش ها برای شناسایی دیباگر استفاده می کنند:

- استفاده از مشخصات فردی دیباگر: نام پروسه ، مشخصات ثبت شده در رجیستری ، مولفه های هدر و...
- بررسی حافظه مجازی اختصاص یافته به برنامه و تست وجود دیباگر ( Hook آدرس یا Hook آدرسِ آدرسِ توابع دیباگ

ویندوز و بررسی وجود پوینتر به آنها یا عدم وجود آن)
- بررسی پردازنده ( مخصوص روتین های Ring0 )

برای تمامی این Trick ها و روش هائی که از هر کدام منشعب می شود نیز ، بالتبع روش های متقابلی وجود دارد ؛ در مجموع چیزی به عنوان متوقف کردن قطعی همه دیباگر ها منطقا غیر ممکن است ؛ بهترین روتین های موجود که هر کدام با مدت ها تحقیق و بررسی ارائه شده اند در کمترین زمان ممکن bypassشدند. استفاده از روتین های آنتی دیباگ برای حفاظت از نرم افزار روش مناسبی برای کند تر کردن فعالیت نفوذگران تازه کار است ، نه چیزی بیشتر .

به جای لیست کردن دیباگرها و توضیحات غیر مفید در مورد هر کدام ، توصیه های شخصی ام را می نویسم که صرفا نشان دهنده ی دیدگاه خودم است نه چیز دیگر.

Windbg : دیباگر مایکروسافت . رابط کاربری نسبتا خوبی دارد و به خوبی با Windows Symbol files متصل می شود . مطمئنا داشتنش برای کسانی که به هر دلیل به دیباگر نیاز دارند یک ضرورت است.

Ollydbg : قدرتمند ترین دیباگر Assembler Level و Application Mode موجود . اسکریپت های متعددی برای خودکار سازی بسیاری از وظایف توسط افراد مختلف برایش نوشته شده . زندگی بدون Olly برای نفوذگران نرم افزاری قطعا غیر ممکن است . OllyDbg توسط BCB6 توسعه داده شده و می شود.

SoftICE : قدرتمند ترین دیباگر Kernel Mode که به صورت همزمان Source Level و Assembler Level نیز هست . به جای تکیه بر رابط کاربری ، دستورات پیچیده ای دارد که انعطاف بیشتری به روند دیباگ می دهد . غیر حرفه ای ها نمی توانند رابطه ی خوبی با آن برقرار کنند . نسخه های متعددی دارد که هر کدام را باید در یک فضای خاص استفاده کرد. Visual SoftICE به عنوان آخرین نسخه ارائه شده ، همراه با DriverStudio ی DevPartnet ارائه می شود و امکانات بصری خوبی به نسخه های سنتی SICE اضافه کرده است.

kd : یا Kernel Debugger ، دیباگر استاندارد DDK ویندوز یا Driver development Kit است . یک Kernel Mode درایور که هم به صورت Assembler Level و هم به صورت Source Level کار می کند . با SICE قابل مقایسه نیست اما چون سازگاری خوبی با Windows Symbol Filesدارد ، برای دیباگ درایورها فوق العاده مفید است.

دیباگرهای دیگری هم وجود دارند که هر کدام ممکن است نسبت به موارد فوق مزایا یا نقائصی داشته باشند ؛ لیکن تجربه نشان می دهد OllyDBG برای اغلب کاربردها ، SICE برای حرفه ای ها و kd نهایتا برای توسعه گران Ring0 ، همه نیازها را برطرف می کنند.

## ۲- Disassembler

ابزاری برای تبدیل کدهای زبان ماشین به معادل اسمبلی آن هاست . زبان اسمبلی به عنوان یک زبان سطح پائین ، به ازای یک دستور یا اصطلاحا Instruction به یک دستور زبان ماشین تبدیل خواهد شد فلذا بازگرداندن کد باینری به معادل اسمبلی ان کار چنان دشواری نیست.

سوال : یک برنامه اجرائی ( PE ) تحت ویندوز داریم که با MASMیا Macro Assembler نوشته شده است . با یک Disassembler آن را به کدهای اسمبلی تبدیل می کنیم . آیا خروجی لزوما با کد اسمبلی اولیه یکسان خواهد بود ؟

جواب : لزوما خیر. Disassembler های امروزی عموما به اندازه کافی هوشمند و خوب هستند اما نمی شود انتظار داشت "هوش" داشته باشند. این ابزارها با استفاده از Knowledge Baseای که برای کامپایلرهای مختلف داخلشان تعبیه شده ، کدهای باینری را به معادل اسمبلی تبدیل می کنند اما تضمینی وجود ندارد توسعه گر اصلی برنامه نیز دقیقا همین کدها را نوشته باشد. میزان خطای Disassembler ها رابطه مستقیمی با میزان پیچیدگی و سطح بالا بودن زبان دارد . یعنی Disassembler با دریافت ورودی که با MASM یا GCC تولید شده خطای کم تری خواهد داشت نسبت به زمانی که قرار است فایل باینری تولید شده توسط دلفی یا VC را بررسی کند؛ خصوصا اگر کتابخانه های مفصلی مثل VCL و MFCهم استفاده شده باشند؛ لیکن در مجموع می شود تا حدود زیادی به خروجی Disassembler های امروزی برای درک منطق و روند فعالیت برنامه اعتماد کرد.

سوال: Disassembler چه کاربردهایی دارد ؟

جواب : فرض کنید قرار است روتین مقایسه ی سریال نامبر یک Protection کودکانه را بررسی کنید . دیباگر به شما کمک خواهد کرد تا بتوانید نقل و انتقال مقادیر بین متغیرها و

محل انجام مقایسه را پیدا کنید و Disassembler کمک خواهد کرد با مشاهده ی دقیق و تک تک دستورات ، انتخاب های خوبی برای عبور از ان حفاظ یا تولید یک Patch و دستکاری نسخه ی باینری برنامه داشته باشید . برای یک نفوذگر نرم افزاری حرفه ای ، مطالعه ی خروجی Disassembler از یک روال حفاظتی ، معادل مطالعه ی سورس کد است.

Disassembler های معتبر و قابل اتکا تحت ویندوز یعنی IDA Pro ، WinDasm و PView دارای امکانات دیگری هم هستند:

- آنالیز کد و ایجاد ارتباطات بصری بین اجزاء و روتین های مختلف برنامه ؛ نمایش توالی اجرا توابع و کاربری از اشیاء و...
- آنالیز کد باینری و تشخیص توابع داخلی به کار برده شده توسط کامپایلرها و ارائه کامنت های مفید
- ارائه کردن امکانات یک دیباگر در کنار Disassembler
- و ...

