PROGRAMAR

REVISTA PORTUGUESA DE PROGRAMAÇÃO • WWW.PORTUGAL-A-PROGRAMAR.PT

EDIÇÃO #44 - FEVEREIRO 2014

ISSN 1647-0710

CRIAR EXTENSÕES PARA GOOGLE CHROME

A PROGRAMAR

DESBRAVANDO O GOTO

JSF PARTE II

PASCAL OPERATOR OVERLOADING

MITOS DO JQUERY

AVENTURAS C# + SQL

SHAREPOINT

COLUNAS

C#

COMUNIDADES

INTEGRAR O FACEBOOK NUMA APLICAÇÃO WINDOWS PHONE NETPONTO

NO CODE

DAS NOVAS TECNOLOGIAS NAS CRIANÇAS COM NECESSIDADES EDUCATIVAS ESPECIAIS IMPACTO

EDITORIAL

EQUIPA PROGRAMAR

Contacto
revistaprogramar@portugal-a-programar.org

Website
http://www.revista-programar.info

ISSN
1 647-071 0

Try { } catch{ }

"(...)dentro de poucos anos as pessoas terão mais facilidade de comunicar através de máquinas do que pessoalmente, cara a cara."

Joseph Licklider,

In: THE COMPUTER AS A COMMUNICATION DEVICE

É num misto de receio e orgulho que escrevo o editorial desta edição que é a primeira do ano 2014. “Cresci” a ler os editoriais da nossa revista e hoje encontro-me a pensar no que escrever nestas linhas, esperando estar à altura dos “Grandes” que por aqui deixaram a sua marca e a sua experiência.

Numa época em que hoje é o que já não o será amanhã, marcada principalmente pela evolução tecnológica, desejo-vos a todos um ano marcado positivamente pela evolução. Aos amadores e aos profissionais, passando pelos iniciantes, acabando nos especialistas, sem nunca esquecer os curiosos… a todos, que o ano seja de conquistas.

Chegados a mais uma edição da Programar, a nº 44, e aproveitando o facto de o número desta edição, curiosamente, ser uma capicua em que o fim se pode tornar o início, quero recordar aos leitores que esta é uma revista de nós para vós, mas também de nós por vós. Porque sóis vós, leitores que nós movem edição a edição. Por mais e melhor.

E porque nesta nossa equipa “cabe sempre mais um… ”, a todos os que lêem estas linhas deixo o convite… Se tiverem uma ideia, concretizem-na. Se tiverem vontade de partilhar, força!

Porque a partilha de ideias promove o crescimento intelectual e boas equipas provocam mudanças. E porque a Programar é um reflexo da partilha que existe na nossa comunidade do P@P.

Numa altura que a tecnologia é o mote e que o “longe se torna perto”… Sonhem, escrevam, programem e concretizem. E se por ventura, tropeçarem… recomecem de novo. Tornando cada falha um melhoramento pessoal, caminhando em prol da evolução. Porque se todos temos dias de Segmentation Fault, a verdade é que também todos aprendemos que não haverá nunca um try sem um catch.

Porque nós, programadores, somos “gente de ideias”, e não desistimos!

Por isso e por muito mais… Programar… ontem, hoje e amanhã!

Até à próxima edição.

Rita Peres

ÍNDICE

TEMA DE CAPA

A PROGRAMAR

COLUNAS

ANÁLISES

COMUNIDADES

NO CODE

EVENTOS

28 a 28 de Fevereiro XXI SINFO IST Lisboa
11 a 17 de Abril 9º Encontro Nacional de Estudantes de Informática (Universidade de Aveiro)

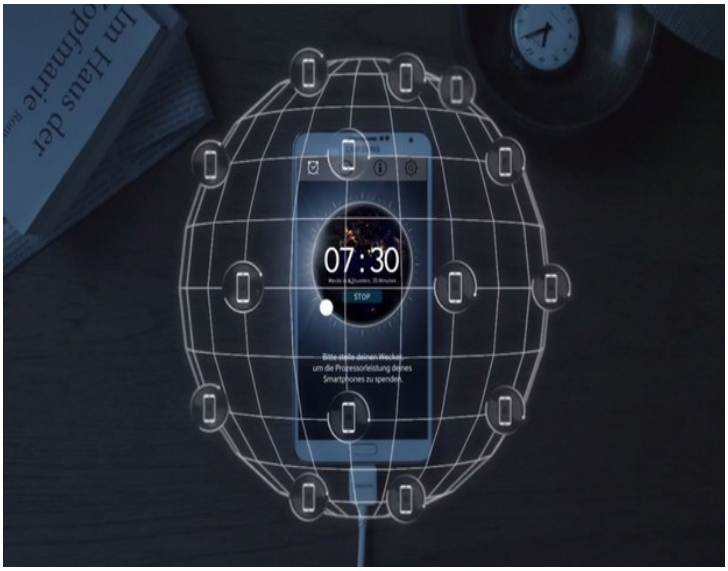
Para mais informações/eventos: http://bit.ly/PAP_Eventos. Divulga os teus eventos para o email eventos@portugal-a-programar.pt

NOTICIAS

O seu telemóvel pode ajudar a curar cancro

Num projeto semelhante ao Folding@Home, também o poder de processamento dos telemóveis pode ser usado para ajudar na investigação da luta contra o cancro.

A app Power Sleep, disponível na Play Store, permite usar o poder de processamento dos smartphones Android e usar essa capacidade na investigação contra o cancro. A app foi criada pela Samsung Áustria e pela Universidade de Viena, e funciona também como um alarme. Assim que o utilizador programa o alarme, a app percebe que o telefone já não vai ser usado e começa a processar os dados necessários para o cálculo de comparações entre sequências de proteínas.



A app está ligada à SIMAP, a Similarity Matrix of Proteins, uma base de dados que é usada por investigadores da genética, bioquímica, biologia molecular e da luta contra o cancro.

O entendimento de como as proteínas são dispostas é um dos primeiros passos nestas investigações, mas requer bastante poder de processamento. Numa altura em que já existem redes de computadores ligados em todo o mundo para esta investigação, acrescentar o poder de processamento dos smartphones vai contribuir para acelerar as descobertas, explica o Quartz.

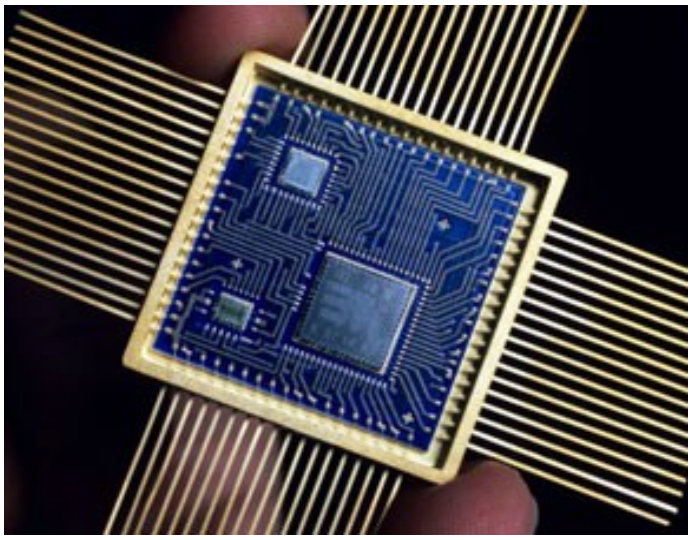
Para os utilizadores mais zelosos de bateria e dos custos de ligações de dados, os responsáveis indicam que a app só usa o dispositivo se este estiver completamente carregado, ligado à corrente elétrica e com o Wi-Fi ligado.

Veja o vídeo preparado pela Samsung.

Fonte: «Exame Informática»

AMD anuncia Opteron A1100 octa-core ARM de 64 bits para servidores

Quem poderia pensar que os pequenos CPUs ARM, outrora relegados para utilizações mais modestas onde o baixo consumo era o factor crítico, viriam a fazer repensar a arquitectura usada nos servidores? No entanto é isso mesmo que está a acontecer, e a AMD quer estar na vanguarda desta ofensiva com o seu Opteron A1100.
O mundo de hoje depende da "cloud" para funcionar, e a sustentar essa mesma cloud estão milhões de servidores espalhados pelo mundo - máquinas que estão a funcionar 24h por dia, 7 dias por semana - e onde os custos energéticos, tanto do seu funcionamento como do seu arrefecimento, acabam por também ser factor decisivo. É por isso que o que importa hoje em dia é a relação potência/consumo e não a "potência pura". Será mais eficiente ter uma dúzia de máquinas menos potentes mas que aqueçam menos e gastem menos energia do que uma só, com potência idêntica, mas que gaste muito mais energia.
Este Opteron A1100 da AMD é um SoC octa-core com cpus ARM Cortex A57 de 64 bits, e que foi expressamente concebido para ser utilizado em servidores: suportando até 128GB de RAM, 8 portas SATA3 (com largura de banda suficiente para utilizar todas ao máximo simultaneamente), 2 portas ethernet de 10Gbps, e PCIe 3.0.



Não me parece difícil imaginar que não vão faltar interessados... e já dou comigo a imaginar quanto mais tempo faltará até que estes chips ARM e seus derivados comecem a surgir em máquinas desktop destinadas ao grande público. Mas... também será de esperar que a Intel esteja a preparar a sua resposta a esta "ameaça" à arquitectura x86 na qual tem dominado nas últimas décadas. De uma forma ou de outra, vai ser interessante ver o impacto dos ARM nesta nova área menos "mobile" a que normalmente os associamos.

Fonte «http://abertoatedemadrugada.com/»

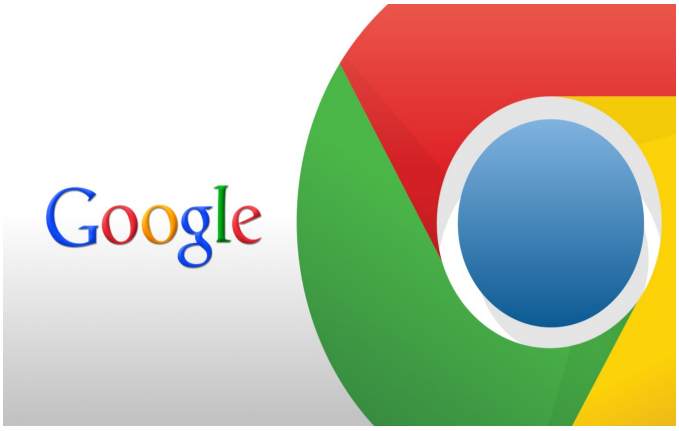
TEMA DE CAPA

Criando Extensões para Google Chrome

Criando extensões para Google Chrome

Atualmente o browser do nosso computador é um dos softwares mais utilizados diariamente, adicionalmente, temos inúmeros plugins que acrescentam funcionalidades extra. Um plugin é um pequeno software desenvolvido para adicionar funcionalidades extra a softwares maiores, como o exemplo dos browsers que utilizamos. Os plugins podem-nos ajudar a automatizar pequenas tarefas no dia-a-dia ou até melhorar a nossa experiência enquanto utilizadores da Web.

Neste artigo que vos trago hoje, vou explicar como podemos desenvolver o nosso próprio plugin para o Google Chrome.



Antes de começarmos deverás ter como requisitos mínimos, os seguintes conhecimentos:

• HTML

• CSS

• JavaScript

Para que possamos começar a desenvolver, basta criar um novo directório no nosso computador, abrir o nosso editor de texto preferido e voilá, estamos prontos!

“Se alguma vez desenvolveste um Website ou aplicação Web, então vais habituar-te rapidamente ao desenvolvimento de plugins”

Declarações

Devemos começar por criar um ficheiro de manifesto, ficheiro este que contém todas as informações do nosso plugin, chamado manifest.json. Este ficheiro é nada mais nada menos do que um ficheiro com o conteúdo em formato JSON.

Neste artigo vamos criar um plugin para que nos informe quais as medidas do browser de uma forma simples.Para isso, criamos um ficheiro manifest.json com o seguinte aspecto:

```
{
"manifest_version": 2,
"name": "Ruller",
"description": "Este plugin diz-nos quais as medi-
das atuais do nosso browser.",
"version": "1.0",
"icons": { "16": "icons/16x16.png", "48":
"icons/48x48.png", "128": "icons/128x128.png" },
!
"browser_action": {
"default_icon": {
"19": "icons/19x19.png",
"38": "icons/38x38.png"
},
"default_title": "Ruller"
}, !
"background": {
"scripts": ["background.js"]
}, !
"content_scripts": [
{
"matches": ["http://*/*", "https://*/*"],
"js": ["jquery.min.js","content.js"]
}
]
} !
```

Passando à explicação:

• manifest_version - É a versão do ficheiro manifest.json que acabamos de criar, está atualmente na versão 2, sendo que ficheiros na versão 1 não serão aceites pelos Google Chrome.

• name - Nome do nosso plugin, o que for definido neste campo será visível publicamente na página dos plugins do Chrome.

• description - Breve descrição do plugin, tal como o nome, a descrição será visível na página dos plugins do Chrome.

• version - Versão atual do nosso plugin, deverá ser definido pelo programador.

• icons - Aqui definimos quais os ficheiros a utilizar para as diversas dimensões dos ícones, os

mesmos devem estar no directório criado anteriormente e deverão ser especificados no manifest.json

• browser_action - Aqui definimos o aspecto do ícone que está visível ao lado da omnibar do

Chrome, especificando quais os ícones a utilizar bem como o título que será visível no tooltip por cima do ícone.

• background - Conjunto de páginas ou scripts a serem utilizados em plano de fundo. Mais para a frente irei regressar a este assunto.

- content_scripts - São os ficheiros que fazem com que o nosso plugin funcione. Devemos ter atenção às dependências do nosso código, neste caso é necessário a framework jQuery, logo tempos de importar primeiro o jQuery. Neste exemplo vamos usar o ficheiro do jQuery e o content.js, ficheiro que contém o código a ser desenvolvido. Aqui podemos acrescentar regras e carregar estes ficheiros apenas em alguns Websites, neste caso carregamos os ficheiros para todos!

Background:

O ficheiro que definimos como background no manifesto, neste caso irá servir como "vigia", estando à espera que o utilizador o chame. Neste ficheiro colocamos o seguinte código:

```
chrome.browserAction.onClicked.addListener
(function(tab)
{
chrome.tabs.getSelected(null, function(tab)
{
chrome.tabs.sendMessage(tab.id, "");
});
});
```

Explicando, estamos a dizer ao nosso background.js que assim que for feito um clique no ícone do Plugin, este envia uma mensagem indicando que o clique foi efetuado. Na mensagem é obrigatório passarmos o id do Tab em que estamos, bem como uma mensagem, que neste caso é uma string vazia.

Listener

Como vimos anteriormente, assim que o utilizador efetua o clique no botão do nosso plugin, é enviada uma mensagem. Mas para quem? É aqui que entra o ficheiro content.js, que definimos no manifest.json. No content.js temos o seguinte código:

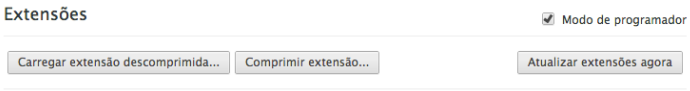
```
chrome.extension.onMessage.addListener(function()
{
var width = $(window).width();
var height = $(window).height();
alert("Largura:" + width + "\nAltura: " +
height);
});
```

O código acima, está à espera que seja enviada uma mensagem. Assim que a mesma é enviada este adiciona um listener, que vai fazer com que ele seja o receptor da mensagem, executando assim a sua função. Neste caso, irá mostrar-nos quais as medidas do nosso browser, no momento em que damos o clique no nosso plugin.

Utilização

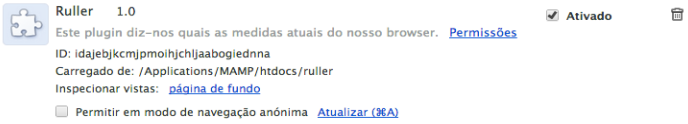
Agora que temos o nosso plugin feito, vamos colocar o mesmo em funcionamento, mãos à obra!

Para isso, basta activarmos o modo de programador do Google Chrome. Abrimos a página dos plugins do Google Chrome e selecionamos a caixa que diz "Modo de programador".



De seguida, clicamos em "Carregar extensão descomprimida" e indicamos onde está o directório criado anteriormente. Assim que estiver concluído, podemos ver o nosso recém criado plugin activado e pronto a utilizar.

Agora que está tudo terminado, podemos testar o nosso plugin!

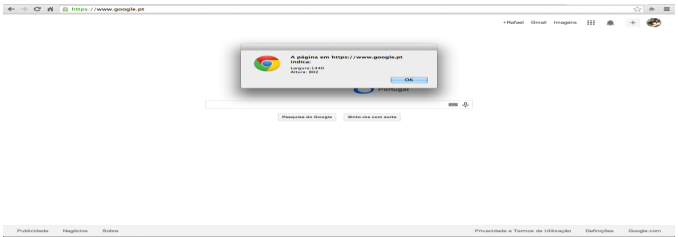


Conclusões

Com este artigo aprendemos a dar os primeiros passos na criação de plugins para o Google Chrome.

Aproveito a ocasião para dar a conhecer um plugin desenvolvido por mim, que tem como objectivo desbloquear o conteúdo social em que somos obrigados a dar Like para que o possamos ver. O plugin chama-se Social Unlocker e podem obter mais informações em: http://www.rafaelalmeida.pt/socialunlocker/

Gostaria de agradecer ao Jorge Paulino pelo convite e pela oportunidade. Espero que tenham gostado e espero que este artigo sirva de ajuda para que possam criar os vossos plugins!



Escrito por Rafael Almeida

Aluno de Eng. Informática da Universidade Fernando Pessoa. Programador Free-Lancer e autor do Plugin Social Unlocker.
http://www.rafaelalmeida.pt

A PROGRAMAR

JSF–Parte 2 – Como Criar um Projeto JSF

Aplicação de Licenças de Utilizador no SharePoint 2013

Funções Anónimas

Pascal–Operator Overloading

Quero fazer uma aplicação simples! E agora? Por onde começo?

Desbravando o goto!

Arduino: const vs #define

Mitos do jQuery

JSF - parte 2 - Como Criar um Projeto JSF

Este segundo artigo da série incidirá sobre o processo de instalação e configuração do ambiente de desenvolvimento para um projeto JSF, que comprovará o sucesso da operação. De seguida, analisaremos a estrutura do mesmo e faremos as primeiras modificações. Em resumo, o artigo divide-se em:

- Instalação do JDK: o SDK do Java
- Instalação do IDE: o NetBeans com plugins Java EE
- Instalação e registo de um servidor aplicacional para Java: o Tomcat
- Configuração e explicação de um projeto JSF

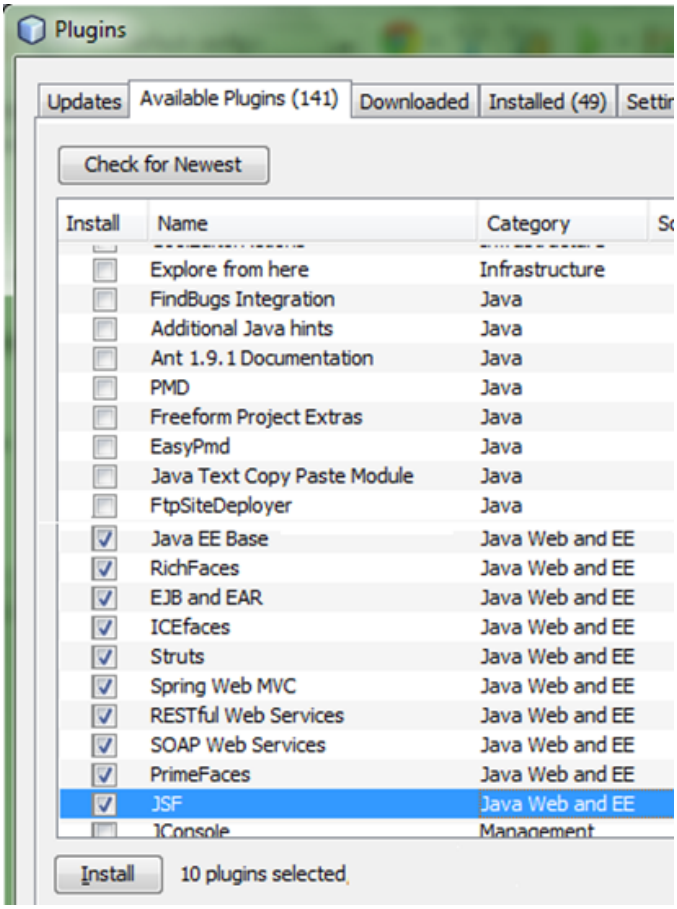
1. *(se já tem o NetBeans e o JDK no sistema então salte para o passo 4)*
   Se ainda não tem nada instalado, pode obter um "2 em (JDK+IDE) no site de downloads da Oracle. Aqui, procure pelo link da versão do JDK com o NetBeans; no ecrã posterior, realize o download para o seu S.O./ arquitetura.

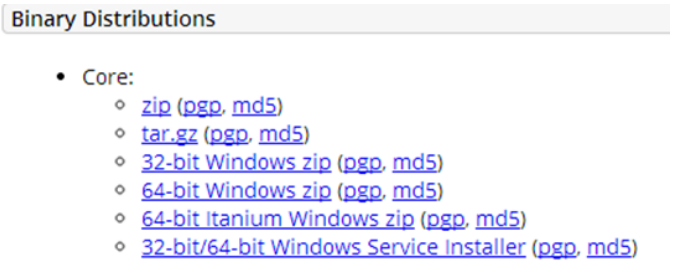
Se, por outro lado, já tem o JDK no sistema, então deve ir diretamente ao site de downloads do NetBeans. Aqui, deve "sacar" a versão do NetBeans "Java EE".

2. Instale a versão que tiver transferido.
3. Abra o NetBeans

Nota: no caso de ter obtido a versão IDE+SDK, há um passo adicional: deve ir a 'Tools' > 'Plugins' > 'Available Plugins', selecionar os plugins da categoria 'Java Web and EE' e instalá-los (em 'Install'). De seguida, deve reiniciar o IDE.

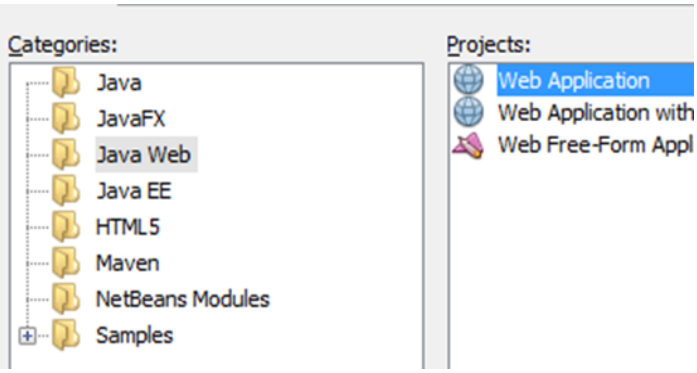
4. *(se já tem o Tomcat no seu PC, salte para o passo 6)*
   Vamos instalar agora o servidor aplicacional (web server).
   Dirija-se à página de downloads do Tomcat e opte pelo download adequado ao seu S.O./arquitetura.





5. Descompacte o ficheiro do servidor para uma pasta à sua escolha.
6. Agora que tem o JDK, o IDE e o servidor aplicacional instalados, está pronto para criar um projeto JSF.

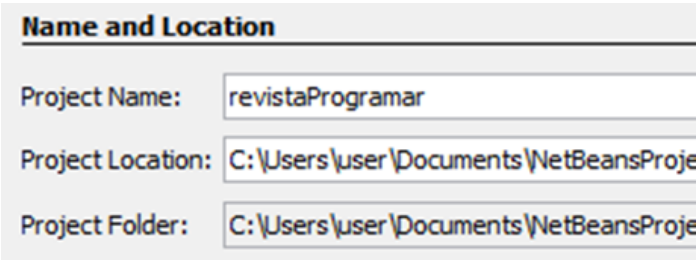
Vamos criar um projeto do zero. Para tal, vá a **'File' > 'New Project'** e selecione **'Java Web' > 'Web Application'**.



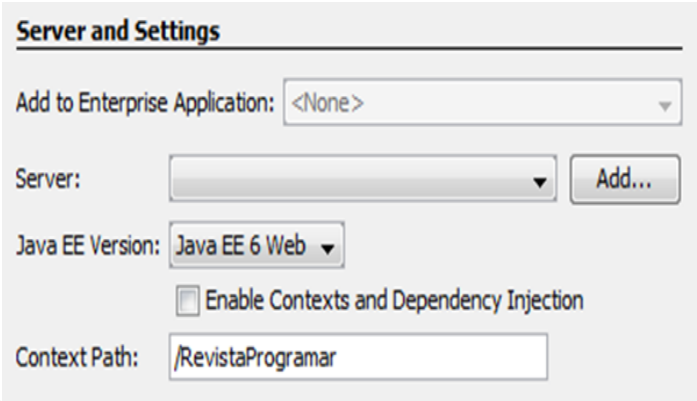
Nota: Com o tipo de projeto criado, quem irá gerir as bibliotecas (dependências) associadas aos projeto será o NetBeans. O Maven é uma alternativa quando as coisas se complicam (nesse caso deve escolher 'Maven' > 'Web Application'). Se preferir, pode criar um projeto com base num "sample project". Para tal, no ecrã de 'New Project', deve selecionar 'Samples' e escolher um projeto que lhe agrade, à partida dentro de 'HTML5', 'Java EE' ou 'Maven'. Em ambos os casos, o resto do tutorial não se aplica.

Clique em 'Next'.

7. Deve agora **dar um nome ao projeto** e fazer 'Next':

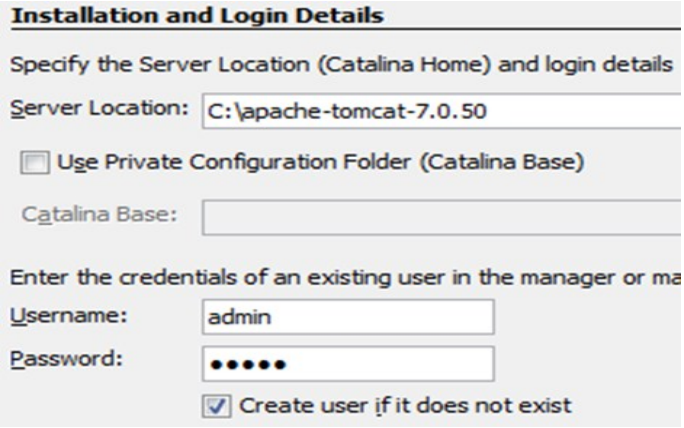


8. *(se já tinha o Tomcat registado no NetBeans salte para o passo 11)*
Como o NetBeans ainda não sabe da existência do Tomcat, devemos agora registá-lo (num próximo projeto ele já estará disponível e isto não será mais necessário). Para tal, **clique em 'Add'**, ao lado de 'Server':



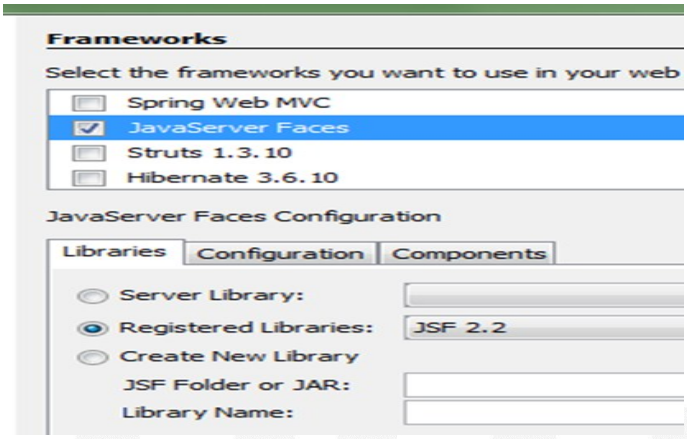
9. Selecione 'Apache Tomcat' e clique em 'Next':

10. Neste ecrã de configuração:
- indique a pasta base para onde descompactou o Tomcat no passo 5 ('Browse');
- insira admin/admin nos campos 'Username' e 'Password';
- clique em 'Finish'.

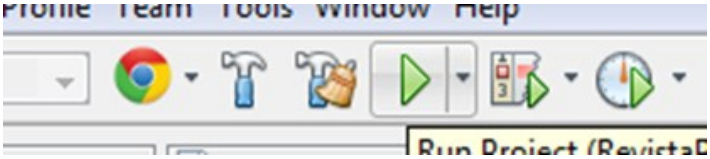


11. Nesta altura já deverá ter o Tomcat presente e selecionado na listagem, pelo que deverá pressionar 'Next':

12. Poderá agora indicar as bibliotecas iniciais (jars) a associar ao projeto. **Selecione o 'JavaServer Faces' (JSF)** e faça 'Finish':



13. Agora já tem tudo configurado, está pronto para correr o projeto. Para tal, **clique no botão de 'Run Project'**:



Voilá… depois de alguns instantes verá uma página do browser com um "Hello from Facelets"!

Vamos então conhecer um pouco melhor as entranhas do projeto. A melhor maneira é olhar para a tab 'Projects'. Esta tab tem a organização lógica do projeto, organizando os ficheiros por categoria e não por como realmente estão no sistema de ficheiros (para isso, use a tab "Files").

"Web Pages": onde encontrará os Facelets do projeto. Em MVC, são a "view". Estes ficheiros representam o markup (com componentes JSF) que será convertido para HTML (são substitutos dos JSP). São materializados em ficheiros ".xhtml".

"Source Packages": o código Java propriamente dito, nomeadamente os "managed beans". Os managed beans são o "controller" em MVC, ou seja, o "backing code" (ou seja API) disponibilizada aos componentes presentes nos Facelets. Por exemplo, poderiamos ter um managed bean chamado "Carrinho.java" com os métodos "getProdutos", "compra", "adicionaProduto", "retiraProduto", etc. Depois, num Facelet, era possível ter a expressão "#{carrinho.produtos}" (o get é implícito e o "p" passa a minúsculo). Esta mesma pasta pode também conter serviços, DAOs, validadores, conversores, classes do domínio, DTOs, entre outros.

"Libraries": onde estão representadas as bibliotecas necessárias ao projeto. Encontrará lá, no mínimo, o JDK e uma implementação da norma JSF. Futuramente, poderá ter uma biblioteca de componentes (ex.: o PrimeFaces) (se o projeto fosse do tipo Maven, esta pasta seria substituida por "Dependencies" e "Java Dependencies").

> ***"O Maven é uma alternativa quando as coisas se complicam (nesse caso deve escolher 'Maven' > 'Web Application').***

"Configuration Files": os ficheiros de configuração do projeto. Por exemplo, o ficheiro "web.xml" que contem algumas configurações de alto nível do projeto (por exemplo, o servlet do JSF). O "context.xml" define em que endereço o projeto é lançado no browser (se o projeto fosse do tipo Maven, esta pasta seria substituida pela "Project Files" e teria um "pom.xml" para gerir as bibliotecas incluidas).

O próximo artigo ensinará a criar um managed bean e a fazer uso do mesmo num Facelet.

AUTOR

Escrito por Luís Soares

Formado em Engenharia Informática e de Computadores no Instituto Superior Técnico (Licenciatura e Mestrado). Sou *web developer, _tendo já colaborado em projetos de telecomunicações e dos _media*. Gosto de linguagens de alto nível, de reutilizar código, de refactoring para simplificar. Gosto de ensinar. Escrevi um livro sobre jQuery (goo.gl/nw2Zb).
Os meus contactos estão em luissoares.com, para qualquer dúvida sobre o artigo ou outra informação.

Aplicação de Licenças de Utilizador no SharePoint 2013

De entre as várias novas funcionalidades incluídas no SharePoint 2013, há uma que injustamente me passou despercebida até recentemente: **User License Enforcement** (Aplicação de Licenças de Utilizador). Digo injustamente porque é uma funcionalidade que já fazia falta há muito tempo e acho que merece mais atenção do que tem recebido.

Até agora, qualquer utilizador com acesso a uma *farm* de SharePoint teria acesso a todas as funcionalidades incluídas na edição instalada do SharePoint (*Standard* ou *Enterprise*), independentemente do nível de CAL (*Client Access License*) que lhe esteja associado. Isto significa que, se uma empresa possui uma *farm* de SharePoint com licença *Enterprise*, qualquer utilizador conseguirá aceder a todas as funcionalidades *Enterprise*, mesmo que lhe tenha sido atribuída uma CAL Standard.

Na maioria das grandes empresas, por questões de otimização de custos, é bastante comum encontrar cenários em que apenas são atribuídas licenças *Enterprise* a alguns utilizadores, ficando todos os outros com licenças Standard. Uma das formas de assegurar que cada utilizador consegue aceder apenas às funcionalidades incluídas no seu nível de licença é ter duas *farms* de SharePoint separadas, uma com edição Standard e outra com a edição *Enterprise*. No entanto, esta não é uma solução aceitável na maioria dos casos.

O SharePoint 2013 resolve esta questão com a nova capacidade de Aplicação de Licenças de Utilizador (ou *User License Enforcement*) que permite mapear licenças com utilizadores específicos ou com grupos de Active Directory. Quando o *User License Enforcement* está ativo, os utilizadores conseguem aceder apenas às funcionalidades incluídas na sua licença, as restantes serão bloqueadas. Quando está inativo, que é a opção por omissão, o SharePoint funciona como nas versões anteriores.

A gestão do User License Enforcement é realizada exclusivamente através de *cmdlets* de PowerShell na SharePoint 2013 Management Shell. Existem oito *cmdlets* que são explicados em detalhe mais abaixo:

- Get-SPUserLicensing
- Enable-SPUserLicensing
- Disable-SPUserLicensing
- Get-SPUserLicense
- Get-SPUserLicenseMapping
- New-SPUserLicenseMapping
- Add-SPUserLicenseMapping
- Remove-SPUserLicenseMapping

Ativar e Desativar o User License Enforcement

Para validar se esta funcionalidade está ativa, deve ser utilizado o *cmdlet* **Get-SPUserLicensing** na consola de gestão do SharePoint 2013. Retornará **true** se a funcionalidade está ativa, e **false** caso contrário (ver Figura 1). Como indicando anteriormente, o User License Enforcement está inativo por omissão.

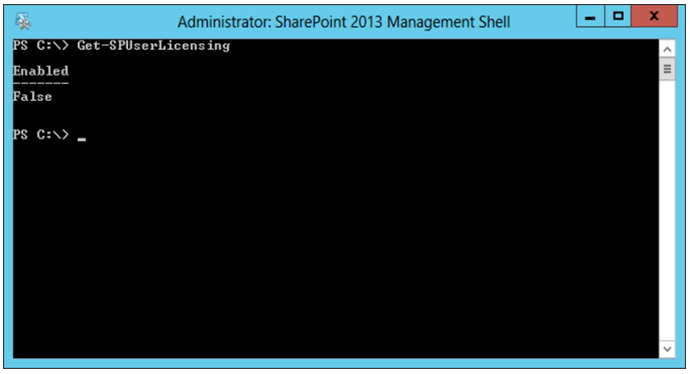


Figura 1 – Utilização do cmdlet Get-SPUserLicensing

Para o ativar, basta executar o *cmdlet* **Enable-SPUserLicensing**. Para o desativar, basta executar o *cmdlet* **Disable-SPUserLicensing**. Nenhum dos *cmdlets* requer parâmetros ou retornará (ver Figura 2).

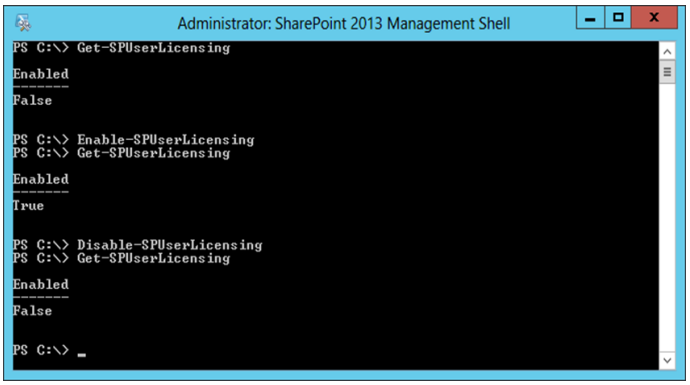


Figura 2 – Utilização dos cmdlets Enable-SPUserLicensing e Disable -SPUserLicensing

Agora que já falámos sobre como ativar e desativar o User License Enforcement, é aconselhável mantê-lo desativado até que esteja corretamente configurado. De outra forma poderá, involuntariamente, bloquear o acesso às funcionalidades *Enterprise* para utilizadores que deveria ter acesso a elas, ou permitir o acesso a funcionalidades que os utilizadores não deveriam ser autorizados a aceder.

Obter Licenças Disponíveis

Para validar quais as licenças que estão disponíveis na sua *farm* de SharePoint, poderá utilizar o *cmdlet* **Get-SPUserLicense** (ver figura 3).

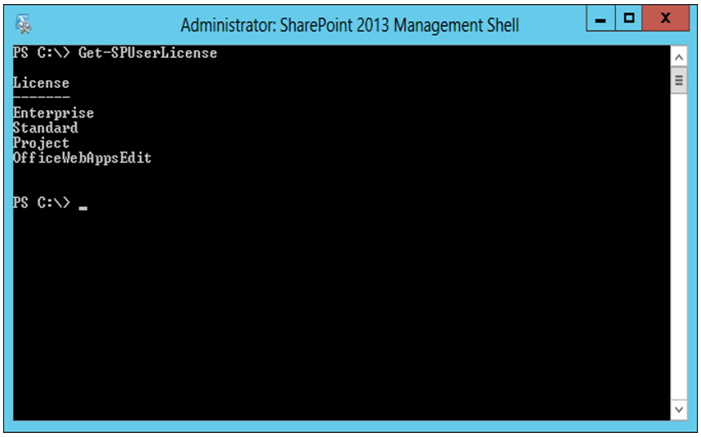


Figura 3 – Utilização do cmdlet Get-SPUserLicense

Existem cinco tipos de licença (embora na minha farm existam apenas quatro):

- Enterprise
- Standard
- Project
- OfficeWebAppsEdit
- Duet

As licenças **Enterprise** e **Standard** correspondem às duas edições do SharePoint Server, que já existem há várias versões do produto. Para consultar as diferenças entre as duas, basta aceder ao TechNet: http://technet.microsoft.com/en-us/library/sharepoint-online-service-description.aspx#bkmk_FeaturesOnPremise.

Quanto aos outros tipos: a licença **Project** é utilizada para controlar o acesso às funcionalidades de Project Server, a licença **OfficeWebAppsEdit** é utilizada para permitir aos utilizadores a edição de documentos nas Office Web Apps, e a licença **Duet** é utilizada para permitir o acesso às funcionalidades do Duet (integração entre SharePoint e SAP).

Mapeamento de Licenças de Utilizador

Mapear utilizadores em licenças é, na prática, mapear *claims* em direitos. Para criar um novo mapeamento é necessário executar o *cmdlet* **New-SPUserLicenseMapping** que pode ser utilizado de várias formas, dependendo do tipo de *claim* que está a ser mapeado.

Pode mapear-se um grupo de Active Directory numa licença, executando o seguinte comando:

```
$mapping = New-SPUserLicenseMapping
-SecurityGroup "Enterprise Users"
-License "Enterprise"
```

É também possível mapear um *Forms-based Role* numa licença usando os parâmetros **–RoleProvider** e **–Role** em vez do parâmetro **–SecurityGroup**:

```
$mapping = New-SPUserLicenseMapping
            -RoleProvider "MyRoleProvider"
            -Role "Enterprise Users"
            -License "Enterprise"
```

Finalmente, é possível mapear uma *claim* numa licença utilizando os parâmetros **–ClaimType**, **-OriginalProvider** e **–Value**, ou apenas o parâmetro **–Claim** com uma referência para um objecto *SPClaim*.

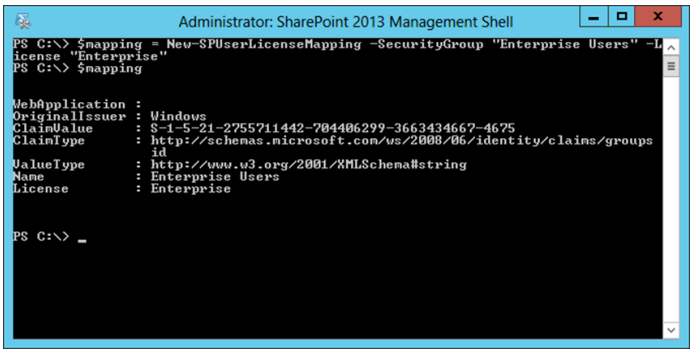


Figura 4 – Utilização do cmdlet New-SPUserLicenseMapping

Aquando da criação do mapeamento, é ainda possível especificar qual a *Web Application* a que o mesmo se aplica, permitindo a configuração de licenças de utilizador por *web application*. Se não for especificada uma *web application*, o mapeamento é aplicado a toda a *farm*.

Uma vez criado o mapeamento, é necessário executar o *cmdlet* **Add-SPUserLicenseMapping** para o adicionar à *farm*:

```
Add-SPUserLicenseMapping –Mapping $mapping
```

Obter Mapeamentos de Licenças de Utilizador

Para verificar quais os mapeamentos configurados na *farm*, pode utilizar-se o *cmdlet* **Get-SPUserLicenseMapping** sem quaisquer parâmetros. Note-se que os mapeamentos definidos ao nível de uma *web application* não serão retornados quando se obtêm os mapeamentos definidos para toda a *farm*.