در مجموع Disassembler به عنوان دومین ابزار مهم در حوزه ی امنیت نرم افزار ، یکی از عناصر لا ینفک یک روند حرفه ای آنالیز و بررسی باینری است. IDA Pro توسط یک تیم توسعه گر هماهنگ ، قدرتمند ترین Disassembler موجود در محیط ویندوز است. (یک نسخه ی مبتنی بر لینوکس هم دارد ، لیکن انتظارات نسخه ی ویندوزی را نمی شود از آن داشت) این محصول با Borland C توسعه داده شده و یک دوره ی فشرده ی آموزشی ان توسط خود شرکت توسعه گر به مدت چهار روز ، برای هر نفر چیزی حدود هزار و پانصد دلار هزینه در بر خواهد داشت WinDasm . دیگر تحت توسعه نیست و نسخه های جدیدتری نخواهد داشت . آخرین نسخه رسمی ۸.۹ است که دو نسخه ۹ و ۱۰ هم توسط افرادی که برایش Patch هائی نوشته اند منتشر شدند.

WinDasmمدت ها به عنوان ابزار شماره یک استفاده شده ( خصوصا با توجه به گران قیمت بودن IDA ) و بسیاری از مقالات و راهنماهای موجود مبتنی بر ان نوشته شده اند. IDA Pro با داشتن آنالایزر قدرتمند و امکان شناسائی کتابخانه های مختلف ، داشتن پلاگین های متعدد و قابلیت های بی شمار ( که بی اغراق امکان ندارد حتی بتوان در قالب یک کتاب در مورد همه شان حرف زد) بهترین گزینه موجود است؛ هر چند تسلط به ان حتی برای کسانی که دانش بالایی دارند واقعا دشوار و پر هزینه خواهد بود.

نکات: دستور mov al, 0x61 یا انتقال عدد ۹۷ به al روی IA32می تواند چنین معادلی داشته باشد: B061 یا باینری : ۱۰۱۱۰۰۰۰۰۱۱۰۰۰۱ ؛ اما این ترجمه زمانی درست انجام می شود که Disassembler بفهمد باید B061 را اولا در بخش کد و ثانیا به عنوان کد ماشین در نظر بگیرد ؛ در ضمن Instruction Setمورد نظر یعنی مثلا IA32 را به دقت و با در نظر گرفتن همه ی جزئیات پیاده سازی کرده باشد ؛ از ان جایی که هنوز انسان ها برنامه ی کامل و بدون نقصی تولید نکرده اند ، Disassembler ها هم در اثر پیچیدگی های نرم افزار ، تفاوت خروجی کامپایلرهای مختلف و تفاوت های جزئی برخی از Binary Encoding های Assembler ها ( مثلا MASMمایکروسافت و BASM بورلند و تفاوت هایشان ) ایضا نقایص موجود در پیاده سازی IS باعث می شود Disassemblerنتواند همیشه همه چیز را درست تشخیص دهد. به عنوان مثال 01100001 10110000 همواره معادل یک دستور ماشین نیست؛ فقط به شرطی که در محل مناسب قرار بگیرد یک دستور ماشین است ، در حالی که داخل برنامه اجرائی وجود دارد. برای دیدن این تفاوت ها دو تا فایل اجرایی ، یکی ساده و یکی بزرگ و پیچیده را به هر دوی IDA Pro و WinDasmداده و خروجی ها را تماشا کنید

### ۳- Decompiler

به لحاظ تئوریک یعنی ابزاری برای تبدیل یک برنامه ی باینری اجرایی یا یک کتابخانه یا درایور به سورس کد اصلی؛ قبل از ورود به بحث لازم است یک طبقه بندی از موجودیت هایی که ممکن است ذیل عنوان Decompiler مطرح شوند داشته باشیم:

- برنامه های اجرایی باینری: برنامه هایی که عموما با زبان های سطح بالایی نظیر VC یا دلفی نوشته شده و به کدهای "مخصوص" به ویندوز/معماری ماشین ( مثلا Win32/IA32 یعنی ویندوز ۳۲ بیتی روی اینتل ۳۲ بیتی ) ترجمه می شوند
- .کتابخانه های اشتراکی: بسته های نرم افزاری که عموما با زبان های سطح بالا برای کاربری در سایر برنامه ها تولید می شوند و وابسته به سیستم عامل و معماری سخت افزاری هستند.

- برنامه های تفسیری: برنامه هایی که قبل از هر بار اجرا باید توسط یک مفسر ترجمه شوند . به عنوان مثال برنامه های VB6که به صورت PCode منتشر شده و هر بار قبل از اجرا توسط VB runtime تفسیر می شوند.

- برنامه های مبتنی بر زمان اجرا: برنامه هایی که برای اجرا نیاز به بستر از پیش فراهم شده ای برای روند اجرا دارند. مانند برنامه های دات نت و جاوا.

- درایور ها: کدهای سطح کرنلی که مختص سیستم عامل و معماری سخت افزاری هستند و عموما با زبان های سطح پایین تولید می شوند.

سوال: آیا معنای تئوریک Decompiler برای همه این گروه ها محقق شده ؟ می شود ؟ خواهد شد ؟

جواب : خیر.

هیچ Decompiler ای برای گروه های اول و دوم و پنجم ارائه نشده ، نمی شود ، نخواهد شد . یعنی دریافت سورس کد کامل نرم افزارهای اجرایی از نسخه باینری ان ها مطلقا غیر ممکن است . این عدم امکان فنی نیست که در آینده با پیشرفت دانش امکان پذیر شود ؛ یک نفی منطقی است . یعنی منطقا امکان باز تولید سورس کد کامل یک برنامه تولید شده با محیط هایی مثل Delphi یا VC وجود نداشته ، ندارد ، نخواهد داشت.

سوال : پس نرم افزارهای متعددی که تحت عنوان Decompilerمنتشر می شوند چه ؟

جواب : با توجه به تعریف Decompiler جواب داده شد.

سوال : در مورد گروه های سوم و چهارم چه ؟

جواب : برای این دو گروه Decompiler وجود داشته و دارد؛ با یک توضیح کوچک .برنامه هایی هستند که می توانند از برنامه ی اجرایی( VB به عنوان نماینده گروه سوم ) یک سورس کامل قابل کامپایل تولید کنند ، اما این سورس ، لزوما قرار نیست همان سورسی باشد که توسعه گران نرم افزار تولید کرده اند؛ برای دات نت ( نماینده ی گروه چهارم ) نیز Decompiler های متعددی وجود دارد؛ اما هیچ کدام قول نمی دهند خروجی آن ها لزوما همان سورس کدی باشد که برنامه از آن تولید شده.