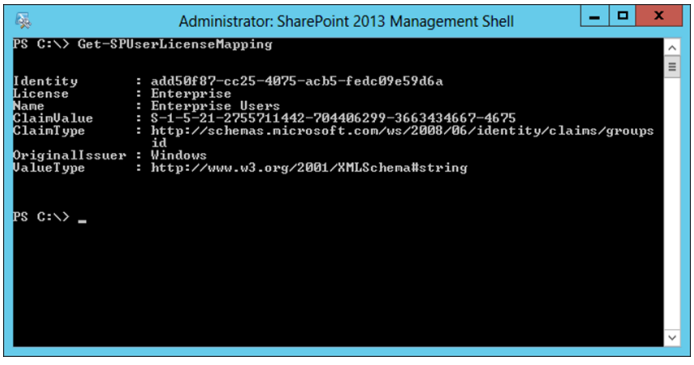


Figura 5 – Utilização do cmdlet Get-SPUserLicenseMapping

Para listar os mapeamentos configurados para uma *web application* específica, basta adicionar o parâmetro **–Web Application** com o respetivo endereço URL.

Remover Mapeamentos de Licenças de Utilizador

Para remover um mapeamento utiliza-se o *cmdlet* **Remove-SPUserLicenseMapping**. O único parâmetro obrigatório é a identidade do mapeamento (um GUID) que é facilmente obtido executando o *cmdlet* **Get-SPUserLicenseMapping**.

Efeitos do User License Enforcement

No que respeita ao User License Enforcement, as duas perguntas mais comuns são:

- Onde é que a licença é aplicada?
- O que acontece quando um utilizador tenta aceder a uma funcionalidade que não está disponível para o seu tipo de licença?

Sobre a primeira pergunta, a licença é aplicada nas seguintes situações:

- Quando é acedida uma página com web parts que requerem uma licença específica (por exemplo a Excel Viewer Web Part ou InfoPath Form Viewer);
- Ao aceder à galeria de web parts para adicionar uma nova web part a uma página;
- Ao selecionar um site template na criação de um novo site;
- Ao tentar editar um documento nas Office Web Apps.

Em relação ao que acontece nestas situações, há duas hipóteses:

- Um utilizador sem a licença exigida não verá os componentes aos quais não tem acesso (web parts e site templates); ou
- O SharePoint negará o acesso ao componente para o qual o utilizador não tem licença, apresentando-lhe uma mensagem que explica a razão pela qual não pode aceder-lhe.

Notas Adicionais

Há algumas notas adicionais que me parece importante lembrar:

- A funcionalidade de User License Enforcement está disponível apenas para ambientes SharePoint 2013 *On Premise*. O SharePoint Online utiliza um modelo de licenciamento por utilizador pelo que não necessita desta funcionalidade para controlar o acesso a funcionalidades específicas.
- O User License Enforcement só funciona com *Web Applications* que usem autenticação baseada em *Claims*.
- Uma vez ativado, terá que ser associada uma licença a todos os utilizadores, até às contas de serviço. Se nos esquecermos de associar uma licença a um utilizador, este será classificado como *Unlicensed* e ser-lhe-á vedado o acesso à maioria das funcionalidades do SharePoint.

> ***"a nova capacidade de Aplicação de Licenças de Utilizador que permite mapear licenças com utilizadores específicos ou com grupos de Active Directory."***

ReferênciasPoderá encontrar informação adicional sobre *cmdlets* PowerShell no TechNet: http://technet.microsoft.com/en-us/library/jj219609.aspx.

AUTOR

Escrito por André Vala

Licenciado e Mestre em Engenharia Informática e de Computadores pelo Instituto Superior Técnico, é actualmente consultor sénior na |create|it| e co-fundador da Comunidade Portuguesa de SharePoint. Autor do blog http://bit.ly/WxtVLz, trabalha com SharePoint desde 2006, altura em que surgiu a primeira versão beta do SharePoint 2007. Tem participado em vários projectos nacionais e internacionais sobre SharePoint, e participa frequentemente como orador em eventos da Microsoft relacionados com o mesmo tema.

Funções Anónimas

Neste artigo vamos falar de funções anónimas, da sua história, a sua usabilidade e ainda mostrar dois exemplos de implementação. Um será em Python, enquanto o segundo será em JavaScript, ou seja, mais orientado para o contexto Web. Mas antes de falarmos em funções anónimas vamos primeiro refletir sobre o que é uma função, que é algo, ligeiramente, complicado de explicar a um leigo em programação.

Já pensaram no que responderiam se lhes perguntássem o que é uma função? A mais simples resposta era fazer uma analogia com a matemática: algo que aceita valores de entrada, transforma-os de alguma maneira e retorna algo no fim. Continuando com a analogia, em matemática, costumamos dar nome às funções, assim como damos enquanto estamos a programar, mais corretamente designado por identificador. Agora imagine que tem uma função, mas que esse identificador não existia. Esta seria uma função que não tinha de estar ligada a qualquer identificador. Aqui temos o princípio básico das funções anónimas.

No presente artigo, vamos tratá-las por funções anónimas, mas são também conhecidas por funções lambda, por terem sido criadas a partir do trabalho de Alonzo Church enquanto desenvolvia o cálculo lambda. Elas são típicas de linguagens funcionais, como o Haskell, Lisp, Ocaml, entre outras. Com o passar dos anos, outras linguagens arranjaram mecanismos de implementar estas funções como por exemplo C#, JavaScript, Python ou ainda Ruby.

Coloca-se uma nova questão: se estamos felizes com as funções comuns e parecem funcionar tão bem, porque devemos usar funções anónimas? E quando devemos usá-las? Dependendo da linguagem com que estamos habituados a programar, já deve ter surgido aquela situação em que o parâmetro de uma função é outra função. Aqui teriamos de declarar uma função no início ou no fim do bloco de código para ser usada uma única vez. Com as funções anónimas, e caso a linguagem suporte closure ganhamos uma maior simplicidade no código, escrevendo estas funções estrategicamente colocadas onde são usadas. São também muito utilizadas na técnica de Currying, para transformar funções de múltiplos parâmetros numa cadeia de funções, onde apenas é passado um de cada vez.

Posto tudo isto, e como cada caso é um caso, vamos meter mãos à obra, primeiro com Python e depois com JavaScript, como já havia sido referido.

Em Python, ao contrário de JavaScript, o uso de funções anónimas é assegurado através da palavra reservada lambda, baseando-se no nome original dado a este tipo de funções, tal como descrito em cima. Este tipo de funções podem ser definidas em qualquer local do código e, como já referido anteriormente, sem um identificador respetivo. São muito utilizadas como complemento às funções de manuseamento de listas e outras estruturas semelhantes, tal como o map, filter e reduce. Para tornar mais claro esta informação segue em baixo a respresentação de duas funções, uma anónima e outra não. A versão utilizada do Python foi a 2.7.

1 – Cálculo do número N da sequência de Fibonacci, utilizando funções normais.

```
def fib_normal(N):
      return N if N<2 else fibs(N-1)+fibs(N-2)

      print fib_normal(10)
```

2 – Cálculo do número N da sequência de Fibonacci, utilizando funções anónimas.

```
fib_anonima = lambda N: N if N<2 else fib_anonima
                          (N-1)+fib_anonima(N-2)
      print fib_anonima(10)
```

No exemplo em cima apresentado podemos ver que em ambos os casos é calculado a soma dos N primeiros elementos da sequência de Fibonacci. No primeiro caso é apresentada a função fib_normal(N) que retorna essa soma, enquanto que no segundo caso o cálculo é feito recorrendo à declaração do lambda, que retorna essa mesma soma na variável fib_anonima.

Em JavaScript, não se torna muito diferente do exemplo anterior. Para declararmos uma função em JavasScript podemos fazer de duas formas:

1 – Declarando uma função com o seu nome, neste caso myFunction:

```
function myFunction() {
    alert("Hello PaP!");
}
```

2 – Ou ainda, usando o operador de função, tornando assim a função anónima:

```
var myFunction = function () {
    alert("Hello PaP!");
}
```

Repare-se que neste último caso a função não tem nome.

A grande diferença entre estes dois tipos, em termos de execução em JavaScript, é que as funções declaradas normalmente são movidas para o início do código e são criadas desde que o programa começa a execução. As que são escritas com o operador apenas são geradas quando chega o momento de as utilizar.

Este mecanismo permite uma diminuição do tempo de compilação, no entanto não permite a avaliação, por parte do compilador, de potenciais erros de execução que possam ocorrer durante a execução do programa.

Vamor ver outro exemplo em que as funções anónimas são habitualmente usadas em JavaScript:

```
var myFunction;
if (x != 0) {
    myFunction = function () {
        alert("!=0")
    }
}
else {
    myFunction = function () {
        alert("= 0")
    }
}
```

Neste exemplo, podemos ver que as instruções a desempenhar pela função representada pela variável myFunction serão diferentes consoante o valor de x.

> *[...] se estamos felizes com as funções comuns e parecem funcionar tão bem, porque devemos usar funções anónimas? E quando devemos usá-las?*

Conclusão:

As funções anónimas devem ser usadas conforme a experiência, a linguagem de desenvolvimento e o objetivo do programador, tendo em conta que, normalmente, as soluções desenvolvidas com recurso a estas são mais elegantes e fáceis de ler. Mais ainda podemos concluir que com o aparecimento do paradigma funcional nas linguagens de programação é possível utilizar todo potencial deste tipo de funções.

> *Agora imagine que tem uma função, mas que esse identifica - dor não existia. Esta seria uma função que não tinha de estar ligada a qualquer identificador. Aqui temos o princípio básico das funções anónimas.*

Esperamos ter agradado os leitores.

AUTOR

Escrito por Cristiano Ramos

Estudante de Engenharia Informática na UBI

cdgramos@live.com.pt

www.cdgramos.com

AUTOR

Escrito por João Silva

Estudante de Engenharia Informática na UBI

j0a0.silva.kle@gmail.com

www.linkedin.com/in/joaopfsilva

Pascal – operator overloading

Várias são as linguagens nas quais podemos fazer *overload* de funções; esta funcionalidade permite que uma função possua várias versões que admitam diferentes conjuntos de argumentos, ficando o compilador encarregue de seleccionar qual dos *overloads* é o correcto aquando da invocação dessa função. Uma das linguagens com essa capacidade é o Object Pascal moderno.

Em Pascal, este tipo de polimorfismo também se aplica aos operadores, os quais podem de igual forma ser *overloaded*.

Na definição da linguagem Pascal segundo a documentação do *Free Pascal*, os *tokens* (palavras que constituem o código do nosso programa) podem pertencer a várias categorias, cada uma delas com funções ou características particulares. Uma dessas categorias é a dos **operadores**, os quais são um símbolo ou conjunto de dois símbolos, com uma função específica, admitindo um ou dois **operandos** e devolvendo um resultado. Os operadores são habitualmente funções com nomes e sintaxe especiais (nomes compostos por símbolos e invocação infixa, por oposição à tradicional invocação prefixa).

Dentro destes operadores, muitos deles podem ser *overloaded*, permitindo ao programador defini-los para novos tipos de operandos, e com comportamentos diferentes do habitual, aumentando assim a expressividade do código. A isto chamamos ***operator overloading***, objecto de estudo do presente artigo.

Todo o código do artigo foi compilado com *Free Pascal Compiler*, versão 2.6.2, em ambiente GNU/Linux (Ubuntu 12.04 LTS).

Overload de operadores aritméticos

Nem todos os operadores podem ser *overloaded*, mas os mais comuns podem: os operadores de atribuição, aritméticos e de comparação.

Vamos partir de casos práticos para entender como se faz *overload* de operadores em Free Pascal. Imagine-se, por exemplo, que pretendemos multiplicar os valores de um *array* de números inteiros, do tipo *Integer*, por um número também ele *Integer*, retornando um novo *array*.

Regra geral, o que nós fazemos é iterar pelos elementos do *array*, aplicando o dito cálculo. O resultado pode ficar no mesmo *array* ou ser atribuído a um novo. Decerto o leitor já teve situações nas quais necessitou de realizar aplicação de cálculos simples a um *array* e tornou-se possivelmente cansativo implementar estes ciclos, mesmo com recurso a funções.

Com a capacidade de *overloading* dos operadores torna-se possível reduzir o ruído do código. Para tal, implementamos numa *unit* os *overloads* que nos forem úteis.

Podemos, então, começar a tirar apontamentos acerca da sintaxe da implementação de *operating overloading*:

```
operator simbolo (operando1 : tipo1; operando2 :
tipo2) resultado : tipo3;
```

Operator é uma palavra reservada comummente não reconhecida pela esmagadora maioria dos editores de texto e IDE's (embora muitos nos permitam definir palavras reservadas adicionais). No entanto, é uma palavra reservada que equivale essencialmente à ***function*** ou ***procedure***. Neste caso, indica ao programa que vai haver um *overload* de um operador pré-existente, representado por ***simbolo***.

Entre parêntesis indicamos os operandos, tal como o fazemos com os argumentos de uma função (afinal de contas, um operador é uma função com dois argumentos). De seguida, vem algo que difere da sintaxe comum: aparece um novo identificador. Em Pascal (Standard e Free Pascal), o resultado de uma função é atribuído a uma variável local com nome igual ao da própria função; no caso de um operador não existe um identificador alfanumérico mas sim um símbolo (o operador), o qual é um nome válido para uma variável, e ao qual não pode ser atribuído um valor. Portanto, temos de criar o identificador que represente o resultado. Especificamos o seu nome, por conseguinte, após a declaração dos operandos, seguido do tipo de dados de *output*.

Vejamos com mais pormenor a seguinte implementação:

```
operator * (n : integer; list : TIntegerAr-
ray) res : TIntegerArray;
var i : word;
begin
   SetLength(res, Length(list));
   for i := Low(list) to High(list) do
      res[i] := n * list[i];
end;
```

Estamos a fazer *overload* ao operador de multiplicação "*", o qual tem dois operandos: *n*, do tipo *integer*, que vai ficar à esquerda do operador, e *list*, do tipo *TIntegerArray*, que vai ficar à direita. O resultado da operação será atribuído ao identificador *res*, e é-nos indicado que o *output* é do tipo *TIntegerArray*.

Não podemos indicar de forma explícita que o tipo de dados é *array of integer*, uma vez que o compilador não o aceita.

Deste modo, temos de criar um novo tipo de dados de modo a que possamos fornecer este tipo na forma de um identificador.

```
type TIntegerArray = array of integer;
```

A partir do momento em que temos este *overload* definido, podemos utilizar o operador "*****" para multiplicar uma variável do tipo *integer* por outra do tipo *TIntegerArray*. No entanto, foi referido que um ficava à direita e outro à esquerda. Vejamos o que acontece se tentarmos compilar o seguinte código (o *overload* do operador está na *unit operover* (de ***Operator Overloading***) uma *unit* própria onde iremos colocar o código).

```
{$mode objfpc}
program artigo44;
uses operover;

var list1 : TIntegerArray = nil;
    list2 : TIntegerArray = nil;
    i : integer;

begin
   (* Código que adiciona valores a "list1" *)

   list2 := list1 * 2;
   for i := Low(list1) to High(list1) do
      writeln('2 * ', list1[i] : 2, ' = ',
                                  list2[i] : 2);
end.
```

O Free Pascal vai indicar o seguinte:

```
artigo44.pas(16,20) Error: Operator is not over-
loaded: "TIntegerArray" * "ShortInt"
```

Nós definimos que o primeiro operando é *integer*, e nós fornecemos-lhe um *TIntegerArray*. Isto acontece porque o *overloading* de operadores é sensível à ordem pela qual recebe os operandos. Vamos, portanto, criar um segundo *overload* que define a ordem inversa dos operandos:

```
operator * (list : TIntegerArray; n : integer)
                          res : TIntegerArray;
begin
   res := n * list;
end;
```

Desta vez, a variável do tipo *TIntegerArray* aparece à esquerda do operador, e a *integer* à direita. Para implementar este *overload* recorremos ao *overload* definido anteriormente. A partir deste momento, podemos multiplicar um *integer* por um *TIntegerArray* em qualquer ordem. Vamos testar o código anterior, e consideremos que *list1* tem os valores {2, 7, 5, 9, 4}; este é o *output* do programa:

```
2 *  2 =  4
2 *  7 = 14
2 *  5 = 10
2 *  9 = 18
2 *  4 =  8
```

Como podemos verificar, *list2* possui os valores de *list1* multiplicados por 2 e pela ordem original. O nosso operador *overloaded* funcionou. Podemos fazer o mesmo para a soma. No entanto, a subtracção e a divisão não têm a propriedade comutativa, ou seja, ***a-b*** não é o mesmo que ***b-a***, e ***a/b*** não é o mesmo que ***b/a***. Nestes casos, talvez seja de vital importância definir bem o *overloading* para cada ordem de operandos. Isto é, *5-list* deverá dar um resultado diferente de *list-5*.

Vamos continuar com o *overload* do operador de adição "**+**".

```
operator + (n : integer; list : TIntegerArray)
                           res : TIntegerArray;
var i : word;
begin
   SetLength(res, Length(list));
   for i := Low(list) to High(list) do
      res[i] := n + list[i];
end;

operator + (list : TIntegerArray; n : integer)
res : TIntegerArray;
begin
   res := n + list;
end;
```

Como se pode verificar, é em tudo igual ao *overload* do operador "*****". De seguida iremos implementar o *overload* do operador de subtracção "**-**". No entanto, para aumentar a versatilidade dos operadores e tornar, por último, esta implementação mais compacta, iremos primeiro implementar o menos unário (o operador "**-**" que indica um valor negativo, como **-4**):

```
operator - (list : TIntegerArray) res :
                                  TIntegerArray;
var i : word;
begin
   SetLength(res, Length(list));
   for i := Low(list) to High(list) do
      res[i] := -list[i];
end;
```

Este operador só tem um operando, à direita. Neste momento podemos definir o *overload* do operador "**-**", de subtracção, de forma bastante compacta:

```
operator - (n : integer; list : TIntegerArray)
                           res : TIntegerArray;
begin
   res := n + (-list);
end;

operator - (list : TIntegerArray; n : integer)
                           res : TIntegerArray;
begin
   res := (-n) + list;
end;
```

Repare-se que recorremos ao operador "**+**" e, no primeiro caso, ao menos unário para definir estes *overloads*. Podíamos ter feito com ciclos, mas esta é uma técnica que permite poupar algumas linhas de código, tornando o código mais legível.

Resta-nos o operador de divisão "*/*", que retorna um número real. Todavia, o nosso *output* tem sido um *array of* ***integer***. Para este caso, necessitaremos de um novo tipo de dados para o *output*:

```
type TRealArray = array of real;
```

PASCAL - OPERATOR OVERLOADING

```
operator / (n : integer; list : TIntegerArray)
res : TRealArray;
var i : word;
begin
   SetLength(res, Length(list));
   for i := Low(list) to High(list) do
      res[i] := n / list[i];
end;

operator / (list : TIntegerArray; n : integer)
                              res : TRealArray;
var i : word;
begin
   SetLength(res, Length(list));
   for i := Low(list) to High(list) do
      res[i] := list[i] / n;
end;
```

Repare-se que recorremos ao operador "**+**" e, no primeiro caso, ao menos unário para definir estes *overloads*. Podíamos ter feito com ciclos, mas esta é uma técnica que permite poupar algumas linhas de código, tornando o código mais legível.

Resta-nos o operador de divisão "*/*", que retorna um número real. Todavia, o nosso *output* tem sido um *array of* ***integer***. Para este caso, necessitaremos de um novo tipo de dados para o *output*:

> *Implemente o* overload *de todos os operadores de comparação (excepto o de igualdade e o de diferença) de forma a comparar dois* arrays. *Caso os* arrays *tenham tamanhos diferentes, deverá retornar* False. *Caso contrário, se todo o par de elementos cumprir a comparação, devolve* True.

Vamos trocar por miúdos: se fizermos **{1,2,3}>={1,4,2}** iremos obter *False*. Os *arrays* têm o mesmo tamanho, mas o segundo par de elementos (o 2 do primeiro *array*, e o 4 do segundo) não cumprem a comparação: **2>=4** é falso, pelo que a comparação dos *arrays* vai retornar *False*.

Não implementamos os operadores de igualdade e de diferença pelo simples motivo de estes, por defeito, já terem a capacidade de comparar *arrays*.

Para implementar estes *overloads* de forma compacta, vamos analisar a seguinte propriedade matemática:

- ***a>b*** é o mesmo que ***b<a***;

Portanto, basta implementar um dos operadores, e o outro será a aplicação do anterior, mas invertido. Em código:

```
operator > (list1, list2 : TIntegerArray)
                                 res : boolean;
var i : word;
begin
   res := Length(list1) = Length(list2);
   if res then
      for i := Low(list1) to High(list1) do
         if  not(list1[i]  >  list2[i])  then
begin
            res := false;
            break;
         end;
end;
```

```
operator < (list1, list2 : TIntegerArray)
                                 res : boolean;
begin
   res := list2 > list1;
end;
```

Portanto, para implementar o *overload* de um operador de comparação com a ordem dos operandos invertidos, recorremos ao *overload* do operador inverso.

O mesmo se aplica aos operadores "**<=**" e "**>=**".

Por fim, os operadores de igualdade ("**=**") e diferença ("**<>**") também podem ser *overloaded*.

Overload do operador *IN*

Desde a versão 2.6.0 do *Free Pascal Compiler* que o operador ***IN*** também pode ser *overloaded*. Este operador verifica, por defeito, se um determinado valor pertence a um *set*. No entanto, ele não tem a capacidade de fazer esta verificação com um *array*.

Desta forma, vamos implementar um *overload* que o permita:

```
operator in (n : integer; list :
                     TIntegerArray) res : boolean;
var elem : integer;
begin
    res := false;
    for elem in list do
        if n = elem then begin
            res := true;
            break;
        end;
end;
```

Overload do operador de atribuição

Para terminar, só falta abordar um operador que também pode ser *overloaded*: o operador de atribuição, "**:=**".

A par do menos unário, este operador só tem um operando, o qual se encontra igualmente do seu lado direito. Para demonstrar como se faz *overloading* deste operador, vamos receber um *set of byte* e transformá-lo num *array of integer*.

```
type TIntegerArray = array of integer;
     TByteSet = set of byte;

operator := (list : TByteSet) res :
                                    TIntegerArray;
var elem : byte;
begin
   res := nil;
   for elem in list do begin
      SetLength(res, Length(res)+1);
      res[High(res)] := elem;
   end;
end;
```

O tipo de dados set implica que os dados fiquem automaticamente por ordem, independentemente da ordem em que os declaramos no código. Isto será visto de seguida.

Conclusão

Consideremos a implementação de todos estes overloads numa unit denominada operover, tal como foi referido ao longo do artigo. Vamos testá-la com o seguinte programa:

```
{$mode objfpc}
program artigo44;
uses operover;

var list1 : TIntegerArray = nil;
    list2 : TIntegerArray = nil;
    i : integer;

begin
   list1 := [2,7,5,9,4];
   list2 := -list1 * 2;
   for i := Low(list1) to High(list1) do
    writeln('2 * ', -list1[i]:2, ' = ', list2
                                        [i]:3);

   writeln('list1 > list2? ':16, list1 >
                                        list2);
   writeln('list1 <= list2? ':16, list1 <=
                                        list2);
   writeln('-4 in list2? ':16, -4 in list2);
end.
```

O output deste programa é o seguinte:

```
2 * -2 =  -4
2 * -4 =  -8
2 * -5 = -10
2 * -7 = -14
2 * -9 = -18
 list1 > list2? TRUE
list1 <= list2? FALSE
   -4 in list2? TRUE
```

Como podemos constatar, apesar do set ter sido fornecido numa ordem aleatória, os dados foram colocados por ordem crescente, isto por causa do modo como funciona o tipo de dados set. Podemos portanto concluir que os operadores estão a funcionar conforme era a nossa intenção.

Desta forma terminamos esta incursão pelo mundo do overloading de operadores em Free Pascal. Basicamente qualquer tipo de dados pode ser fornecido a um operador, desde que seja feito o seu overload. Esta é uma ferramenta que permite tornar a linguagem muito mais expressiva, compacta e até intuitiva.

> ***Com a capacidade de overloading dos operadores torna-se possível reduzir o ruído do código***

> ***Esta funcionalidade de polimorfismo também se aplica aos operadores, os quais, em algumas linguagens, podem de igual forma ser overloaded. [...] o overloading de operadores é sensível à ordem pela qual recebe os operandos.***

AUTOR

Escrito por Igor Nunes

Curioso na área da tecnologia e em especial da programação, tem uma saudável relação com o Object Pascal e é conhecedor das bases de outras linguagens de programação, como Haskell, C, Python e VB.NET. No P@P, é membro da Wiki Team e Moderador Global.

Quero fazer uma aplicação simples! E agora? Por onde começo?

No artigo da edição anterior falamos em árvores de vantagem na linguagem C, em que utilizamos como exemplo prático um *stand* de automóveis. Aproveitando o mote do *stand* de automóveis, nesta edição trago-vos um programa simples acerca da gestão dessas viaturas.

Ao contrário do que é costume, desta vez a linguagem base deste artigo é o *C#* com uma ajuda por parte do *SQL*. Desta vez a escolha recaiu na linguagem *C#* pelo simples motivo de que no meu entender, criar interfaces gráficas é mais rápido e simples fazê-lo usando o C# do que o C.

Ao longo do artigo vamos explicar de forma simples e sem grandes algoritmos como fazer de forma rápida uma pequena aplicação que interaja com uma base de dados.

Este é portanto um artigo mais vocacionado a todos os nossos os leitores que ainda estão a dar os primeiros passos neste tipo de aplicação, até porque a ideia deste artigo surgiu das dificuldades que eu própria senti quando recentemente iniciei numa nova etapa.

Vamos por partes então… o que precisamos para este artigo?

- Do SQL Server, que para este artigo foi utilizada a versão Express que pode ser obtida gratuitamente no site da Microsoft (http://www.microsoft.com/en-us/download/details.aspx?id=29062).
- Do No-IP, que para este artigo foi utilizado a sua versão Free que pode ser obtido em (http://www.noip.com/)
- Do Visual Studio (neste artigo foi utilizado a versão VS 2010)

Se poderíamos ter utilizado outros softwares para este artigo? Sim, de facto poderíamos, e o leitor caso queira experimentar por si implementar o nosso exemplo pode utilizar outros equivalentes que prefira.

1ª Fase – No-IP e o IP Fixo

Agora que já temos tudo o que precisamos, vamos começar por registarmo-nos no No-IP para termos uma conta *free* e vamos instalar o programa no computador que queremos que tenha a nossa Base de Dados. Para quem não conhece, o No-IP traduz o nosso IP dinâmico para um IP Estático que pode ser acedido em qualquer parte. Isto porque a maior parte das nossas ligações à internet quando nos ligamos à rede nunca temos o mesmo IP Público, este IP altera-se quase sempre que tentamos estabelecer uma nova ligação. O No-IP é uma forma simples de termos sempre o mesmo IP público, isto porque este software faz a tradução entre o IP público da ligação que temos no momento, para o host que criamos na nossa conta free.

Quando executamos o No-IP colocamos o email usado e a password escolhida.

Ilustração 1 - Ecrã de Login No-IP

Seguidamente escolhemos o host que criamos (neste artigo foi usado como host o artigoprogramar.no-ip.biz como mostra a imagem seguinte)

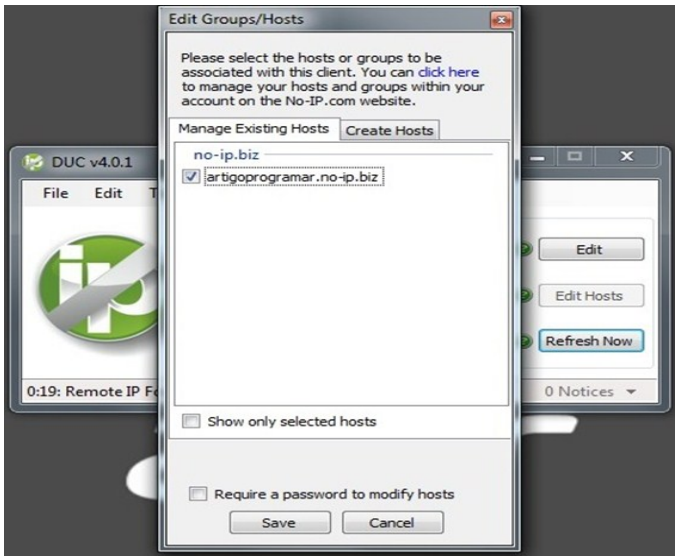


Ilustração 2 - Escolha de Host No-IP

Ao clicarmos em "Save", o No-IP actualiza-se e como mostra a figura seguinte passa a verificar periodicamente qual o IP público que temos no momento, para que o host que escolhemos anteriormente.

Ilustração 3 - Verificação de IP público do No-IP

Agora que já temos o No-IP activo precisamos de o nosso router saiba que os pedidos que receber de "fora" da sua rede têm que ser enviados para o computador que tem instalado a nossa base de dados. A forma mais rápida de fazermos isso é dar um IP Fixo ao computador em questão.

Neste caso, indo a *Painel de Controlo – Ligações de Rede – Propriedades* e depois *Propriedades TCP/IPv4* demos por exemplo o ip 192.168.2.100.

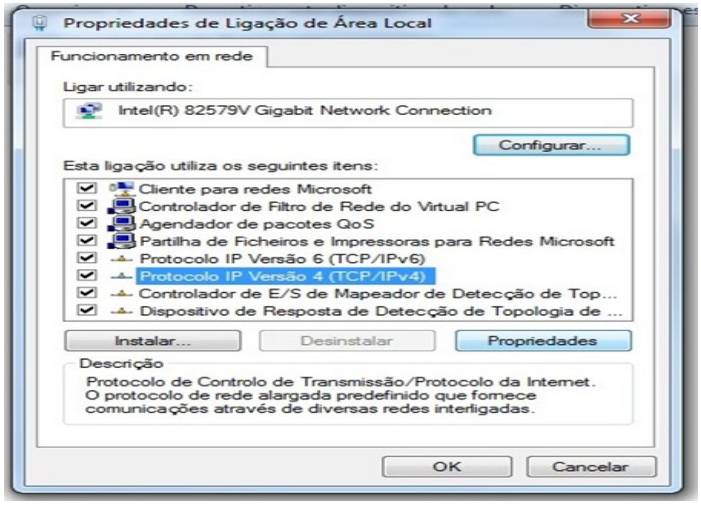


Ilustração 4 - Propriedade de Área Local do Computador da BD

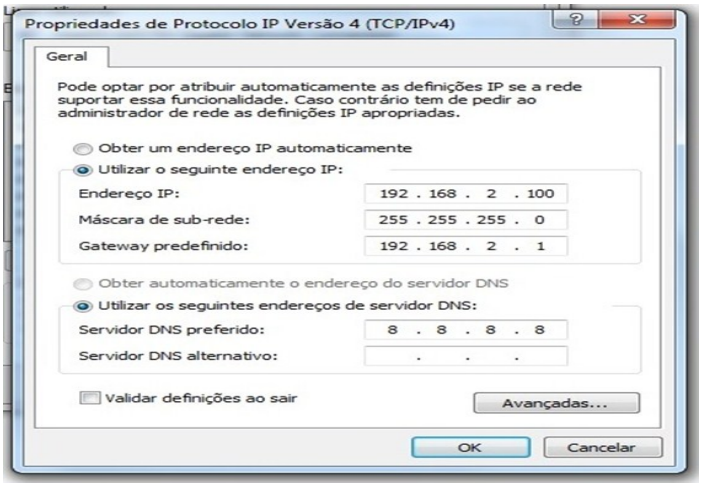


Ilustração 5 - Dar um IP Fixo na nossa rede interna ao computador da BD

Ou seja neste momento na nossa rede privada, o computador onde iremos instalar o SQL Server irá sempre responder no IP 192.168.2.100.

Nota: *Caso apenas quiséssemos implementar este exemplo na nossa rede de casa por exemplo, não precisaríamos de utilizar o No-IP. Isto porque dentro da nossa rede interna, os computadores iriam "reconhecer-se" entre si.*

2ª Fase – SQL SERVER

Nesta fase vamos então instalar a nossa Base de Dados. Após fazermos o download da versão indicada ao nosso sistema operativo, basta executarmos o assistente de instalação e seguirmos os passos que nos vão sendo mostrados.

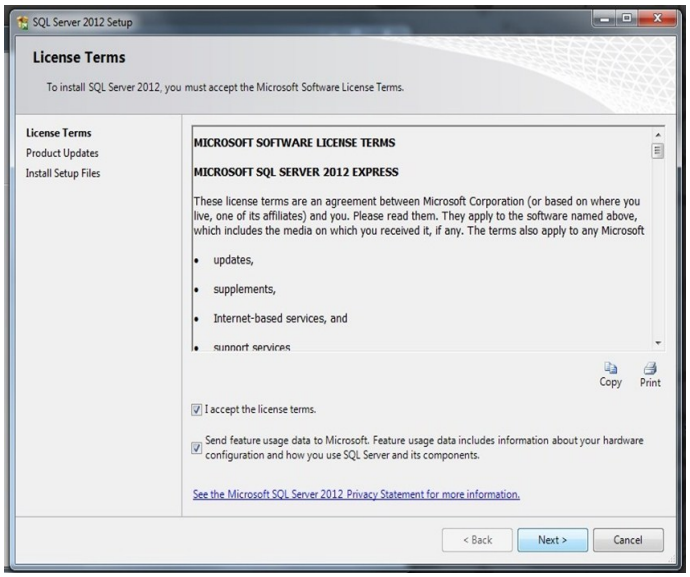


Ilustração 6 - Aceitar as condições da Versão EXPRESS do SQL SERVER

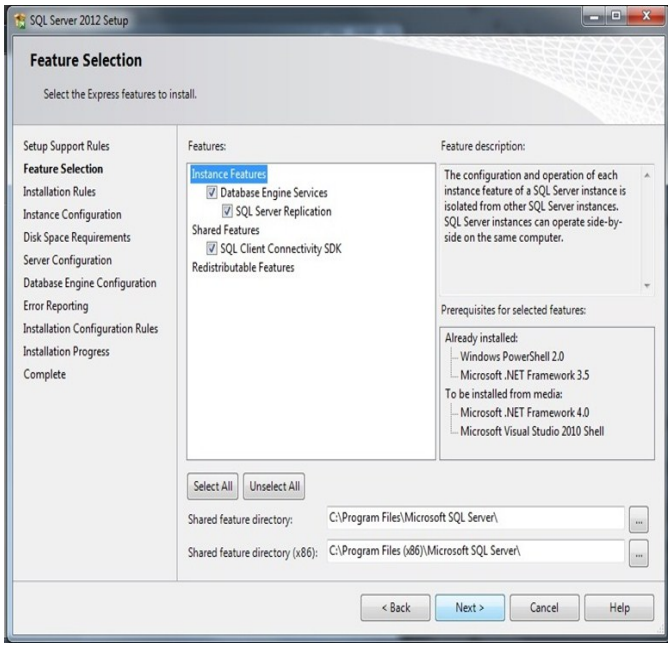


Ilustração 7 - Opções de Instalação

QUERO FAZER UMA APLICAÇÃO SIMPLES! E AGORA? POR ONDE COMEÇO?

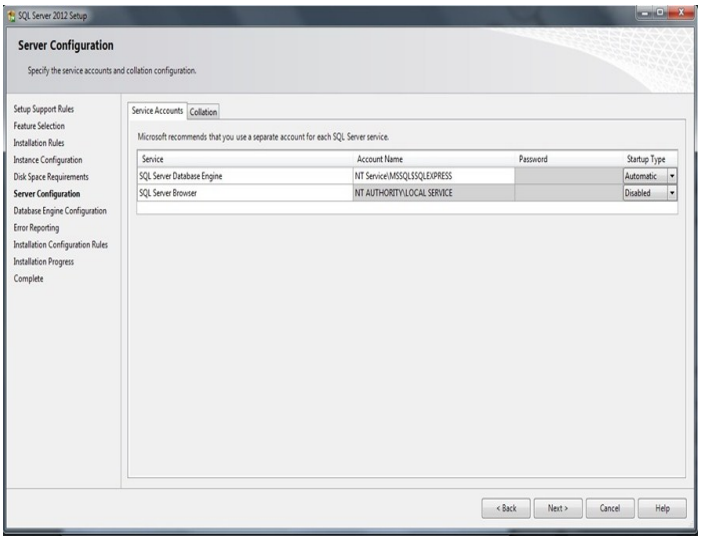


Ilustração 8 - Opções de Instalação - Continuação

Ora heis que neste momento chegámos a meio da nossa instalação da Base de Dados. Ser-nos-á pedido que confirmemos como queremos aceder à nossa BD. Neste ecrã devemos escolher a opção *Mixed Mode* que nós irá permitir aceder à BD quer por meio da autenticação da sessão do Windows quer por meio da autenticação no *SQL Server*. Neste momento ser-nos-á pedido para escolhermos uma password de acesso para o login sa (*System Administrator*).

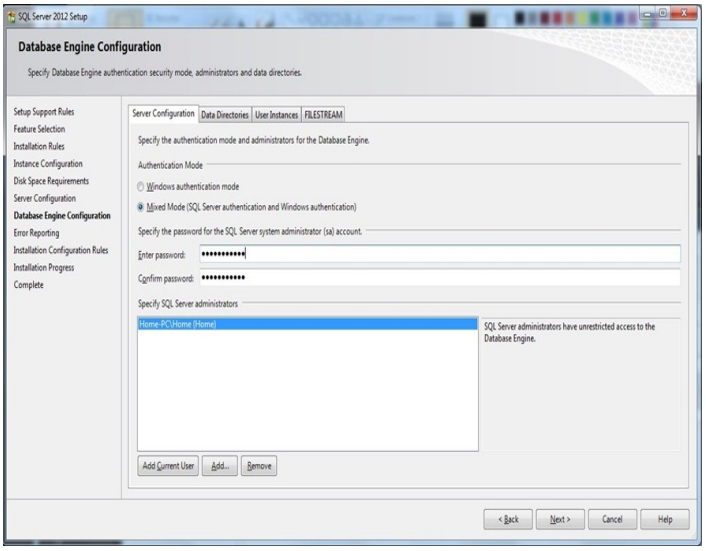


Ilustração 9 - Escolher a password de acesso à BD

Convém não esquecermos qual a password escolhida para que mais tarde consigamos aceder facilmente à base de dados e podermos de forma rápida e eficaz alterarmos a informação nela guardada.

Para este artigo os dados utilizados foram:

Login: sa

Password: p@ppassword

Mais à frente neste artigo voltaremos a utilizar estes dados.

Por fim chegaremos ao menu que nos indica que instalamos o SQL Server com sucesso como mostra a imagem seguinte.

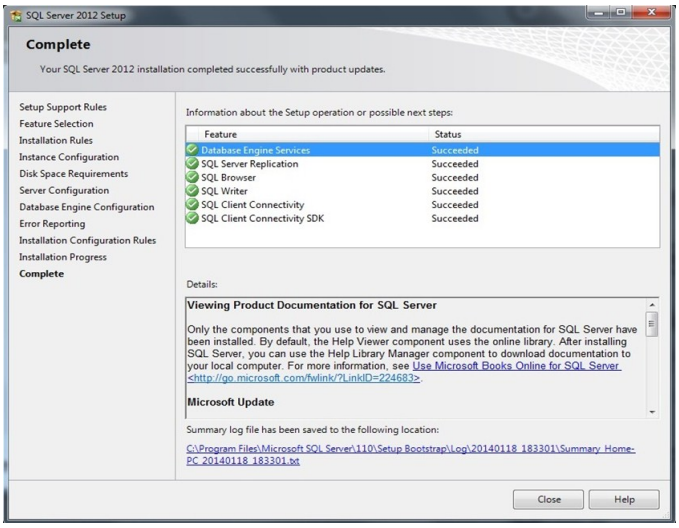


Ilustração 10 - Menu de confirmação de Instalação

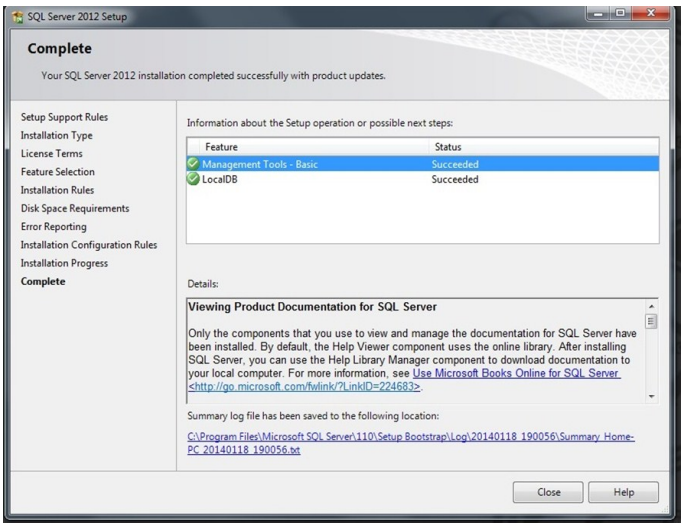


Ilustração 11 - Menu de confirmação de Instalação

Neste momento após termos instalado o SQL Server (neste caso a BD e o *Management Studio SQL Server*), se formos aos programas instalados já poderemos ver que nos aparecem os novos programas como mostra a figura seguinte.