سوال : آیا اصولا وجود Decompiler لازم است ؟

جواب : برای اهداف مثبت و خیرخواهانه خیر . حتی برای اهداف غیر خیرخواهانه نیز وجود Decompiler یک لازمه نیست . هیچ کسی از وجود ابزاری که بتواند برنامه ی وی را به سورس قابل قبولی مبدل کند خوشحال نخواهد شد؛ این ابزار کمکی به توسعه نرم افزار نمی کند و سود اقتصادی ، پیشرفت علمی و افزایش قابلیت های صنعت نرم افزار را بیشتر نخواهد کرد . حتی برای اهداف مخرب هم وجود چنین ابزاری لازم نیست ، چون بسیاری از کسانی که در این مسیر فعالیت می کنند برای تخریب امنیت یک نرم افزار نیازی به دسترسی به سورس ان ندارند. کشف نقاط ضعف امنیتی یا عبور از حفاظ های نرم افزار عموما در محیط هایی اتفاق می افتد که سورس وجود ندارد و تمام فرایند تخریب از طریق مهندسی معکوس یا Reverse Engineeringانجام می گیرد .

سوال : برنامه هایی که با جاوا و دات نت نوشته می شوند چقدر امن هستند ؟

جواب : چون نقطه ی صفری وجود ندارد، میزانی قابل ارائه نیست؛ اما در مقام مقایسه:

- بررسی و Trace و بازبینی روند اجرای برنامه هایی که با محیط هایی نظیر دات نت و جاوا تولید می شوند به مراتب دشوارتر از برنامه هایی است که با محیط هایی نظیر دلفی و VCتولید می شوند؛ چرا که وجود Runtimeهای بزرگی مانند JRE یا CLR باعث می شود پیچیدگی فراخوانی ها ، مدیریت حافظه ، مدیریت ریسمان ها و پردازه ها و غیرهم به مراتب از برنامه های اصطلاحا Native (مانند خروجی های VC) بیشتر باشد . پس فی المثل درک جزئیات فنی یک الگوریتم ، وقتی با دات نت نوشته شده باشد و به خوبی با Framework مخلوط باشد واقعا دشوار تر از درک جزئیات فنی الگوریتمی است که با Delphi کامپایل شده.

- عبور یا تخریب حفاظ های نرم افزارهایی که با زبان هایی نظیر جاوا و دات نت نوشته می شوند به مراتب آسان تر از برنامه هایی است که با امثال دلفی و VC تولید می شوند. چرا که اگر از درک جزئیات یک الگوریتم بگذریم ، وابستگی کامل این برنامه ها به یک لایه ی میانی به نام زمان اجرا و عدم وابستگی به عناصر زیر ساختی سستم عامل و پردازنده و سخت افزار و همچنین امکانات بیشتر نفوذگران نرم افزار در تغییر

محتویات این برنامه ها باعث می شوند از این دیدگاه برنامه های Native وضع بهتری داشته باشند.
(بگذریم از این حقیقت که برای یک حرفه ای اهمیت خاصی ندارد که یک برنامه با دلفی کامپایل شده یا Managed CPP
)

سوال : روش های حفاظتی که برای مقابله با Decompiler ها مورد استفاده قرار می گیرد چقدر قابل اعتمادند؟
جواب : برای امثال دات نت و جاوا ، تقریبا هیچ . برای سایر محیط ها Decompiler دشمن خطرناکی به حساب نمی اید . فی المثل برنامه ای با عنوان DeDe یا Delphi Decompilerمدعی است که یک Decompiler برای دلفی است؛ اما در واقع فقط می توانید یک سری اطلاعات دریافت کنید و نه سورس کد کامل . ممکن است در برخی موارد این اطلاعات بتواند به یک نفوذگر نرم افزاری کمک خاصی بکند، اما من حیث مجموع این گونه برنامه ها تهدید خطرناکی به حساب نمی آیند. بگذریم از این واقعیت که یک نفوذگر نرم افزاری برای حذف روتین حفاظتی نرم افزار یا جستجوی یک سرویس برای نقاط ضعف متداول ، نیازی به یک Decompiler ندارد . تمام وقایع تلخی که سالهاست شاهدش هستیم تحت شرایطی می افتد که هیچ Decompilerضعیفی هم وجود ندارد.
مسئله Decompilation همیشه با دو مشکل مواجه است . یکی پیاده سازی "زبان" و دیگری درک "منطق" . یک کامپایلر برای تولید خروجی لزوما نباید منطق یک برنامه را درک کند؛ اما یک Decompiler باید بتواند روند صحیح عمل بازگشت را تشخیص دهد ، که این بدون داشتن درک نسبی نسبت به منطق برنامه عموما امکان پذیر نیست. اما تفاوتی که من برای توضیحات مربوط به ان ۵ دسته قائل شدم با این "مشکلات" ارتباطی نداشتند. یعنی یک Decompiler برای Delphi و یکی برای دات نت هر دو تا حدودی با هر دو مشکل مواجه هستند( در واقع دلیل این که نوشتم خروجی دیکامپایلر لزوما سورس کد اولیه نیست وجود همین دو مسئله است ) دلیل تمایز دیکامپایل برنامه های Native و امثال دات نت و جاوا در یک کلمه خلاصه می شود : جزئیات پیاده سازی کامپایلر. جاوا و دات نت هر دو روی بسترهائی قرار دارند که اصطلاحا Open Standard هستند. یعنی بسیاری از جزئیات پیکره بندی ان ها برای توسعه گران ثالث منتشر شده. جاوا نسخه های سورس آزاد هم دارد و دات نت به عنوان یک استاندارد صنعتی ثبت و مشخصات جزئی و فنی CLR منتشر شده است. اما دلفی ؟ VC ؟ و امثالهم ؟
شما هیچ اطلاعات مستندی در مورد نحوه ی عملکرد و جزئیات کامپایلرها ندارید و نمی توانید داشته باشید. هیچ انسان عاقلی را ایضا نمی شناسم که تا به حال به مهندسی معکوس کامپایلرها دست زده باشد؛ ان هم کامپایلرهای غول پیکری مانند دلفی و VC و ... ؛ و این بدان معناست که کسی نمی تواند در تمام حالات رفتار کامپایلر ، نوع برخورد با عبارات و ترجمه ، مدیریت استثنا ها و محصور سازی توابع سیستم عامل و ... را پیش بینی و پیاده سازی کند؛ و این یعنی هر لحظه امکان دارد با یک تغییر کوچک در کد ، خروجی به خصوصی تولید شود که یک تحلیل گر باینری نتواند ان را به معادل مناسبی مبدل کند؛ یا معادل انتخابی لزوما کاربردی هم باشد؛ در واقع می شود گفت تولید ابزاری که بتواند خروجی کامپایلر A را کاملا صحیح تحلیل کند ، مساوی است با روند "کامل" و "موفق" مهندسی معکوس همان کامپایلر.
در درس بعدی (شمارهٔ بعد) به Compressor, Packer, Encryptor می پردازم.