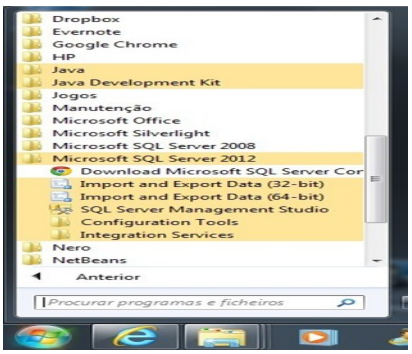


Ilustração 12 - Instalação bem sucedida SQL Server

Neste momento estamos quase prontos para começar a criar a Base de Dados que vai suportar este nosso artigo.

Mas antes de começarmos convém tomar em conta alguns pormenores que apesar de serem pequenos, têm uma importância extrema ao bom funcionamento que pretendemos.

O SQL Server "responde" em duas portas especificas, a porta 1433 (TCP) e a porta 1434 (UDP). Contudo por defeito, quando o instalamos pela primeira vez essa opção vem desactivada. Convém portanto activarmos essa opção antes de prosseguir. Para activar basta ir ao *Painel de Controlo – Programas – Microsoft SQL Server 2012 – Configuration Tools* e abrir o *SQL Server Configuration Manager*.

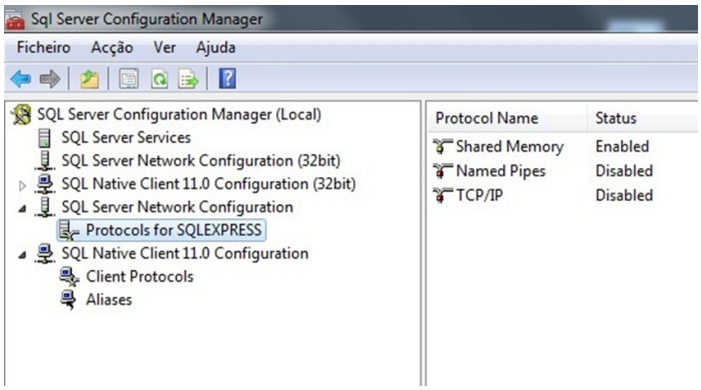


Ilustração 13 - Menu SQL Server Configuration Manager

Como podemos ver pela figura anterior o protocolo TCP/IP está desactivado, devemos então activa-lo (isto é, passa-lo de *Disabled* para *Enable*).

Posto isto vamos então executar o *SQL Server Management Studio* que nos irá permitir facilmente criar a nossa BD.

Executando-o, irá aparecer-nos o seguinte ecrã.

Ilustração 14 - Menu de entrada SQL Server

Carregando em "Connect" entraremos no SQL Server. No lado esquerdo do ecrã é visível o "Object Explorer" que nos irá ajudar a atingir o nosso objectivo…

Para criar uma nova base de dados, clicamos com o botão do lado direito do rato em cima de "Databases" e escolhemos "New Database" como é mostrado na figura seguinte.

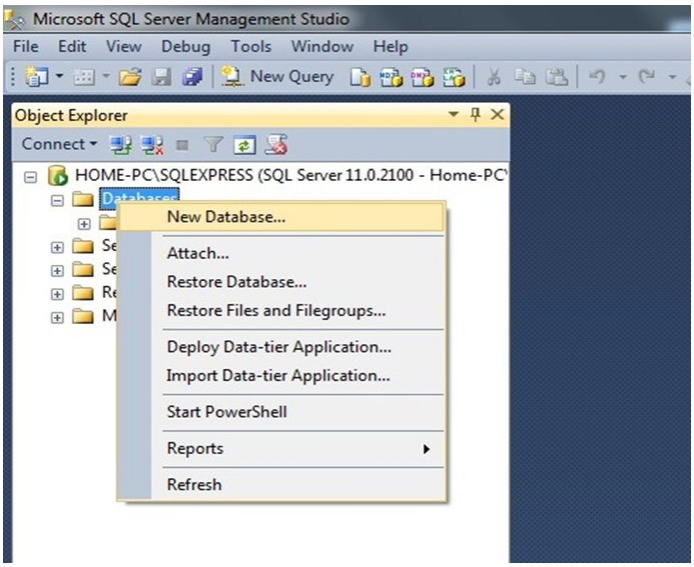


Ilustração 15 - Criar uma nova BD

Aparecerá um novo menu onde devemos colocar o nome da BD assim como as características que queremos que a mesma tenha. Com é possível ver na imagem seguinte demos o nome de PAP à base de dados e deixámos as definições de defeito.

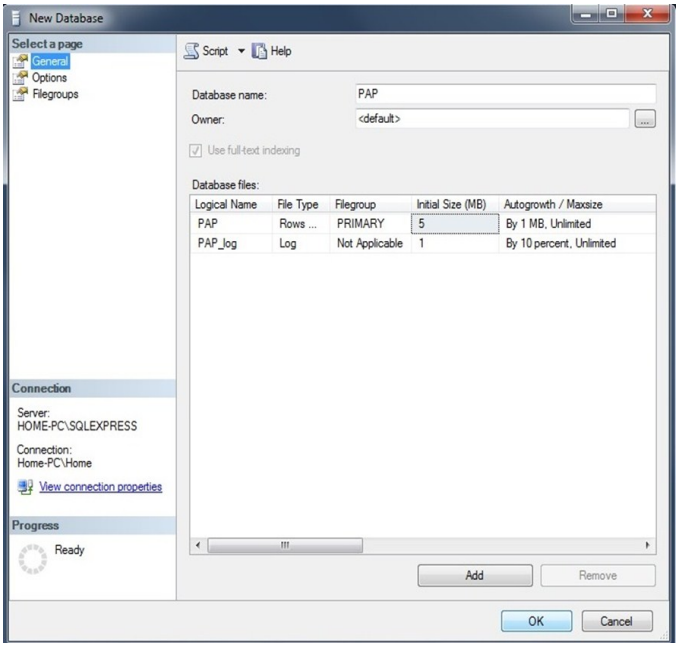


Ilustração 16 - Criar nova BD

Agora que já temos a base de dados precisamos de lhe adicionar as tabelas correspondentes.

De forma análoga quando expandimos a nossa PAP (a base de dados que acabamos de criar), são nos mostradas as sub pastas que compõe a mesma. Clicando mais uma vez com o

botão do lado direito em cima da subpasta Tables escolhemos a opção “New Table” como ilustra a figura seguinte.

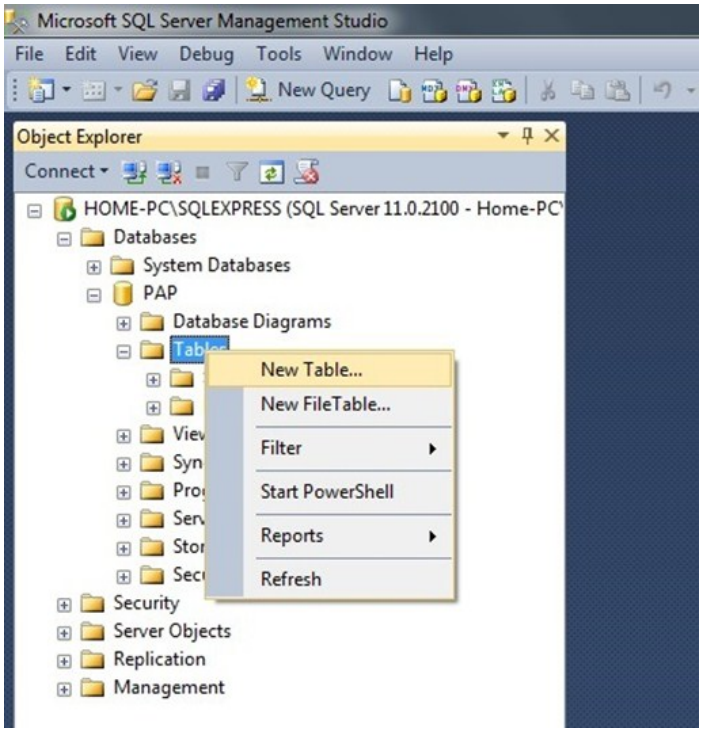


Ilustração 17 - Criar nova tab

Agora basta-nos inserir os campos na mesma e escolher qual o tipo de campos que queremos guarda nessa tabela da BD. Por fim ao guardarmos a tabela que acabamos de criar damos-lhe o nome que escolhermos, a figura seguinte mostra como foi criada a tabela “*Carro*” que irá ser usada no nosso projecto para este artigo.

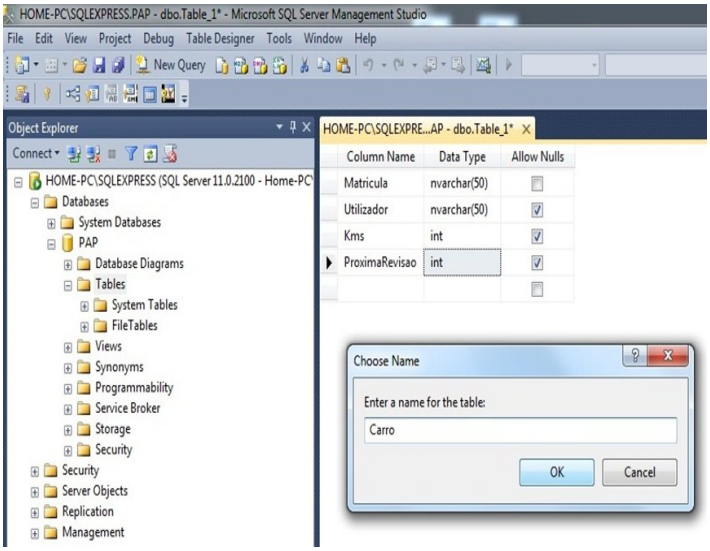


Ilustração 18 - Criação da Tabela Carro

Ao irmos criando as diferentes tabelas que compõem a BD as alterações vão-se tornando visíveis, como mostra a figura seguinte.

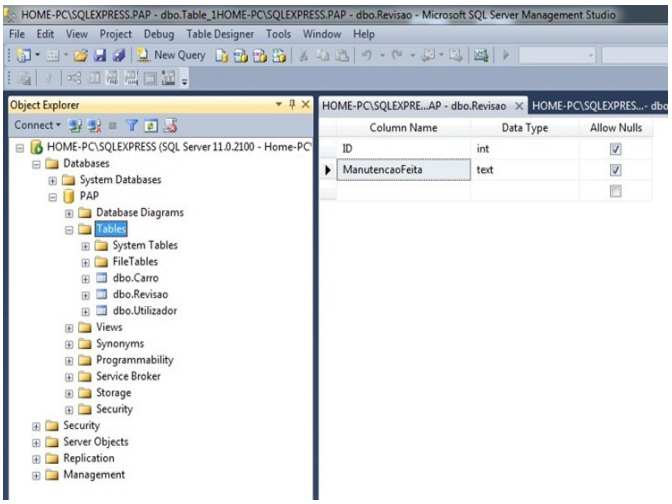


Ilustração 19 - Criação de várias tabelas BD

Neste exemplo consideramos quatro tabelas, que nos permitirão guardar toda a informação dos nossos veículos.

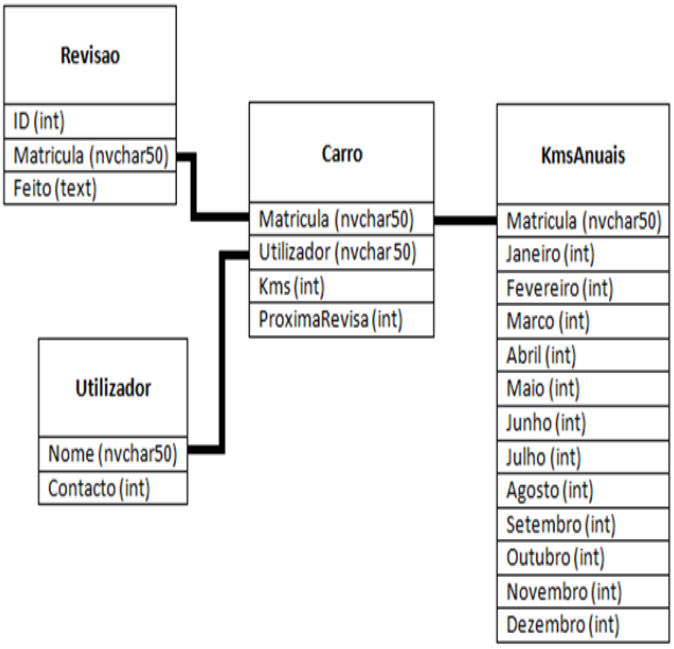


Ilustração 20 - Tabelas Usadas no Exemplo

Antes de passarmos à última fase deste artigo vamos voltar a fazer o ponto de situação.

Neste momento temos o No-IP que nos permite ter um ip fixo público, temos o SQL Server com a base de dados PAP que criamos, em que essa mesma base de dados tem 4 tabelas (Carro, Utilizador, Revisao e KmsAnuais).

Agora que já temos tudo, podemos passar então à nossa aplicação propriamente dita.

3 ª Fase – A aplicação

Nesta fase, executamos o *Visual Studio* e criamos um novo projecto, neste caso criamos um projecto do tipo *Windows Forms Application* que é já bastante conhecido.

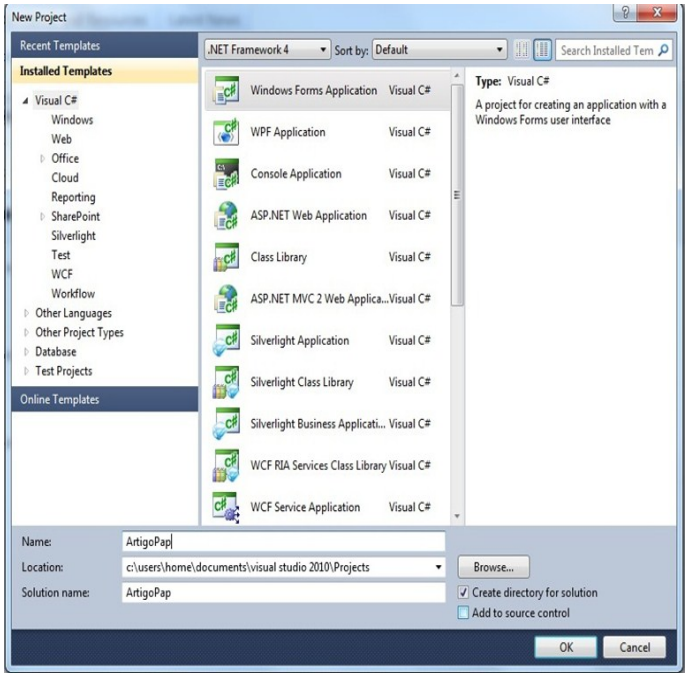


Ilustração 21 - Criação do Projecto Windows Forms

Nota: *Para o bom funcionamento deste projecto deve estar instalado também a .NET Framework 4.0 ou superior.*

Para ser mais simples explicar a aplicação em si, foram colocado diversos botões cada um contendo um código especifico de forma a que todos os leitores pudessem facilmente observar as alterações de que vamos falar neste artigo.

Então a que nos propormos executar no projecto ilustra este artigo?

1. Inserir um utilizador no sistema
2. Inserir um carro no sistema
3. Actualizar os Kms Mensais feitos pelo carro
4. Saber quais os carros que estão próximos de ir à próxima revisão.
5. Gerir as revisões feitas pelos carros
6. Gerir os Kms feitos por essas mesmas viaturas

Como foi referido para melhor ilustrar o código de cada parte destes pressupostos, foram criados diversos botões, cada um com a sua funcionalidade específica.

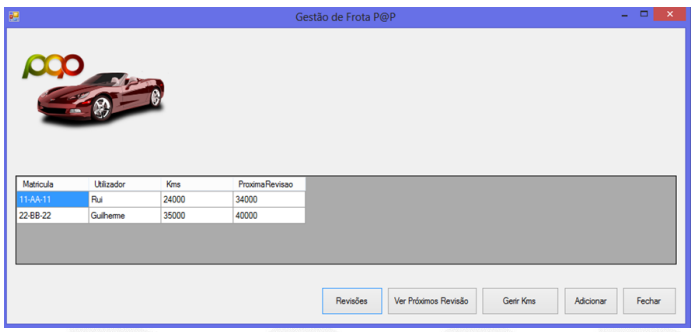


Ilustração 22 - Menu entrada da Aplicação

A imagem anterior mostra os carros que estão na base de dados assim como o utilizador e os Kms que faltam para que seja necessária a próxima inspecção desse veiculo.

Para isso foi criada a função mostraCarros:

```
private void mostraCarros() {


            //Separador Tabela Litros
        SqlConnection conexao = new SqlConnection();
            conexao.ConnectionString = @"Data
              Source=artigoprogramar.no-ip.biz;user
  id=sa;password=p@ppassword;Trusted_Connection=no;
               database=PAP;connection timeout=30";
            conexao.Open();

            string sql2 = "select * from Carro";
            SqlCommand cmd2 = new SqlCommand(sql2,
                                              conexao);
            cmd2.Connection = conexao;
            cmd2.CommandText = sql2;

            // cria o dataadapter...
            SqlDataAdapter adapter2 = new
                                    SqlDataAdapter();
            adapter2.SelectCommand = cmd2;
            try
            {
                // preenche o dataset...
                System.Data.DataSet dataSet2 = new
                              System.Data.DataSet();
                adapter2.Fill(dataSet2);

                dataGridView1.DataSource =
                                              dataSet2;
                dataGridView1.DataMember =
                      dataSet2.Tables[0].TableName;

            }
            catch

            {
                // MessageBox.Show("Erro !");
            }
            conexao.Close();

        }
```

Vamos agora tomar um pouco de atenção à função anterior.

Lembram-se do nome do host que criamos com o No-IP e do login e da password que escolhemos quando instalamos a BD? É nesta ligação que iremos usar esses dados.

```
SqlConnection conexao = new SqlConnection();
conexao.ConnectionString = @"Data
              Source=artigoprogramar.no-ip.biz;user
  id=sa;password=p@ppassword;Trusted_Connection=no;
               database=PAP;connection timeout=30";
conexao.Open();
```

Traduzindo: "Informamos" o nosso programa que a BD a usar é a que está no endereço artigoprogramar.no-ip.org, em que faremos o login com o utilizador "sa", com a password "p@ppassword", usando a base de dados "PAP".

Seguidamente dizemos-lhe para abrir a conexão à base de dados e através da instrução SQL, pedimos os dados que nos interessam da tabela que queremos.

Neste caso string sql2 = "select * from Carro";, ou seja, queremos tudo (*) o que está presente na tabela Carro.

Guardamos essa mesma informação num SQLAdapter e depois escolhemos usando um datagridview1, mostramos essa informação aos nossos utilizadores finais. Ou seja, o datagridview usa como fonte de dados, a informação presente no SQLAdapter que criámos.

Comecemos então pelo botão "Adicionar":

```
private void adicionarbutton_Click(object sender,
                                   EventArgs e)
        {
            Adicionar add = new Adicionar();
            add.ShowDialog();
            mostraCarros();
        }
```

Ora como é visível, o botão adicionar instancia a classe Adicionar enviando-nos para a interface gráfica dessa mesma classe. Assim que fizermos as alterações nesse submenu, ao voltarmos ao menu inicial, é novamente chamada a função mostraCarros que vai actualizar novamente a listagem a ser mostrada.

Quando entramos na Classe Adicionar vemos o seguinte Layout:

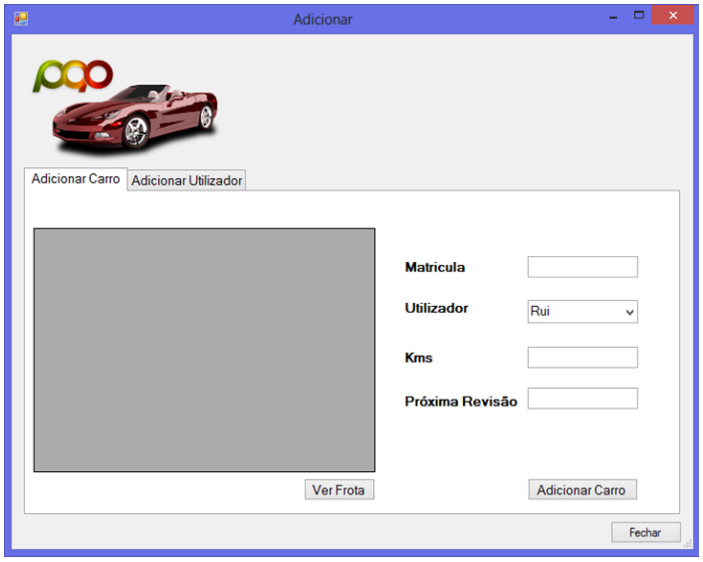


Ilustração 23 - Classe Adicionar

Temos um *TabControl* com duas *TabPages* uma para "Adicionar Carro" e outra para "Adicionar Utilizador".

Comecemos pela *TabPage* "Adicionar Carro", para adicionar um carro temos que inserir a matricula do mesmo, o utilizador do carro, assim como os Kms que o veiculo tem e com quantos Kms deverá essa viatura ir à próxima revisão.

Neste exemplo "obrigamos" o utilizador a escolher um nome de um utilizador que se encontre nos dados da *ComboBox*, ou seja, quando adicionamos um carro, somos obrigados a escolher um utilizador que esteja já inserido no sistema.

Mas como podemos "dizer" à nossa *ComboBox* qual o conjunto de dados que a mesma deve apresentar?

Uma das formas mais simples é usar *DataBindings*. E como?

Se clicarmos na *ComboBox* com o botão direito do rato, irá aparecer a imagem seguinte.

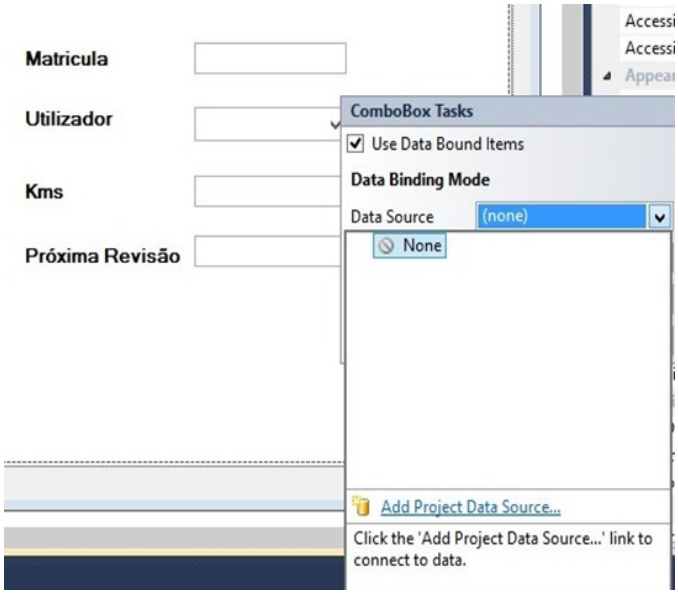


Ilustração 24 - Criação de DataBinding

Escolhendo a opção "*Use Data Bound Items*", poderemos adicionar uma nova "*Data Source*" ao nosso projecto. Na figura seguinte são visíveis as diversas opções que podemos adquirir assim como a que foi escolhida para este exemplo.

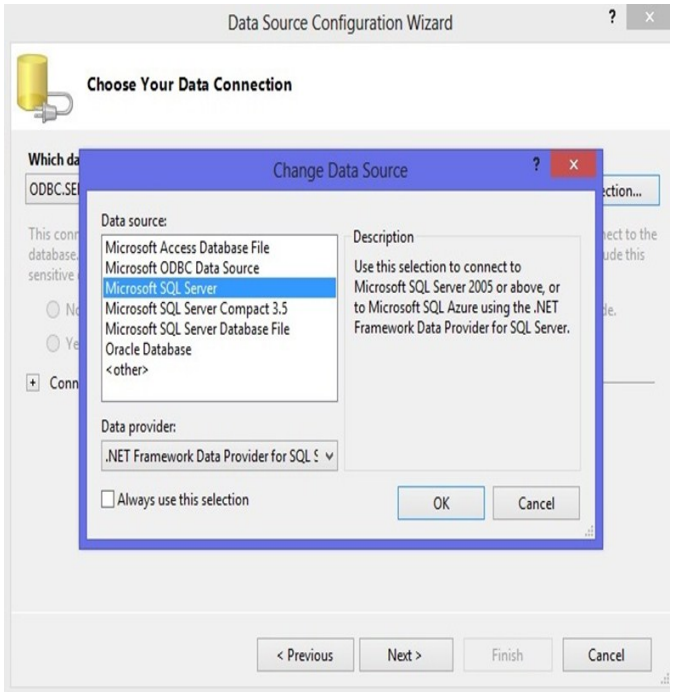


Ilustração 25 - Criação de DataBinding

No menu seguinte devemos então colocar os dados referentes à nossa BD.

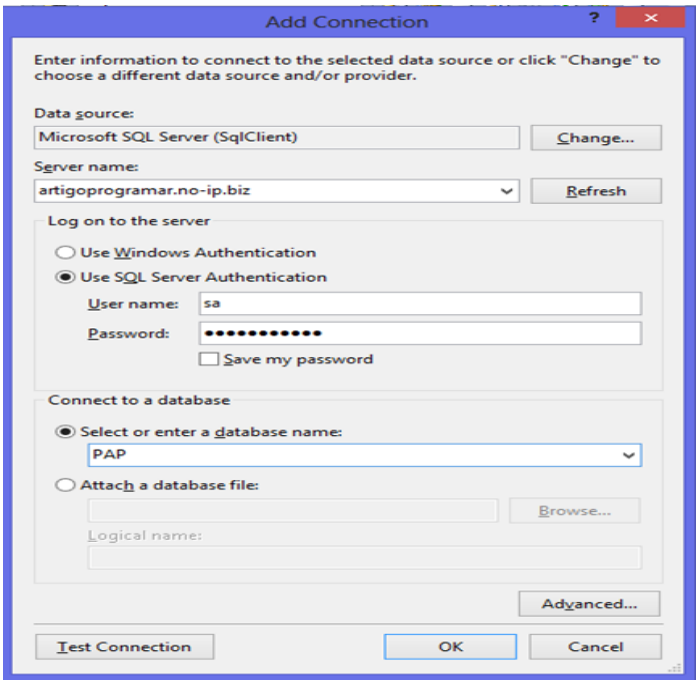


Ilustração 26 - Inserir Dados da ligação

Depois disto é efectuada a ligação à nossa BD e são-nos mostradas as tabelas que compõem a mesma.

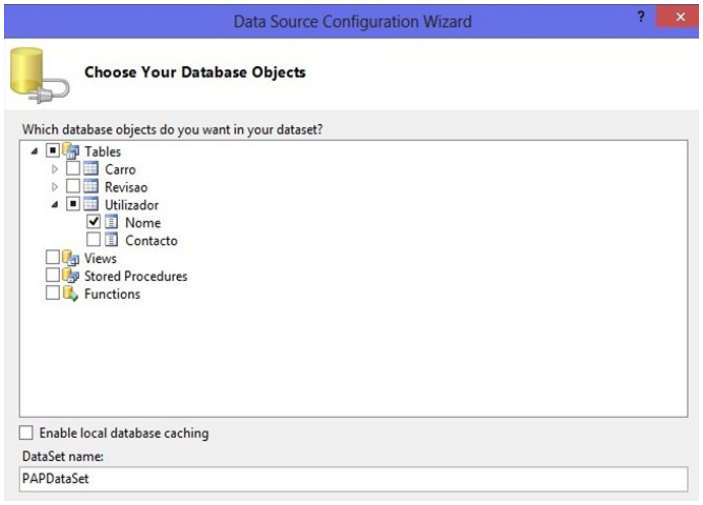


Ilustração 27 - Escolher Campos da Tabela Utilizador

Neste caso escolhemos o campo Nome da Tabela Utilizador. Após isso o wizzard traz-nos de volta ao ponto inicial, e escolhemos quer para o *Display Member*, como para o *Value Member* o tal campo "Nome" que escolhemos na BD.

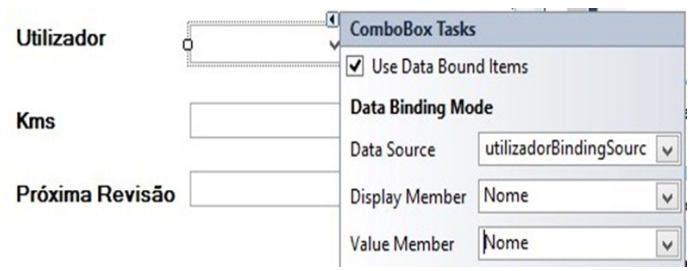


Ilustração 28 - Fonte de Dados da ComboBox

(Por curiosidade, caso tivéssemos escolhido no wizzard ambos os campos da tabela Utilizador, poderíamos escolher para o Display Member e para o Value Member valores diferentes caso o desejássemos.)

Quando executamos a nossa aplicação irá aparecer então o utilizador "Rui" no valor da *ComboBox*. (Aparece o utilizador "Rui", pois foi o que adicionamos previamente à BD.)

Ilustração 29 - ComboBox

Mas supondo agora que queríamos inserir um novo carro na BD, o qual queríamos que tivesse o utilizador "Guilherme".

Para que isso acontecesse, tendo em conta que explicitamos anteriormente que os dados a aparecer na *ComboBox* seriam o nome de todos os utilizadores inseridos na BD, apenas teremos que adicionar um novo utilizador ao sistema. Posto isto, clicando na *TabPage* "Novo Utilizador".

Vamos adicionar o Guilherme assim como o seu contacto telefónico.

Ilustração 30 - Adicionar novo utilizador

Clicando então no botão adicionar Utilizador, é-nos dada a informação de que o utilizador foi inserido com sucesso, e é-nos mostrada toda a informação da tabela Utilizador, mostrando que o Guilherme já faz parte do nosso sistema.

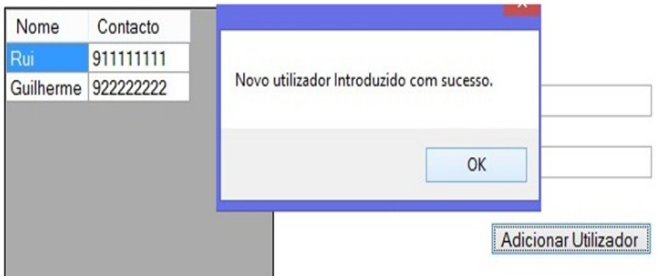


Ilustração 31 - Novo utilizador Adicionado

Quando voltamos ao separador “Adicionar Carro”, clicando na *ComboBox*, já temos no nosso conjunto de dados o “Guilherme”, como mostra a figura seguinte.

Ilustração 32 - Dados da ComboBox

Mas voltando um pouco atrás, qual o código do botão “Adicionar Utilizador”?

```
private void AdicionarUtilizadorbutton_Click(object
sender, EventArgs e)
        {
            string nome = nometextBox.Text;
            int contacto = Convert.ToInt32
                                (contactotextBox.Text);
  SqlConnection myConnection = new SqlConnection();
   myConnection.ConnectionString = @"Data
              Source=artigoprogramar.no-ip.biz;user
  id=sa;password=p@ppassword;Trusted_Connection=no;
               database=PAP;connection timeout=30";
     myConnection.Open();

            try
            {

    String sql = "INSERT INTO Utilizador (Nome,
           Contacto)" + " VALUES('" + nome + "'," +
                                          contacto + ")";
   SqlCommand commandTarefa = new SqlCommand(sql,
                                           myConnection);
   commandTarefa.ExecuteNonQuery();
   MessageBox.Show("Novo utilizador Introduzido com
                                              sucesso.");
   myConnection.Close();

            }
            catch
            {
                MessageBox.Show("ERRO ao Inserir
                          novo utilizador.", "Error",
       MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

            nometextBox.Text = "";
            contactotextBox.Text = "";

            verUtilizadores();
            this.utilizadorTableAdapter.Fill
                      (this.pAPDataSet.Utilizador);
        }
```

Mais uma vez recapitulamos a ConnectionString à BD, só que desta vez em vez de seleccionarmos todos os dados presentes na tabela, usamos a instrução simples da linguagem SQL de forma a inserir uma nova linha na tabela Utilizador, com os dados que inserimos nas textBox’s anteriormente preenchidas por nós. Através do SqlCommand pedimos a execução da nossa String sql, usando a ConnectionString da nossa BD. Caso os dados sejam introduzidos com sucesso essa mensagem é mostrada ao utilizador, e fechamos conexão à BD.

Depois disso, voltamos a chamar a função *verUtilizadores* (análoga à função *mostraCarros()*), de forma a mostrar no *datagridview* a lista actualizada de utilizadores presentes no sistema.

Gostaria ainda de chamar ao leitor a atenção para outra linha de código que chamamos neste botão.

```
this.utilizadorTableAdapter.Fill
                      (this.pAPDataSet.Utilizador);
```

Esta linha de código foi originada pelo nosso *DataBinding* quando usamos o wizzard para criarmos a fonte de dados que iria preencher a nossa *ComboBox*, quando chamamos novamente a “linha” de código, ela vai renovar novamente a fonte de dados desse elemento e é por isso que o “Guilherme” aparece logo na *ComboBox* da *TabPage* “Adicionar Carro”. Caso não chamássemos esta função, o Guilherme apenas iria aparecer nas opções da *ComboBox* na próxima vez que voltássemos a fazer *load* da classe Adicionar (clicando no botão Adicionar do Menu Principal).

Para adicionar o novo veículo vamos faze-lo de forma análoga à inserção de um novo utilizador:

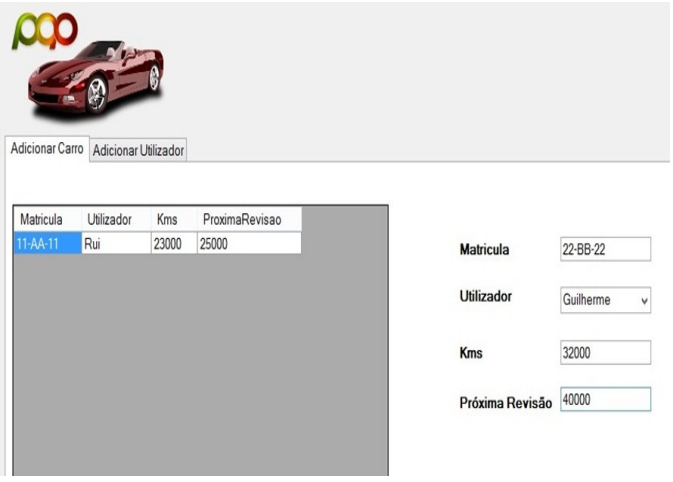


Ilustração 33 - Inserir novo Carro

Carregando no botão adicionar carro, o novo carro é inserido e a informação da tabela Carro é mostrada no *datagridview*.

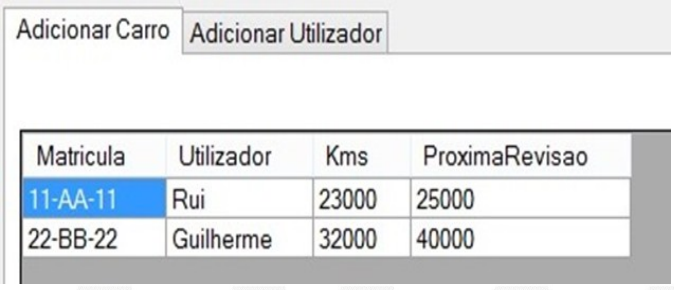


Ilustração 34 - Novo carro adicionado

Código Botão Adicionar Carro:

```
private void adicionarCarrobutton_Click(object
                              sender, EventArgs e)
        {
          string matricula = matriculatextBox.Text;
          string utilizador = comboBox1.Text;
          int kms = Convert.ToInt32
                                (kmstextBox.Text);
            int kmsProximaRevisao = Convert.ToInt32
                          (kmsProximatextBox.Text);
            myConnection.Open();
            try
            {

              String sql = "INSERT INTO Carro
                       (Matricula, Utilizador, Kms,
                    ProximaRevisao)" + " VALUES('" +
                    matricula + "','" + utilizador +
            "', "+kms+", "+kmsProximaRevisao+")";
              SqlCommand command = new SqlCommand
                              (sql, myConnection);
              command.ExecuteNonQuery();

              String sql2 = "INSERT INTO
                        KmsAnuais (Matricula)" +
                  " VALUES('" + matricula + "')";
              command = new SqlCommand(sql2,
                                    myConnection);
              command.ExecuteNonQuery();

              MessageBox.Show("Carro Introduzido
                                   com sucesso.");
              myConnection.Close();
            }
            catch
            {
                MessageBox.Show("ERRO ao Inserir
                               novo carro.", "Error",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

            matriculatextBox.Text = "";
            kmstextBox.Text = "";
            kmsProximatextBox.Text = "";
            mostraCarros();

}
```

Como o leitor pode verificar, o código é análogo ao do botão "Adicionar Utilizador", com a excepção de que a informação do Carro é colocada em duas tabelas. Na tabela *Carro* e na tabela *KmsAnuais* (em que falaremos mais à frente no nosso artigo).

Voltemos agora novamente ao nosso Menu Principal. Um dos outros botões que colocamos neste menu é o botão "Ver Próximos Revisão", e isto porque? No nosso exemplo para ser mais simples visualizar neste momento temos apenas 2 carros, mas poderíamos ter muitos mais. Como seria a forma mais simples de verificar quais os carros que têm que ir à revisão num futuro próximo? Poderíamos fazer mais uma chamada à BD para que através de instruções SQL nos fossem mostrados os carros cuja diferença entre os Kms Atuais e os Kms da próxima revisão fossem iguais ou inferiores a 2000 Kms. Ou já que temos os dados da BD no *datagridview* podemos também facilmente percorrer todas as linhas presentes nesse elemento e pintar por exemplo, de azul todas as linhas dos veículos que estejam dentro dessas condições, correcto?

Matricula	Utilizador	Kms	ProximaRevisao
11-AA-11	Rui	23000	25000
22-BB-22	Guilherme	32000	40000

Ilustração 35 - Mostra carros próximos da revisão

Como o leitor pode verificar, ao carregar no botão "Ver Próximos Revisão" a primeira linha do datagridview foi pintada de azul porque faltam precisamente 2000 Kms para a revisão do veiculo utilizador pelo Rui.

```
private void colorirRevisaobutton_Click(object
                                sender, EventArgs e)
        {
            foreach (DataGridViewRow linha in
                                 dataGridView1.Rows)
            {

                try
                {

             int kms = Convert.ToInt32(linha.Cells
                                [2].Value.ToString());
            int kmsProximaRevisao = Convert.ToInt32
                 (linha.Cells[3].Value.ToString());
            int kmsfalta = kmsProximaRevisao - kms;
                     if (kmsfalta <= 2000)
                     {

                         linha.DefaultCellStyle
                    .BackColor = Color.DeepSkyBlue;
                     }

                }

                catch {

                      }

            }
```

Código do Botão "Ver Próximos Revisão":

Ou seja, de forma simples, a aplicação verifica o valor da célula dos Kms Actuais e o valor da célula dos Kms da Próxima Revisão, vê a diferença, e caso esteja dentro das características procuradas ele "pinta" a linha com a cor azul.

Passemos agora ao Botão Revisões…

Esse botão leva-nos ao submenu Revisões em que nos são mostradas todas as revisões feitas pelos carros do sistema. (o código é análogo ao da função mostraCarros e verUtiliza-

dor, com a diferença de que a função verRevisões selecciona e mostra todos os dados da tabela Revisao)

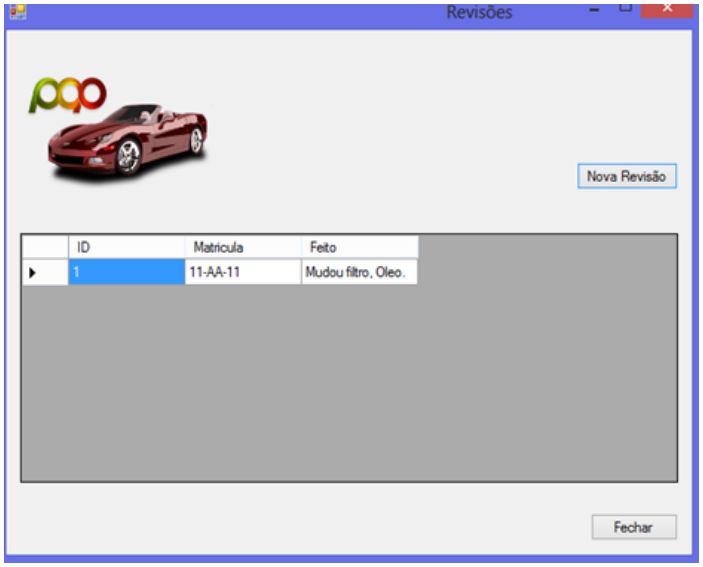


Ilustração 36 - Mostra Revisões

Carregando no botão "Nova Revisão", é nos mostrada uma janela de forma a inserirmos os dados da revisão. Na *combo-Box* aparecem todas as matrículas dos carros que estão no sistema.