# آموزش پایتون Python

## درس اول

نویسنده : مهدی بیاضی

سطح : مبتدی ، متوسط

پایتون یک زبان اسکریپتی و بسیار قدرتمند و گسترده هست و این روزها می توانید آثاری از نفوذش را تقریبا همه جا ببینید. از پروژه های ساده و بسیار کوچک گرفته تا پروژه های عظیم چون برخی از نرم افزار های . NASA شاید بپرسید دلیلش چیست ؟ خب دلایل زیادی دارد . هر زبانی برای یک استفاده بخصوص طراحی شده و در همان زمینه بهتر جواب می دهد . زبانی مثل Pascal برای آموزش برنامه نویسی , زبان هایی چون C و اسمبلی هم برای برنامه های سیستمی و سیستم عامل و زبانی چون php برای طراحی سایت . و استفاده از هر یک به جای دیگری امکان پذیر ولی نا صحیح می باشد . علاوه بر اینکه Python زبانی عملی برای برنامه های کاربردی است ، می شود از آن در زمینه های دیگری مثلا برنامه نویسی سیستمی – رابط کاربری (gui) – کامپونت – برنامه نویسی اینترنت – برنامه های عددی و محاسباتی – برنامه های پایگاه داده – پردازش تصویر – هوش مصنوعی – اشیا توزیع شده – شبیه سازی – روباتیک – برنامه نویسی موبایل – امنیت و شبکه و ... استفاده کرد!

و اما چرا باید از پایتون استفاده کنیم ؟؟

پایتون شی گراست :

پایتون ذاتا یک زبان شی گراست و از ویژگی های پیشرفته شی گرایی چون وراثت – چند شکلی – سربار گذاری عملگر و ... پشتیبانی می کند اگر شما با برنامه نویسی شی گرا آشنایی ندارید پایتون یک راه حل عالی برای یاد گرفتنش است. نکته جالب و یک ویژگی منحصر به فرد پایتون که لقب چسب را برای آن به ارمغان آورده امکان استفاده از کد ها و کلاس های نوشته شده در زبان های دیگری چون C++ و Java هست و در واقع کار چسباندن قطعات کد جدا و فقط نوشتن بدنه اصلی به عهده پایتون است .

پایتون رایگان است !

پایتون یک زبان رایگان و باز متن است که این روزها بحث اکثر محافل شده . با کمی توجه به این نکته که دیر یا زود قانون کپی رایت در کشور ما هم اجرایی خواهد شد، فکر نمی کنم در کشوری مثل ایران کسی حاضر شود نر افزار را با قیمت واقعی ان بخرد (مثلا ۱۰۰ هزار تومن برای ویندوز !) پس می شود گفت تها گزینه برای ما نرم افزار های باز متن و رایگان است. البته تنها دلیل رایگان بودن این دسته از نرم افزار ها نیست . موضوعات مهم دیگری هم هست که هر یک مقوله ای جدا هستند ...

### پایتون یک زبان پورتابل یا قابل انتقال می باشد :

چون پایتون با زبان C نوشته شده می تواند به صورت مجازی بر روی هر سیستمی کامپایل و اجرا شود . اگر یادتان باشد گفتم این یک زبان اسکریپتی است یعنی در حالت معمول به فایل اجرایی تبدیل نمی شود بلکه یک ماشین مجازی فایل کد را خوانده و همزمان ان را تفسیر کرده و اجرا می کند . پس شما می توانید یک برنامه را در ویندوز نوشته و سپس آن را بدون تغییر روی لینوکس یا مکینتاش یا هر سیستم عامل و سخت افزار دیگری که پایتون روی آن نصب باشد اجرا کنید .

پایتون قدرتمند است :

پایتون یک زبان چند رگه است . پایتون از زبان های اسکریپتی (برای مثال Tcl , Scheme , Perl ) و زبان های سیستمی (برای مثال C , C++ و Java) مشتق شده . بنابراین سادگی و راحتی کار زبان های اسکریپتی و ویژگی ها و قدرت زبانهای سطح پایین را داراست .

پایتون قابلیت استفاده از کد های نوشته شده با سایر زبان ها را دارد :

این ویژگی یکی از پرکاربرد ترین و قوی ترین ویژگی های پایتون می باشد . شما می توانید قطعه ای از کد را در زبانی چون C , C++ یا Java نوشته ، آن را تبدیل به فایل اجرایی کرده و سپس از آن در برنامه نوشته شده با پایتون استفاده

کنید . و یا از توابع کتابخانه و کامپوننت هایی چون COM بهره ببرید .

یادگیری و استفاده از پایتون بسیار راحت می باشد :

بی شک و حداقل از نظر من و بسیاری از برنامه نویسان پایتون این زبان یکی از آسان ترین زبان ها برای یادگیری و استفاده می باشد و از آن به عنوان یک زبان سریع برنامه نویسی یاد می کنند . این زبان نیازی به کامپایل ندارد و شما مستقما می توانید پس از نوشتن کد و با یک دستور آن را اجرا کنید . دستورات این زبان بسیار نزدیک به زبان انسان می باشد . برای مثال برنامه Hello World را که اولین برنامه ساده می باشد را در دو زبان C و Python مقایسه کنید:

```
#include <stdio.h>
int main()
{
	printf("Hello World ");
	return 0;
}
```

ابتدا باید این برنامه را بنویسید و آن را با دستوری چون cc helloworld.c -o hello کامپایل و سپس با دستوری دیگر ان را اجرا کنید . اما به وسیله پایتون در یک فایل متنی (مثلا hi.py) بنویسید:

```
print "Hello World !!"
```

و با دستور python hi.py اجرا کنید .

با کم شدن حجم کد برنامه نویسی راحت تر و رفع اشکال و توسعه آن آسان تر می شود .

اهمیت پایتون :

احتمالا با اهمیت لینوکس و نرم افزار های باز متن در ایران آشنا هستید. همانطور که می دانید سیستم عامل ملی ما بر پایه لینوکس می باشد و این لازمه توسعه برنامه نویسی تحت لینوکس را برایمان آشکار می کند . شاید فکر کنید زبان برنامه نویسی و انتخاب اکثر لینوکس کاران C می باشد. این تفکر درستی است اما نه برای برنامه های کاربردی مانند یک برنامه حسابداری . پایتون در سراسر دنیا از اهمیت ویژه ای برخوردار می باشد و رفته رفته به جایگاه اصلی و لایق خود نزدیک می شود . جالب است بدانید مایکرو سافت نیز یک پیاده سازی از این زبان را با نام IronPython در تکنولوژی Net. خود گنجانده است.

توجه : من تمام مثال ها را در لینوکس (FC۴) می نویسم و تست می کنم و همه دستورات در این محیط هست ، منتها شکل دستورات یا نحوه ی اجرای آن ها در سیستم عامل های دیگر مانند ویندوز نیز به همین شکل است. برای شروع کار برنامه ی پایتون را از www.python.org دانلود و نصب کنید.

کد:

```
>>> print "Hello world!"
Hello world!
                    برای خروج Ctrl-D را بفشارید ( Ctrl-Z در برخی سیستم عامل ها)
```

در حالت دوم فایلی را با پسوند py ایجاد و کد خود را درون آن بنویسید : (فایل test1.py)

کنید. اکثر توزیع های کنونی لینوکس یک نسخه از مفسر پایتون را در خود دارند .

قبلا ما از پایتون به عنوان یک زبان برنامه نویسی یاد کردیم . اما پایتون نام یک بسته نرم افزاری به نام مفسر نیز هست که کار اصلی آن اجرای برنامه می باشد . مفسر سطر به سطر کد برنامه را خوانده و همزمان اجرا می کند . در حالت کلی یک برنامه پایتون به ۴ طریق اجرا می شود :

- به حالت محاوره ای
- به عنوان ماجول پایتون
- به عنوان فایل اسکریپت unix
- از داخل یک سیستم دیگر

حالت محاوره ای راحت ترین حالت اجرای کد در پایتون می باشد . کافیست کلمه پایتون رو وارد کنید: (در shell یا Command Prompt)

python

و سپس دستورات خود را به صورت محاوره ای وارد کنید (با نوشتن هر دستور و فشردن دکمه ی Enter ، دستور اجرا شده و نتیجه اش چاپ می شود)

کد:

```
import sys
print sys.argv
```

و سپس از طریق ترمینال آن را اجرا کنید:

کد:

```
 python test1.py -I eggs -o bacon
['test1.py', '-I', 'eggs', '-o', 'bacon']
```

این برنامه بسیار کوچک که با نام test1.py ذخیره شده پارامتر های ورودی را به صورت یک لیست چاپ می کند . دستور معادل آن در ویندوز عبارت است از:

کد:

```
C:\> python test1.py -i eggs -o bacon
['test1.py', '-i', 'eggs', '-o', 'bacon']
```

حالت سوم اجرا در حال اسکریپت می باشد (مختص لینوکس و یونیکس) این نوع فایل متنی حالت اجرایی دارد و مشخصه آن اولین خط آن می باشد که یک مثال ساده عبارت است از:

کد:

```
#!/usr/bin/env python
print 'The Bright Side of Life' # comment
```

توضیح این که اولین خط این برنامه را با کامنت ( توضیحات ) اشتباه نگیرید این سطر به این معنی است که کل کد ادامه فایل را با برنامه ای که آدرس آن پس از عبارت !# آمده فرستاده و به اجرا در می آید . پس این خط آدرس فایل اجرایی مفسر پایتون به همراه علامت های !# است . سپس این فایل را به حالت اجرایی تبدیل کرده و اجرا کنید:

کد:

```
 chmod +x test2.py
 ./test2.py
 The Bright Side of Life
```

## عبارات:

عبارات و انجام محاسبات ریاضی بسیار شبیه سایر زبانهای برنامه نویسی می باشد

کد:

```
3 + 5
3 + (5*4)
3 ** 2
'Hello' + 'World'
```

مقدار دهی متغییر ها:

نوع متغییر ها به صورت پویا تعیین می شود و در طول اجرای برنامه ممکن است تغییر یابد متغییر ها تنها اسمی برای یک شی می باشند و همانند زبان C به یک نقطه از حافظه اشاره نمی کنند

کد:

```
a = 4
b = a * 4.5
c = (a+b)/2.5
a = "Hello World"
```

همانطور که مشاهده می کنید تعریف یک متغیر به وسیله مقدار دهی آن انجام می شود و این نظریه درست نیست که در پایتون نیازی به تعریف متغییر نمی باشد برای مثال استفاده از کدی مثل :

```
print x
```

باعث نمایش یک پیغام خطا خواهد شد چون متغییر x قبلا تعریف (مقداردهی) نشده است .

نکته دیگر این که ما در پایتون نیازی به تعیین نوع یک متغییر نداریم برای نمونه در مثال بالا متغییر a ابتدا دارای مقدار صحیح (int) می باشد و سپس از آن برای نشان دادن یک رشته استفاده شده است .

رشته :

رشته اولین نوع داده ای است که با آن آشنا می شوید . فرق رشته در پایتون با اکثر زبان های برنامه نویسی این است که یک کاراکتر به عنوان یک رشته با طول یک می باشد . توضیح دیگر اینکه علامت نقل قول دوگانه و منفرد تقریبا کار یکسانی انجام می دهند. البته فرق هایی دارند که به مرور و با مثال آشنا خواهید شد. از ۳ علامت نقل گول هم برای وارد کردن رشته هایی دارای علامت های خاص استفاده می شود .

کد:

```
>>> 'Ali said "How\'re we supposed to know that?"'
'Ali said "How\'re we supposed to know that?"'

>>> """This is kind of a special string, because it violates some
...     rules that we haven't talked about yet"""
"This is kind of a special string, because it violates some\n     rules that we
haven't talked about yet"
```

### فرمت بندی رشته:

مانند فرمتد بندی C می باشد:

کد:

```
>>> "Mehdi %s" % (" Bayazee")
'Mehdi Bayazee'

>>> "%-10s %s %10s" % ("Name", "Id", "Family")
'Name       Id     Family'

>>> "Controlling the number of decimal places shown: %.02f" % 25.101010101
'Controlling the number of decimal places shown: 25.10'
```

### اعداد و عملگر ها:

پایتون دارای نوع های پیش فرض integers,long numbers,floating-point(float), imaginary numbers می باشد . برای اطلاع از نوع یک متغییر در هر لحظه می توانید از دستور type استفاده کنید:

کد:

```
>>> x=2000
>>> type(x)
<type 'int'>

>>> type(999999999999)
<type 'long'>

>>> type(1.0)
<type 'float'>
```

در مورد تقدم عملگر و بسیاری از بحث های تکراری توضیحی داده نمی شود و به مرور با آن ها آشنا خواهید شد .