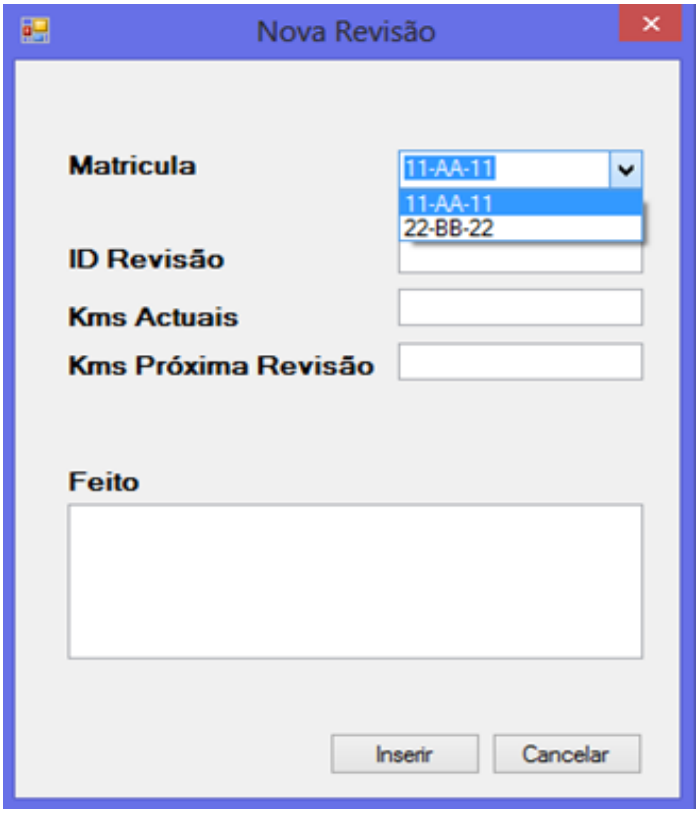


Ilustração 37 - Inserir nova revisão

É pedido para colocarmos os kms actuais do carro e os Kms em que o carro deve ir à revisão assim como o que foi feito.

Para inserirmos a informação basta-nos clicar no botão inserir e a informação será actualizada da BD.

Código do Botão "Inserir Revisão":

```
private void inserirRevisaobutton_Click(object
                                sender, EventArgs e)
        {

            int kmsActuais = Convert.ToInt32
                          (kmsActuaistextBox.Text);
            int kmsProximaRevisao =
     Convert.ToInt32(proximaRevisaotextBox.Text);
            int id = Convert.ToInt32
                           (idRevisaotextBox.Text);
            string feito = feitotextBox.Text;
            string matricula = comboBox1.Text;

            try
            {
                SqlConnection myConnection = new
                                    SqlConnection();
                myConnection.Open();

                String revisao = "INSERT INTO
                Revisao (ID, Matricula, Feito)" +
             " VALUES(" + id + ",'" + matricula +
                                  "', '"+feito+"')";
                SqlCommand commandTarefa = new
               SqlCommand(revisao, myConnection);
                commandTarefa.ExecuteNonQuery();


                SqlCommand c = new SqlCommand
                     ("UPDATE Carro SET Kms = " +
                 kmsActuais + ", ProximaRevisao =
                      "+kmsProximaRevisao+" WHERE
                 Matricula = '" + matricula + "';
                                  ", myConnection);
                c.ExecuteNonQuery();

                myConnection.Close();
                MessageBox.Show("Inserido com
                                        sucesso.");
                this.Close();

            }
            catch
                 {
                 }
        }
```

Como o leitor pode verificar, neste botão inserimos mais uma linha nova na tabela Revisao e actualizamos (UPDATE) os campos Kms e ProximaRevisao da tabela Carro. Mas que linha vamos actualizar nessa tabela? A linha que corresponde aquela matrícula específica.

Antes de passarmos ao botão que nos falta ("Gerir Kms") vamos ponderar um cenário. Imaginemos que tanto o Rui como o Guilherme fazem diversos Kms para angariar clientes para o nosso stand, pode ser conveniente sabermos quantos Kms fazem eles num mês. E ao fim do ano podemos querer saber quantos Kms fizeram eles nesse ano, a sua média, etc. Então convém irmos actualizando de mês a mês a informação sobre as deslocações deles.

No Menu Principal se clicarmos na linha correspondente ao veiculo ser-nos-á apresentada uma janela de forma a poder-

mos actualizar essa informação:

Ilustração 38 - Actualizar Kms Mensais

Clicando no botão actualizar, o carro com a matrícula 22-BB-22 é actualizado, passando a *datagridview* do Menu Inicial a mostrar a nova informação.

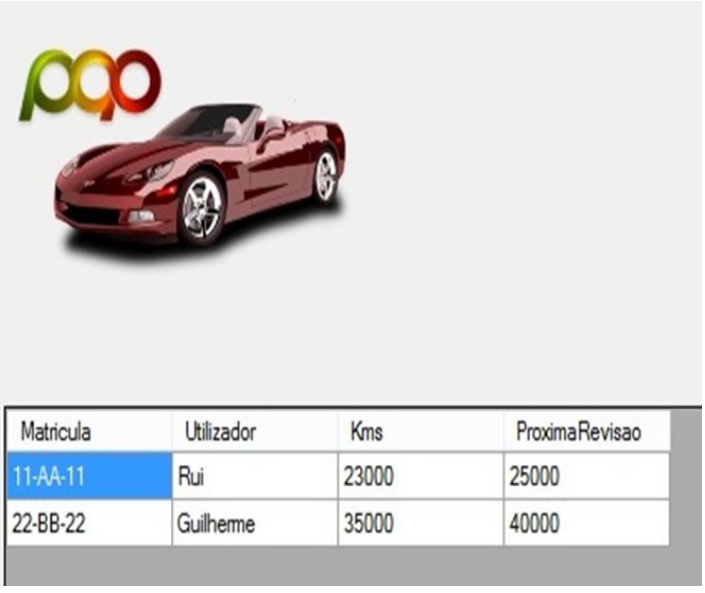


Ilustração 39 - Mostra Carros com Valores actualizados

Na janela Kms se o leitor reparar bem, são mostrados os Kms anteriores e quando estamos a colocar os Kms actuais, o programa mostra-nos a diferença de Kms percorrida. Assim quando escolhemos o mês na ComboBox o que acontece é que iremos guardar essa diferença calculada na tal tabela de KmsAnuais.

Para calcular a diferença entre os Kms que estamos a inserir e os Kms Anteriores do carro temos o evento:

```
private void kmstextBox_TextChanged(object
                                sender, EventArgs e)
        {
            int kmsAnteriores = Convert.ToInt32
                                  (kmslabel.Text);
            int kmsActuais = Convert.ToInt32
                                (kmstextBox.Text);

            diferencaKmslabel.Text = "" +
                    (kmsActuais - kmsAnteriores);

        }
```

Isto é, cada vez que estamos a inserir o valor dos Kms Actuais, o programa converte o valor inserido e calcula novamente a diferença, apresentando-a ao utilizador na label correspondente.

No botão “Actualizar”

```
private void actualizarbutton_Click(object
                                sender, EventArgs e)
 {

   String mês = comboBox1.Text;
   int kms = Convert.ToInt32(kmstextBox.Text);
   int difKms = Convert.ToInt32
                        (diferencaKmslabel.Text);

   try
    {
      SqlConnection myConnection = new
                                   SqlConnection();
      myConnection.Open();

      //Actualizar Tabela Carro - Actualizacao


      SqlCommand c = new SqlCommand("UPDATE Carro
       SET Kms = " + kms + " WHERE Matricula = '"
                    matriculalabel.Text + "'; ",
                                  myConnection);
      c.ExecuteNonQuery();


      //Actualizar Tabela KmsAnuais - Kms

     SqlCommand comma = new SqlCommand("UPDATE
         KmsAnuais SET " + mes + " = " + difKms +
                        " WHERE Matricula = '" +
      matriculalabel.Text + "'; ", myConnection);
               comma.ExecuteNonQuery();
               myConnection.Close();
               MessageBox.Show("Actualizado com
                                    sucesso.");
               this.Close();

            }
            catch { }


        }
```

Como o leitor pode verificar, actualizamos os valores da Tabela Carro, e os valores da tabela KmsAnuais na coluna do mês correspondente.

No exemplo, a diferença de Kms entre os Kms Actuais e os

QUERO FAZER UMA APLICAÇÃO SIMPLES! E AGORA? POR ONDE COMEÇO?

Kms Anteriores é 3000 Kms e escolhemos actualizar o mês de Janeiro, se verificarmos o interior da nossa BD, podemos ver que o valor foi realmente actualizado com sucesso.

	Matricula	Janeiro	Fevereiro
	11-AA-11	NULL	NULL
▶	22-BB-22	3000	NULL
*	NULL	NULL	NULL

Ilustração 40 - Valor correctamente inserido na BD

Suponha agora o leitor que com o passar do tempo actualizamos mensalmente todos os dados pertencentes a estes dois veículos, tendo assim na nossa tabela KmsAnuais todos os dados que correspondem aos kms que o Rui e o Guilherme fizeram durante um ano.

Vamos então agora falar do botão “Gerir Kms”. Este botão leva-nos até ao SubMenu Gerir Kms que nos mostra a tabela KmsAnuais.

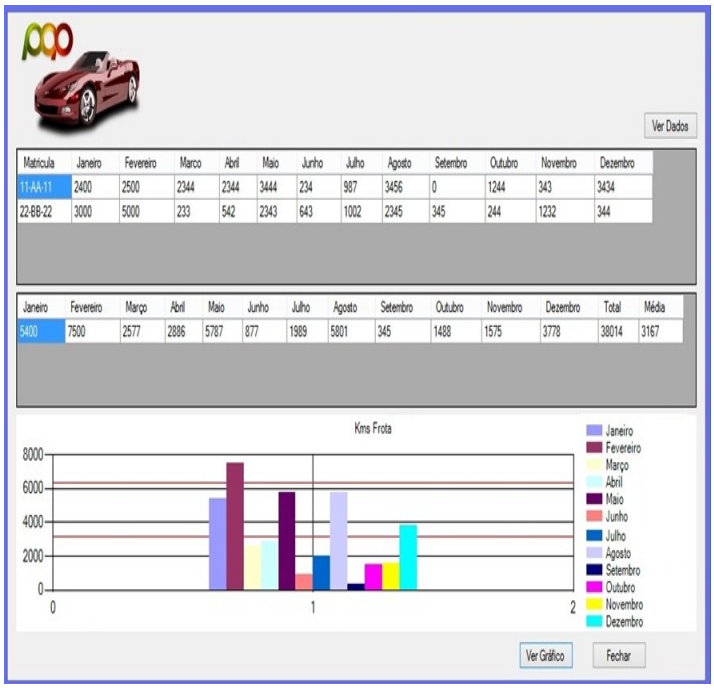


Ilustração 42 - Menu Gerir Kms

Quando clicamos nesse botão é nos apresentado no datagridview2 a soma de todas os valores de todos os meses, assim como o total de de todos os carros assim como a média. Ou seja temos os valores dos kms que a nossa frota percorreu mensalmente, o total de kms anuais e a média desses dozes meses. Essa mesma informação é mostrada em gráfico e a média é representada pela linha vermelha que aparece no gráfico. Ou seja nas barras do gráfico que aparecem abaixo da 1ª linha vermelha significa que nesse mês os dois carros percorreram menos Kms que a média. A barra que corresponde ao mês de Fevereiro que ultrapassa a 2ª linha vermelha significa que nesse mês os dois carros ultrapassaram o correspondente a 2 vezes o valor da média.

Quando ao código do Botão “Ver Gráfico” vamos por partes.

```
private void graficobutton_Click(object sender,
                                        EventArgs e)
        {

            SqlConnection conexao = new
                                    SqlConnection();
            conexao.Open();

            string sql1 = "select SUM(Janeiro) AS
                        Janeiro from KmsAnuais;";
            SqlCommand cmd1 = new SqlCommand
                                    (sql1, conexao);
            cmd1.Connection = conexao;
            cmd1.CommandText = sql1;
            String janeiro = cmd1.ExecuteScalar()
                                    .ToString();

            string sql2 = "select SUM(Fevereiro)
                   AS Fevereiro from KmsAnuais;";
            SqlCommand cmd2 = new SqlCommand
                                    (sql2, conexao);
            cmd2.Connection = conexao;
            cmd2.CommandText = sql2;
            String fevereiro =
                 cmd2.ExecuteScalar().ToString();

            string sql3 = "select SUM(Marco) AS
                        Marco from KmsAnuais;";
            SqlCommand cmd3 = new SqlCommand
                                    (sql3, conexao);
            cmd3.Connection = conexao;
            cmd3.CommandText = sql3;
            String marco =
                 cmd3.ExecuteScalar().ToString();


            string sql4 = "select SUM(Abril) AS
                        Abril from KmsAnuais;";
            SqlCommand cmd4 = new SqlCommand
                                    (sql4, conexao);
            cmd4.Connection = conexao;
            cmd4.CommandText = sql4;
            String abril =
                 cmd4.ExecuteScalar().ToString();


            string sql5 = "select SUM(Maio) AS
                         Maio from KmsAnuais;";
            SqlCommand cmd5 = new SqlCommand
                                    (sql5, conexao);
            cmd5.Connection = conexao;
            cmd5.CommandText = sql5;
            String maio =
                 cmd5.ExecuteScalar().ToString();
              string sql6 = "select SUM(Junho) AS
                        Junho from KmsAnuais;";
            SqlCommand cmd6 = new SqlCommand
                                    (sql6, conexao);
            cmd6.Connection = conexao;
            cmd6.CommandText = sql6;
            String junho =
                 cmd6.ExecuteScalar().ToString();

            string sql7 = "select SUM(Julho) AS
                        Julho from KmsAnuais;";
            SqlCommand cmd7 = new SqlCommand
                                    (sql7, conexao);
            cmd7.Connection = conexao;
            cmd7.CommandText = sql7;
            String julho =
                 cmd7.ExecuteScalar().ToString();
```

```
            string sql8 = "select SUM(Agosto) AS
                          Agosto from KmsAnuais;";
            SqlCommand cmd8 = new SqlCommand(sql8,
                                          conexao);
            cmd8.Connection = conexao;
            cmd8.CommandText = sql8;
            String agosto =
                   cmd8.ExecuteScalar().ToString();
            string sql9 = "select SUM(Setembro) AS
                        Setembro from KmsAnuais;";
            SqlCommand cmd9 = new SqlCommand(sql9,
                                          conexao);
            cmd9.Connection = conexao;
            cmd9.CommandText = sql9;
            String setembro =
                   cmd9.ExecuteScalar().ToString();

            string sql10 = "select SUM(Outubro) AS
                         Outubro from KmsAnuais;";
            SqlCommand cmd10 = new SqlCommand
                                (sql10, conexao);
            cmd10.Connection = conexao;
            cmd10.CommandText = sql10;
            String outubro =
                  cmd10.ExecuteScalar().ToString();

            string sql11 = "select SUM(Novembro) AS
                        Novembro from KmsAnuais;";
            SqlCommand cmd11 = new SqlCommand
                                (sql11, conexao);
            cmd11.Connection = conexao;
            cmd11.CommandText = sql11;
            String novembro =
                  cmd11.ExecuteScalar().ToString();

            string sql12 = "select SUM(Dezembro) AS
                        Dezembro from KmsAnuais;";
            SqlCommand cmd12 = new SqlCommand
                                (sql12, conexao);
            cmd12.Connection = conexao;
            cmd12.CommandText = sql12;
            String dezembro =
                  cmd12.ExecuteScalar().ToString();

            int total = Convert.ToInt32(janeiro) +
Convert.ToInt32(fevereiro) + Convert.ToInt32(marco)
+ Convert.ToInt32(abril) + Convert.ToInt32(maio) +
Convert.ToInt32(junho) + Convert.ToInt32(agosto) +
Convert.ToInt32(setembro) + Convert.ToInt32
(outubro) + Convert.ToInt32(novembro)
                         +Convert.ToInt32(dezembro);
            double media = total / 12;
            this.dataGridView2.Rows.Add(janeiro,
       fevereiro, marco, abril, maio, junho, julho,
     agosto, setembro, outubro, novembro, dezembro,
                                      total, media);
            //KMS
            string[] seriesArray = { "Janeiro",
    "Fevereiro", "Março", "Abril", "Maio", "Junho",
          "Julho", "Agosto", "Setembro", "Outubro",
                           "Novembro", "Dezembro" };
            int[] pointsArray = { Convert.ToInt32
             (janeiro), Convert.ToInt32(fevereiro),
    Convert.ToInt32(marco), Convert.ToInt32(abril),
     Convert.ToInt32(maio), Convert.ToInt32(junho),
   Convert.ToInt32(julho), Convert.ToInt32(agosto),
        Convert.ToInt32(setembro), Convert.ToInt32
              (outubro), Convert.ToInt32(novembro),
                       Convert.ToInt32(dezembro) };
            this.chart1.Palette =
                           ChartColorPalette.Excel;
            this.chart1.Titles.Add("Kms Frota");

            for (int i = 0; i < seriesArray.Length;
                                                 i++)
            {
System.Windows.Forms.
         DataVisualization.Charting.Series series =
            this.chart1.Series.Add(seriesArray[i]);
                series.Points.Add(pointsArray[i]);
            }

            var series3 = chart1.Series[0];
            var chartArea = chart1.ChartAreas
                                [series3.ChartArea];
            chartArea.AxisY.StripLines.Add(new
                                        StripLine
            {
                BorderDashStyle =
                           ChartDashStyle.Solid,
                BorderColor = Color.Red,
                Interval = media
            });




        }
```

Primeiro calculamos a Soma de todas as colunas da BD, e colocamos essa mesma informação dentro do *datagridview2*, assim como o total de todas as somas e a média das mesmas.

Depois seguidamente pegámos nos valores que estavam nas células do *datagridview2* e com esse vector de valores criamos o gráfico e utilizamos o valor da média para traçar os intervalos das linhas horizontais vermelhas.

Chegamos então à recta final do artigo desta edição. Se poderíamos ter feito esta aplicação de exemplo de forma diferente? Podíamos! E na maior parte dos casos poderíamos tê-lo feito de forma mais correcta, mais acertada, consumindo menos recursos computacionais, poderíamo-nos ter preocupado com a segurança (porque não é propriamente correcto apresentar na *ConnectionString* a password da BD…), podíamos ter utilizado forma mais correctas de termos a certeza do sucesso da nossa inserção e actualização de dados de forma a não corromper a BD. No caso do código do botão "Ver Gráfico", poderíamos ter formas mais simples de calcular todos os valores que apresentamos.

Mas este artigo quis apenas desmistificar alguns aspectos das interacção entre as aplicações que criamos e o *SQL Server* e o seu funcionamento em rede.

Por exemplo, no início do nosso artigo criamos o nosso host com o No-IP (artigoprogramar.no-ip.biz), mas demos também ao computador em que instalamos a BD um ip fixo dentro da nossa rede (192.168.2.100). Se quiséssemos que a nossa aplicação funcionasse dentro da nossa rede interna de casa bastaria que na *ConnectionString* colocássemos a informação do IP em que temos o serviço SQL a executar, por exemplo:

```
SqlConnection myConnection = new SqlConnection();
myConnection.ConnectionString = @"Data
Source=192.168.2.100;user
id=sa;password=p@ppassword;Trusted_Connection=no;
```

```
database=PAP;connection timeout=30";
myConnection.Open();
```

Então para quê colocarmos o artigoprogramar.no-ip.biz ? Porque este pormenor nos permite aceder à nossa BD sem estarmos na mesma rede que o computador que detém o serviço. E como? - poderá perguntar o leitor.

A forma mais simples seria abrir no nosso router a tal porta 1433 em que o serviço SQL corre, e adicionar na lista do router que todos os pedidos dirigidos a essa porta seriam enviados para o computador da rede que tem o tal ip fixo que escolhemos no início (no nosso caso 192.168.2.100).

Uma outra forma de acedermos à BD seria criarmos uma VPN na nossa rede (poderíamos usar o 192.168.2.100 na *ConnectionString*) , permitindo-nos em qualquer lugar ligarmos à rede em que temos o SQL a correr, de forma a que a VPN nos permitisse enviar informação encriptada pela internet, dando-nos um ip interno da nossa rede de casa.

> ***Mas este artigo quis apenas desmistificar alguns aspectos das interacção entre as aplicações que criamos e o SQL Server e o seu funcionamento em rede.***

Mas isso será sempre escolha de quem implementa o sistema.

Mais uma vez relembro ao leitor que neste artigo não foram considerados pormenores que são sempre importantes quando implementamos uma aplicação, como foi o exemplo já falado da segurança, e das chamadas à BD. Este é um artigo de exemplo, que procurou mostrar aos nossos leitores mais inexperientes que às vezes a "complicação" somos nós que a fazemos na nossa mente. Que para criarmos aplicações simples precisamos apenas de ideias e vontade. Este artigo é então dedicado a todos os leitores que estejam prontos a dar os primeiros passos e que nem sempre sabem por onde iniciar, aproveito também para deixar um agradecimento especial ao nosso Coordenador da Programar (António Santos), pois sem a sua paciência e dedicação, este artigo nunca teria existido.

> ***Ao longo do artigo vamos explicar de forma simples e sem grandes algoritmos como fazer de forma rápida uma peque- na aplicação que interaja com uma base de dados.***

O código será disponibilizado no fórum do Portugal-a-Programar a todos os leitores que tenham curiosidade de ver o código total do projecto. Encontramo-nos na próxima edição!