### استفاده از سایر انواع پیشرفته پایتون :

پایتون دارای ۳ نوع درونی و پیشرفته تقریبا قابل مقایسه به آرایه و ساختمان Cمی باشد . که عبارت است از :

۱- Tuple

۲- List

۳- Dictionary

Tuple همانند آرایه می باشد با این تفاوت که پس از تعریف و مقدار دهی اولیه قابل هیچ گونه تغییری نمی باشد . علامت مشخصه آن در هنگام تعریف پارانتز می باشد .

کد:

```
>>> filler = (“string”, “filled”, “by a”, “tuple”)
>>> print “A %s %s %s %s” % filler
A string filled by a tuple

>>> print “The second element of the tuple is ‘%s’” % filter[1]
The second element of the tuple is ‘filled’
```

همانطور که متوجه شدید اندیس همانند C از صفر شروع می شود . برای پیدا کردن طول (تعداد اعضا) یک تیوپل از دستور len استفاده می شود هرگونه تلاش برای تغییر یا دسترسی به اندیس بیشتر از طول تیوپل باعث نمایش خطا می شود:

کد:

```
>>> print “%d” % len(filter)
3
>>> print a[len(filter) - 1]
tuple
```

List همانند تیوپل می باشد با این تفاوت که قابلت تغییر و گسترش را دارا می باشد علامت مشخصه آن براکت می باشد.

کد:

```
>>> breakfast = [ “coffee”, “tea”, “toast”, “egg” ]
>>> count = 1
>>> print “Todays breakfast is %s” % breakfast[count]
Todays breakfast is tea
```

و همانطور که گفتیم لیست امکان تغییر را دارا می باشد.

کد:

```
>>> breakfast[2]="milk"
>>> breakfast
['coffee', 'tea', 'milk', 'egg']
```

توجه کنید که شما می توانید تنها عناصر موجود را تغییر دهید برای اضافه کردن عناصر و داده های جدید از دستور append استفاده می کنیم.

کد:

```
>>> breakfast.append(“waffle”)
>>> breakfast
['coffee', 'tea', 'milk', 'egg', 'waffle']
```

اگر قصد دارید تعداد بیشتری از داده ها را در قالب لیست یا تیوپل به لیست موجود اضافه کنید بدین منظور از دستور extend استفاده می کنیم

کد:

```
>>> breakfast.extend([“juice”, “decaf”, “oatmeal”])
>>> breakfast
[‘coffee’, ‘tea’, ‘milk’, ‘egg’, ‘waffle’, ‘juice’, ‘decaf’, ‘oatmeal’]
```

Dictionary شبیه به لیست و تیوپل می باشد با این تفاوت که ایندکس دیکشنری عددی نمی باشد و می تواند شامل رشته یا در اصطلاح هر اسم دیگری باشد.

کد:

```
>>> dic={}
>>> dic["name"]="mehdi"
```

```
>>> dic["family"]="bayazee"
>>> print dic
{'name': 'mehdi', 'family': 'bayazee'}
```

نام ایندکس در دیکشنری keys یا کلید و مقدار هر کلید value نامیده می شود . که برای به دست آورد کلیدها و مقادیر می توانید از توابع keys() و values() استفاده می شود:

کد:

```
>>> dic.keys()
['name', 'family']
>>> dic.values()
['mehdi', 'bayazee']
```

نکات:

می توانید در پایتون از اندیس دهی معکوس استفاده کنید. به این صورت که اندیس ۱- آخرین عضو آرایه می باشد.

تکه تکه کردن (Slicing) آرایه و رشته از طریق ترکیب اندیس آغازین و پایانی صورت می گیرد.

کد:

```
>>>last_names = [ “Douglass”, “Jefferson”, “Williams”, “Frank”, “Thomas” ]
>>>last_names[-1]
'Thomas'
>>>last_names[-5]
'Douglass'
>>> s[1:4]
['Jefferson', 'Williams', 'Frank']
>>> s[0][0:6]
'Dougla'
```

برای مشاهده اعضا هر کلاس (متغییر ها و توابع) می توانید از دستور dir استفاده کنید.

کد :

```
>>> s=[]
>>> dir(s)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__',
'__delslice__', '__doc__', '__eq__', '__ge__', '__getattribute__', '__getitem__',
'__getslice__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__',
'__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__',
'__setattr__', '__setitem__', '__setslice__', '__str__', 'append', 'count',
'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

برای دسترسی با مستندات هر دستور یا کلاس می توانید متغییر __doc__ آن دستور استفاده کنید.

کد:

```
>>> print (s.__doc__)
list() -> new list
list(sequence) -> new list initialized from sequence's items
```

## شرط و حلقه

قبل از اینکه وارد مبحث شرط (if) بشویم یکی از جالب ترین جنبه های پایتون را توضیح می دهم . برخلاف زبان های دیگری چون C و Pascal که از علایم و عباراتی چون {} و begin,end و ... برای نمایش شروع و پایان یک قطعه کد استفاده می کنند پایتون دارای علامت یا عبارت خاصی برای این منظور نمی باشد و عمل مشخص کردن محدوده یک عبارت که شامل قطعه کد می باشد (مثلا تابع – دستورات شرطی – حلقه – کلاس و ... ) توسط فاصله گذاری مشخص می شود . به این صورت که پس از خطی که بقیه دستورات زیر مجموعه آن می باشند به اندازه دلخواه فاصله یا تورفتگی داده می شود. این فاصله های یکسان تا زمانی ادامه می یابد که محدوده دستور پایان یابد . که این روش باعث کاهش مقدار برنامه و افزایش خوانایی برنامه می شود .

دستور شرطی if

شکل کلی if در پایتون به صورت زیر می باشد .

کد:

```
ifشرط:
دستورات
```

مثال : شرط اول به علت اختلاف حرف m مقدار false بوده و هیچ مقداری نمایش داده نمی شود . اما در بخش بعدی با استفاده از تابع عضو کلاس رشته و تبدیل هر دو رشته به حروف کوچک مقدار شرط درست بوده و متن دلخواه چاپ می شود.

کد:

```
>>> if "mehdi" == "Mehdi" :
...     print "Same !!"
...
>>> if "mehdi".lower() == "Mehdi".lower() :
...     print "Same !!"
...
Same !!
```

به فاصله داده شده در اول دستور print توجه نمایید . با این توضیح هرگونه فاصله اضافی که باعث ایجاد ابهام گردد تولید خطا می کند .

در کد زیر فلش زیر حرف p در دستور print نشان دهنده محل اشتباه منطقی موجود می باشد .

کد:

```
>>>     print "Some space !!"
  File "<stdin>", line 1
    print "Some space !!"
    ^
SyntaxError: invalid syntax
```

علامات بکار رفته برای مقایسه دو عبارت و ترکیب عبارات شرطی مطابق علایم زبان C می باشد علایمی چون:

"%" , "=>" , "|" , "&" , "=!" , "==" , "<" , ">"و...

که البته می توانید به جای "&" از and و به جای "|" از or استفاده کنید.

## دستورات

### تکرار:

اولین دستور تکرار یا حلقه while می باشد . به مثال زیر توجه کنید:

کد:

```
>>> omelet={"egg":2,"mushroom":5,"pepper":1,"cheese":1, "milk":1}
>>> ingredients = omelet.keys()
>>> ingredients
['cheese', 'pepper', 'egg', 'milk', 'mushroom']
>>> while len(ingredients) > 0:
...     current = ingredients.pop()
...     print "Adding %d %s to the mix" % (omelet[current],current)
...
Adding 5 mushroom to the mix
Adding 1 milk to the mix
Adding 2 egg to the mix
Adding 1 pepper to the mix
Adding 1 cheese to the mix
```

در این مثال ابتدا یک دیکشنری را تعریف می کنیم . (به تعریف یکجا و جدید توجه کنید ) سپس از طریق دستور keys() کلید (اندیس) های دیکشنری را در متغییر دیگری به نام ingredients می ریزیم . و با وارد کردن نام آن محتویات متغییر را مشاهده می کنیم . (این نوع نمایش فقط در حالت محاوره ای عمل می کند ) سپس با استفاده از دستور while و تا زمانی که تعداد اعضای متغییر ingredients بزرگتر از صفر است وارد حلقه می شویم با تابع pop() که یک عنصر را از لیست جدا کرده (حذف کرده) و بر می گرداند یکی یکی اعضا لیست ingredients که همان کلید های دیکشنری omelet هستند را در داخل متغییر current ریخته و در سطر بعدی چاپ می کنیم .

دستور بعدی برای حلقه ... for ... in می باشد . که شبیه به دستور while می باشد با این تفاوت که در این حالت نیازی به

تعیین شرط پایان حلقه نبوده و در ضمن متغییری که به تک تک عناصر لیست اشاره می کند ایجاد می گردد که می توانید از آن در داخل حلقه استفاده کنید . معادل مثال بالا را با for به صورت زیر می باشد :

کد:

```
>>> for ingredient in omelet.keys():
...     print "adding %d %s to the mix" % (omelet[ingredient],ingredient)
```

همانطور که مشاهده می کنید کد نوشته شده با for بسیار بهینه تر و کوتاهتر می باشد . منطق for در پایتون کمی متفاوت می باشد .

همانند زبان های دیگر حلقه دارای دستوراتی برای جلوگیری از اجرای حلقه بدون تست شرط و نیز جلوگیری از اجرای بخشی از دستورات و تست دوباره شرط می باشد .

با استفاده از break می توانید بدون تست شرط حلقه را ترک کنید . و دستور continue از اجرای دستورات جلوگیری کرده مقدار متغییر را تغییر داده و شرط را بررسی می کند .

## استفاده از else در دستورات تکرار :

این دستور در آخر حلقه های تکرار می اید و زمانی اجرا می شود که حلقه بطور کامل و برای تمامی مقادیر اجرا شده باشد و دستوری مانند break اجرای عادی حلقه را ناقص نکرده باشد .

دو مثال از استفاده else در حلقه for که در اولی به علت اجرای کامل حلقه اجرا نمی شود ولی در مثال دوم اجرا شده است :

کد:

```
>>> for food in ("pate", "cheese", "crackers", "yogurt"):
...     if food == "yogurt":
...             break
... else:
...     print "There's no yogurt!"
...
>>> for food in ("pate", "cheese", "crackers"):
...     if food == "yogurt":
...             break
... else:
...     print "There's no yogurt!"
...
There's no yogurt!
```

به تو در تو بودن if و for و رعایت فاصله دقت کنید . توجه کنید که else دقیقا زیر for نوشته شده است پس متعلق به for می باشد نه if . در حلقه دوم چون yogurt در تیوپل موجود می باشد (تیوپل است چون از پارانتز استفاده شده) شرط if اجرا می گردد و با دستور break از اجرای ادامه حلقه جلوگیری می شود بنابرین بخش else حلقه به اجرا در می آید.

مدیریت خطا ها:

شما در درس های گذشته نمونه هایی از پیام های خطای پایتون را مشاهده نمودید که شامل اطلاعاتی در مورد نوع و نحوه بروز اشکال و خطای موجود می باشد . در مثال بالا اگر سعی کنید به اندیسی دسترسی یابید که تعریف نشده با پیغام خطایی مواجه خواهید شد.

کد:

```
>>> omelet["pate"]
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
KeyError: 'pate'
```

همانطور که مشاهده می کنید نوع خطا KeyError می باشد . شما می توانید با استفاده از دستور try ... except ... خطاهای به وجود آمده را مدیریت نمایید در این حالت کد های مورد نظر در داخل بخش try نوشته می شود و با استفده از بخش except و تعیین نوع خطای به وجود آمده دستوراتی را که باید هنگام به وجود آمدن این خطا اجرا شوند را بنویسید . مثالی را مطرح کرده و در آن سعی می کنیم به اندیسی دسترسی یابیم که وجود ندارد

کد:

```
>>> omelet={"egg":2,"pepper":1,"cheese":1, "milk":1}
>>> try:
...     if omelet["mushroom"] > 0:
...         print "Sure, it have some mushroom"
... except KeyError:
...     print "Aww, there's no mushroom. Lets go shopping !"
...
Aww, there's no mushroom. Lets go shopping !
```

دوباره به تورفتگی کدها دقت کنید . شما می توانید مدیریت دلخواه و قدرتمندی بر روی انواع مختلف خطا های موجود داشته باشید.

خب این از درس اول ، در درس بعدی به توابع و کلاس ها می پردازم.

***

# Resolver One

نویسنده : مهدی عسگری

نرم افزاری که برای این شماره در نظر گرفتم یک صفحه گسترده است که با زبان IronPython[1] نوشته شده. معمولا با شنیدن نام صفحه گسترده یا spreadsheet یاد نرم افزار Excel از بسته ی Office مایکروسافت می افتیم. Resolver One که محصولی از شرکت Resolver Systems[2] است ، صفحه گسترده ای سازگار با Excel است (قابلیت import و export فایل های Excel را نیز دارد) که فرمول های ان به جای VBA در پایتون نوشته می شوند و برای هرگونه عملیاتی کد پایتون تولید می کند. نسخه ی تک کاربره ی این برنامه برای استفاده ی شخصی رایگان است. شرکت مذکور یکی از معدود شرکت هایی است که برای توسعه ی محصولاتشان کاملا از IronPython استفاده می کنند. (به قدری هم استفاده شان از این محصول سنگین و زیاد هست که پروژه ای اپن سورس به نام Ironclad برای استفاده از ماژول های CPython در IronPython ایجاد کرده و همچنین باگی را در IronPython 2.0 پیدا کردند که پس از رفع ان در IronPython 2.0.1 پرفورمنس عملیات مقایسه ی اعداد صحیح و اعشاری ، ۷۴ درصد افزایش پیدا کرد)

تا نسخه ی 1.3.2 (نسخه ی قبلی) از IronPython 1.1 استفاده می کردند و از نسخه ی 1.4 (آخرین نسخه) سویچ کردند به IronPython 2.0

ویژگی متمایز کننده ی این برنامه از دیگر برنامه های مشابه ، قابلیت برنامه نویسی پایتون در آن است؛ به طوری که از این برنامه می شود به عنوان یک framework برای کارهای محاسباتی و visualization و کار های سنگین با داده ها استفاده کرد. (مثلا با کد پایتون خیلی راحت می توانید در یکی از سلول های صفحه گسترده ، یک button قرار بدهید که مثلا پس از کلیک بر روی ان ، داده هایی را از یک پایگاه داده روی وب خوانده و نتیجه یا گزارشی به صورت یک چارت ۳ بعدی رسم کند)

ویژگی ها:

استفاده از کتابخانه های دات نت و کتابخانه های نوشته شده در دیگر زبان های دات نت

سازگاری با دیگر صفحه گسترده ها (مخصوصا Excel و مخصوصا در قسمت فرمول ها)

استفاده از قدرت زبان پایتون و ماژول های قدرتمند ان برای کارهای پیچیده (مثل numpy) که توسط VBA سخت (اگر نه غیر ممکن) است

یک وب سرور داخلی که اجازه می دهد فایلتان را به اشتراک بگذارید تا دیگران بتوانند ان را دیده و حتی از طریق وب ویرایش کنند

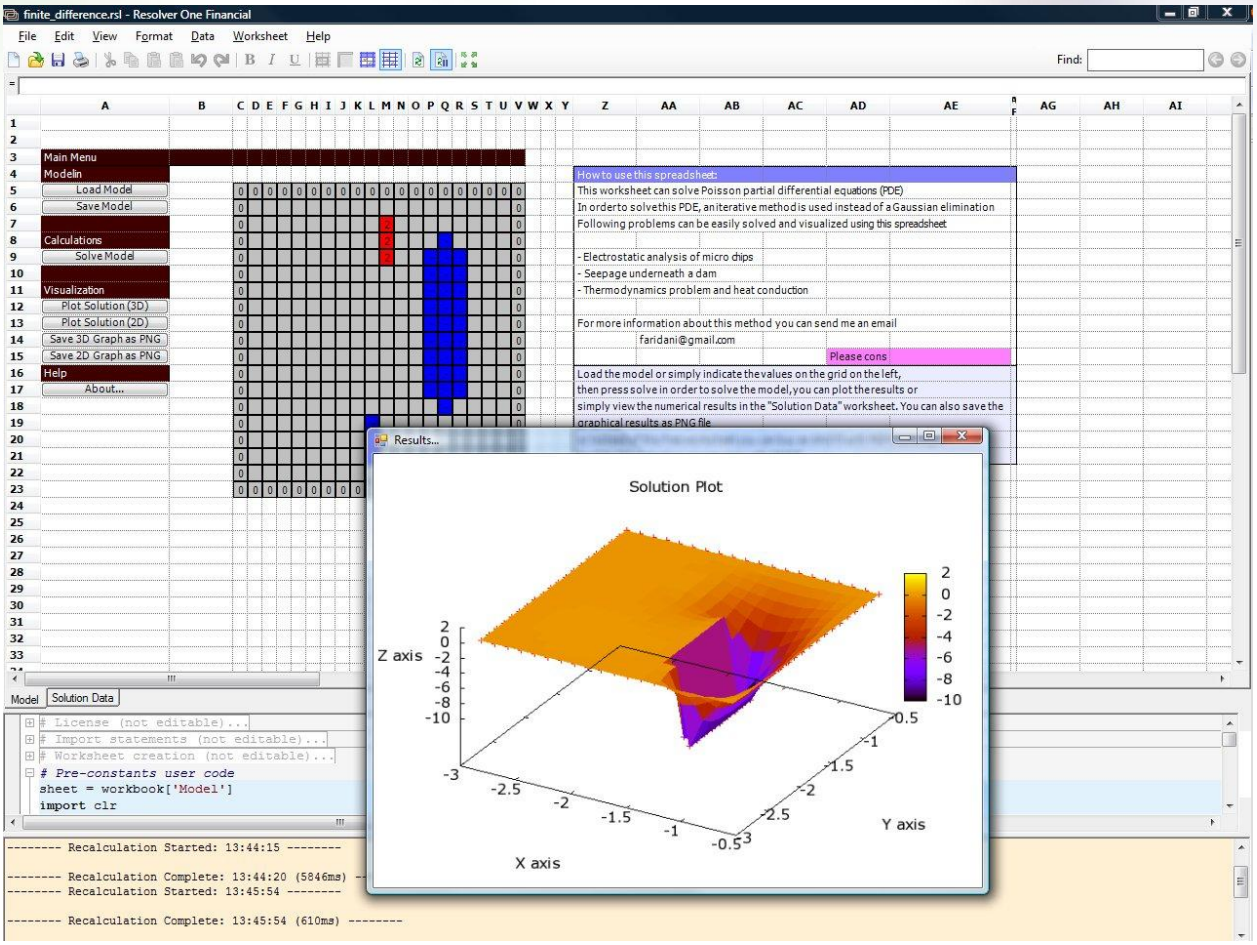
قابلیت اتصال به پایگاه داده های مختلف و دریافت/ارسال داده ها از ان ها

این هم یک مثال پیچیده که با این برنامه تولید شده:

---

[1] IronPython نسخه ای از زبان پایتون است که بر روی پلتفرم دات نت مایکروسافت اجرا می شود

[2] http://www.resolversystems.com




قسمت زیرین برای نوشتن کد پایتون است.

همانطور که می بینید همه چیز قابل تغییر و شخصی سازی است (رنگ پس زمینه ی تک تک سلول ها ، ایجاد ناحیه هایی در صفحه رسم چارت ، استفاده از کتابخانه های دات نت ، قرار دادن اشیایی مثل button روی صفحه ، ....)

برای همکاری با مجله و یا ارسال پیشنهادات و انتقادات خود از ایمیل زیر استفاده کنید:

ezine.barnamenevis@gmail.com