AUTOR

Escrito por Rita Peres

Natural de Castelo Branco, licenciou-se em Engenharia Informática pela Universidade da Beira Interior. Membro do P@P desde Janeiro de 2010.

Desbravando o goto!

Introdução

O assunto dos goto's é um dos tópicos mais discutidos nos fóruns de programação. Varias criticas são reiteradas para a sua não utilização, mas será o goto assim tão maléfico? Não terá realmente a sua utilidade? Com este artigo espero convence-lo que o goto ,como uma ferramenta de programação que é, tem lugar na sua caixa de ferramentas.

O que é então o goto?

Para os mais distraídos o goto esta presente em diversas linguagens de programação. Esta ferramenta permite fazer saltos unidirecionais para locais específicos quer estes sejam especificados pela linha de código em que se encontra ou por uma label.

```
goto label:
```

Na programação estruturada não temos qualquer obrigação em utilizador o goto.

Opiniões Divididas em relação ao goto!

Em 1969 Dijkstra um artigo intitulado por "*Go to Statements Considered Harmful*", e defendia a abolição do goto em todas as linguagens de alto nível. Dijkstra apresenta dois argumentos, um deles é que o goto complica a prova que uma certa parcela de código esta correta e o seu segundo argumento foi que complica a descrição de que que o programa fez ate um dado momento.

Por sua vez Donald E. Knuth escreveu um artigo chamado "Structured ProgrammingGoTo" em que exemplifica as vantagens de usar goto em linguagens procedurais, como é o caso de C.

Usos do Goto!

As linguagens de alto nível atualmente possuem um sistema de garbage collection que facilitem a vida ao programador que não tem que perder tempo a se preocupar em dealocar o espaço de memoria que foi usado. O goto é muito útil neste casos pois simplifica muito o código na leitura e elimina a necessidade de criação de funções para esse efeito o que implicaria possíveis problemas com scope. É também utilizado o goto em casos de error checking. Em outras linguagens de alto nível a próprio linguagem disponibiliza mecanismos que permite efetuar este tipo de ação como é try...exception(excepção), no caso do python. A titulo de curiosidade a implementação do return, break e continue utiliza o goto. Se analisar mos o código gerado pelo compilador veríamos que não há diferenças entre o goto e um if, e sem prejuízos de memoria.

Utilizando os goto:

```
char *buf1, *buf2;
    int err = OK;

   if ((buf1 = malloc(100)) == NULL) {
        err = ERR_MALLOC;
        goto clean_up;
    }
    if (buf2 = malloc(200) == NULL ){
         err = ERR_MALLOC;
         goto clean_up2;
}

clean_up1:
      return err; /* Visto que falhou a alocação,
                     não temos nada a libertar */
clean_up2:
      free(buf1); /* Partindo do principio que a
                     alocação do buf2 é essencial
                  para o funcionamento do programa */
      return err;
```

Sem utilizar os goto:

```
char *buf1, *buf2
int err = OK;
      if ((buf = malloc(20)) == NULL)
            return err;
      if((buf2 = malloc(200)) == NULL){
            free(buf1);
            return err;
      }
```

Neste pequeno exemplo observamos que a solução utilizando os goto permite um codigo mais simples de visualização. Este tipo de técnica esta muito presente no código do kernel do Linux. Os programadores entendem que esta é a melhor técnica para solucionar os seus problemas.

Um outro exemplo clássico do uso de goto é em ciclos encadeados. Como por exemplo:

Em alternativa ao goto podemos utilizar uma flag para

```
for(condição)
      for(condição)
            for(condição)
                  goto final
```

sinalizar quando queremos sair do ciclo:

```
for(condição pretendida ){
      flag = 0;
      for(condição pretendida && !flag){
            for(condição pretendida && !flag){
                  if(ação concluída)
                        flag = 1;
            }
      }
}
```

DESBRAVANDO O GOTO!

Como podemos verificar com a utilização do goto podemos simplificar bastante o nosso código.

Maus usos do goto!

```
MENU:
switch(op){
      case 1: goto ADD;
      case 2: goto SUB;
      case 3 : goto MULTI;
      case 4 : goto DIV;
      default: printf("Por favor escolha uma opção
                        valida\n"); goto MENU;
}
```

Uma opção mais simples seria:

```
while(1)
{
      if (op == 1)
      result = value1 + value2 /* valores pedidos
                                  previamente */
      if (op == 2)
      return = value1 - value 2  /* Verificações
                                    necesarias */
      /*....*/
```

Na ultima opção é mais fácil observarmos o que esta a acontecer. Se alguma vez utilizar o goto certifique se que estão presentes os critérios descritos em cima, se não for este o caso levante se vá beber/comer e volte a refazer, verá que me agradecera quando precisar de ler/modificar o seu código no futuro.

> *[...] Donald E. Knuth escreveu um artigo [...] em que exemplifica as vantagens de usar goto em linguagens procedurais, como é o caso de C.*

Conclusão!

Podemos então concluir, que o goto é apenas uma ferramenta e que a má reputação do goto se deve apenas a sua má utilização por parte do programador.

Com este artigo espero ter lo esclarecido sobre a utilidade e os seus maus usos do goto. Com vista a expandirem do seu conhecimento, acerca do goto, para alem deste artigo o caminho eletrotécnico para o artigo do Knuth será disponibilizado no final.

Anexo:

http://cs.sjsu.edu/~mak/CS185C/KnuthStructuredProgrammingGoTo.pdf

> *[...] o goto está presente em diversas linguagens de programação. Esta ferramenta permite fazer saltos unidirecionais para locais específicos quer estes sejam especificados pela linha de código em que se encontra ou por uma label.*

AUTOR

Escrito por Tiago Sousa

Programador autodidata. Com interesse em Sistemas Operativos. Tem conhecimento de C, Perl, Haskell. Entusiasta do movimento do software livre. Moderador do canal de irc e membro desde da comunidade 2009.

Arduino: const vs #define

Longe vão os tempos em que os computadores possuíam quantidades irrisórias de memória RAM quando comparadas com os dias de hoje. Falo sim em quantidades na ordem dos Kbytes de memória.

Actualmente os programadores descuram um pouco essa afinação e optimização na alocação de memória das suas aplicações, mas ainda existe um pequeno grupo onde ainda é necessário optimizar ao pormenor a alocação de memória, refiro-me então às áreas da robótica e electrónica, de um modo geral na utilização de micro-controladores.

#define

De uma forma simples podemos dizer que o #define é uma directiva do pré-processador do compilador que de certo modo torna mais simples a definição de uma variável estática que será usada múltiplas vezes, ou seja, no caso de ser necessário alterar o valor dessa mesma variável estática basta mudar o valor do #define ao invés de andar a alterar todas as variáveis ao longo do código.

Por exemplo:

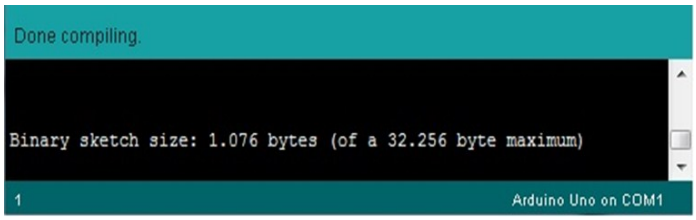
```
#define Pin 10

void setup()
 {
   pinMode(Pin, OUTPUT);
}

void loop()
{
  digitalWrite(Pin, HIGH);
  delay(500);
  digitalWrite(Pin, LOW);
  delay(500);
}
```

O que realmente acontece ao compilar o código acima é o seguinte:

```
#define Pin 10

void setup()
{
   pinMode(10, OUTPUT);
}

void loop()
{
  digitalWrite(10, HIGH);
  delay(500);
   digitalWrite(10, LOW);
   delay(500);
```

A utilização do #define é uma funcionalidade muito útil e poderosa, não havendo na realidade perdas de performance quando comparado com a definição de uma variável do tipo inteiro para definir o Pin.

Compilando o programa usando o #define obtenho:



#const

De uma forma geral o que o modificador #const faz é "dizer" ao compilador que a variável (ou ponteiro) não pode ser alterado no decorrer do código. No entanto continua a ser uma variável e dependendo de onde seja usada pode ou não consumir memória RAM.

Mas como o IDE usado pelo Arduino e seu compilador avr-gcc é inteligente o suficiente para saber que uma variável precedida pelo modificador #const não pode ser alterada dentro do programa activo este irá tentar deixa-la fora da memória RAM.

Pegando do exemplo dado anteriormente e usando o modificador #const o código ficaria assim:

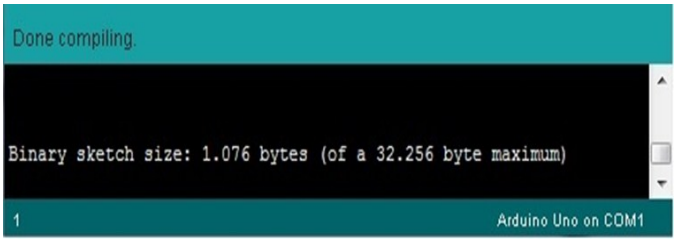
```
#const int Pin = 10;
void setup()
 {
   pinMode(Pin, OUTPUT);
}

void loop()
{
  digitalWrite(Pin, HIGH);
  delay(500);
  digitalWrite(Pin, LOW);
  delay(500);
}
```

O que realmente acontece ao compilar o código acima é o seguinte:

```
#const int Pin = 10;

void setup()
 {
   pinMode(10, OUTPUT);
}
void loop()
{
  digitalWrite(Pin, HIGH);
  delay(500);
  digitalWrite(Pin, LOW);
  delay(500);
}
```

Mas será que a utilização do modificador #const irá consumir mais ou menos memória RAM que o modificador #define?

Compilando o programa usando o #const obtenho:



O que é certo é que nenhum dos casos consome qualquer memória RAM, mas não se deixem enganar pelo valor "Binary sketch size: 1.076 bytes" apresentado pelo IDE do Arduino, pois esse valor refere-se ao HEX file do código e não tem uma relação directa com a memória RAM utilizada na realidade.

Para averiguarmos com exactidão a quantidade de memória RAM ocupada podemos usar a ferramenta AVR-SIZE disponibilizada pelo IDE do Arduino.~

AVR-SIZE

A ferramenta avr-size irá dizer-nos realmente a quantidade de memória RAM estática que o programa está a usar.

O processo de avaliar a quantidade de memória RAM estática utilizada remete-nos para o uso da consola, mas primariamente temos de navegar até a pasta "C:\Program Files (x86)\Arduino\hardware\tools\avr\bin" e encontrar o "avr-size".

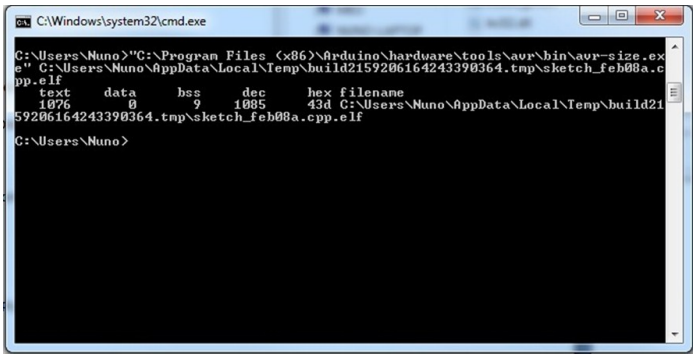
De seguida vamos navegar até ao ficheiro "C:\Users [Nome_utilizador]\AppData\Local\Temp\build[….] [nome_Sketch].cpp.elf"

E de seguida vamos correr o avr-size com o ficheiro "nome_Sketch.cpp.elf" e verificar o resultado.

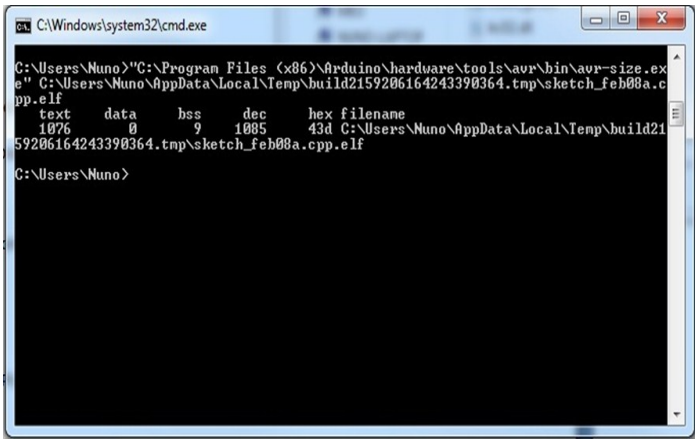
O avr-size irá retornar as seguintes informações:

- text – Não é mais que a memória flash usada para o código
- data – Representa a memória RAM estática realmente usada pelas variáveis inicializadas pelo programa
- bss - Representa a memória RAM estática realmente usada pelas variáveis inicializadas a zero pelo programa
- dec & hex – Representa o somatório da memória RAM e Flash utilizadas pelo programa

Usando #define:



Usando #const:



Como podemos verificar o valor apresentado no campo "data" que representa a memória RAM estática realmente usada pelas variáveis inicializadas pelo programa é 0 (zero), logo podemos concluir que ambos a utilização de ambos os modificadores resulta na utilização da mesma quantidade de memória RAM estática.

O que usar? #define ou #const?

A real questão que se coloca é qual dos modificadores deveremos usar, e é seguramente a questão mais complicada de responder, pois ambas as opções são válidas e ambas ocupam o mesmo espaço de memória RAM estática no decorrer do programa.

Existe quem considere a utilização do #define como uma má prática de programação pelo facto de estar mais propícia a erros de codificação, como por exemplo no pedaço de código seguinte:

Código correcto:

```
#define Pin 10
void setup()
 {
   pinMode(Pin, OUTPUT);
}

void loop()
{
```

ARDUINO: CONST VS #DEFINE

```
  digitalWrite(Pin, HIGH);
  delay(500);
  digitalWrite(Pin, LOW);
  delay(500);
}
```

Código com Erro:

```
#define Pin 10

void setup()
 {
   pinMode(3;, OUTPUT);
}

void loop()
{
  digitalWrite(Pin, HIGH);
  delay(500);
  digitalWrite(Pin, LOW);
  delay(500);
}
```

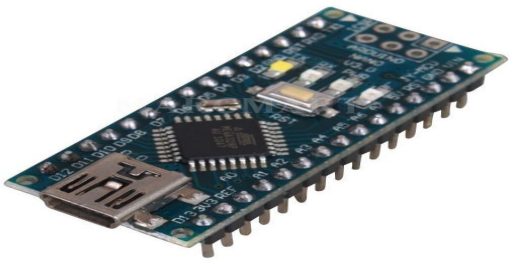
Quando tentássemos compilar o pedaço de código com erro obteríamos um erro do tipo “sketch_feb08a:6: error: expected `;' before ')' token” por parte do compilador.

> ***De uma forma simples podemos dizer que o #define é uma directiva do pré-processador do compilador que [...] torna mais simples a definição de uma variável estática [...]***

Conclusão:

A principal conclusão que podemos retirar é de que desde que o compilador avr-gcc é o suficiente inteligente para manter uma variável precedida pelo modificador #const fora da memória RAM estática do programa a sua utilização poderá ser uma vantagem. No entanto a “substituição” da variável precedida pelo modificador #define cumpre igualmente a sua função mas está mais propícia à ocorrência de erros de codificação.

> ***[...] o que o modificador #const faz é “dizer” ao compilador que a variável (ou ponteiro) não pode ser alterado no decorrer do código. No entanto continua a ser uma variável e dependendo de onde seja usada pode ou não consumir memória RAM.***



AUTOR

Escrito por Nuno Santos

Curioso e autodidacta com uma grande paixão pela programação e robótica, frequênta o curso de Engenharia Informática na UTAD alimentando o sonho de ainda vir a ser um bom Engenheiro Informático. Estudante, Blogger, e moderador no fórum Lusorobótica são algumas das suas actividades. Os seus projectos podem ser encontrados em: http://omundodaprogramacao.com/

Mitos do jQuery

Mito #1: O jQuery substitui JavaScript

Bibliotecas/toolkits/frameworks JavaScript são “apenas” um conjunto de utilitários escritos em/para JavaScript que ajudam a gerir uma página e as suas interações. Nenhuma biblioteca JavaScript substitui o JavaScript; uma biblioteca apenas junta diversas funções e outros elementos reutilizáveis. É certo que, por uma questão de coerência de código, se estivermos a usar uma biblioteca, as várias instruções JavaScript podem ser feitas recorrendo a essa biblioteca mas isso não quer dizer que apenas com JavaScript não se o faça.

Por outro lado, há situações em que não devemos forçar o uso de uma biblioteca. Quando a velocidade é determinante (e não há problemas de suporte cross-browser), é escusado colocar "mais chamadas no meio"; mais vale recorrer diretamente às API/JavaScript nativos do browser.

> *“[...]uma biblioteca apenas junta diversas funções e outros elementos reutilizáveis [...]”*

Mito #2: O jQuery serve para construir interfaces

Este é um grande mito que teima em permanecer. O jQuery serve essencialmente para pequenas manipulaçoes no DOM. Deve evitar-se ao máximo construir "pedaços" de (ou toda a) interface em JavaScript/jQuery que, por norma, servem para definir comportamento; a estrutura da interface deve ser definida em markup HTML, o seu local por excelência (pela mesma razão que não se deve ter CSS no código ou JavaScript no markup). É suposto que JavaScript/jQuery manipulem elementos já existentes no DOM. É certo que por vezes é necessário criar elementos programaticamente (ex. $(' ')), mas isso deve ser evitado. Por exemplo, em chamadas Ajax, o servidor pode enviar "pedaços" diretamente em HTML; ou então o HTML pode já estar definido (escondido) e ser manipulado e depois exibido.

Esta separação da interface do seu comportamento facilita a manutenção (ao programador web e ao web designer) mas também a graceful degradation, que dita que o site deve funcionar com o JavaScript desligado (pelo menos ter uma versão com a funcionalidade mínima) sem "rebentar".

Isto também se aplica ao jQuery UI (ou de outros plugins) que geralmente assumem que há markup HTML, conferindo-lhe apenas um look&feel.

> *“O programador deve escolher aquela que mais se adequa às necessidades (consoante os browsers e plataformas a suportar, as funcionalidades a desenvolver, etc.)”*

Mito #3: O jQuery é a melhor biblioteca JavaScript

É certo que o jQuery é a biblioteca mais popular, pela sua versatilidade e facilidade de aprendizagem. Contudo, é incorreto dizer que é a melhor. É raro haver uma “solução milagrosa para todos os problemas; há apenas a ferramenta ideal para o problema em causa. Existem inúmeras bibliotecas JavaScript, fieis a diferentes filosofias. O programador deve escolher aquela que mais se adequa às necessidades (consoante os browsers e plataformas a suportar, as funcionalidades a desenvolver, etc.). Algumas vezes nem sequer é precisa uma biblioteca. Outras vezes basta usar polyfills... Por vezes basta usar uma microframework (que se destina a colmatar uma necessidade muito específica)...

MITO #1: O JQUERY SUBSTITUI JAVASCRIPT

Em conclusão, não se deve forçar o uso de nenhuma biblioteca. Deve haver argumentos que justifiquem o seu uso. Bons argumentos são o site/webapp adquirir alguma complexidade ou haver diversos browsers e/ou plataformas a suportar. Nessa altura, o jQuery é uma boa aposta, mas há que considerar outras possibilidades.

> ***"Não quer isto dizer que não se deva usar bibliotecas (a velocidade de desenvolvimento aumenta muito, assim como a inteligibilidade do código)! Mas ao usar-se uma, deve saber-se o que se está a passar "por baixo"***

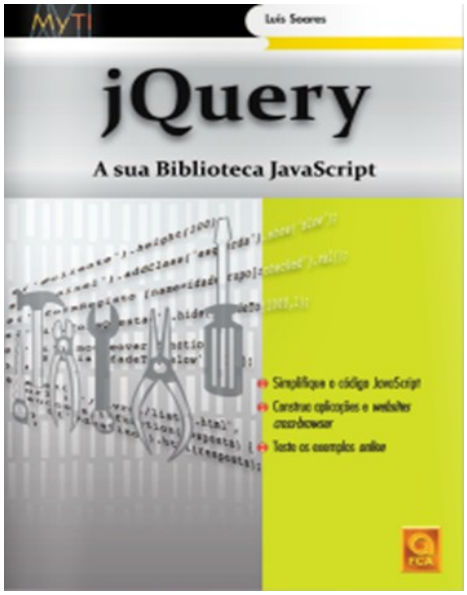
Mito #4: O jQuery torna o código mais rápido

Há até quem ache que a página fica mais rápida com jQuery... Na maioria das vezes, o código até fica mais lento. O jQuery facilita imenso a escrita de código JavaScript, pelo que é fácil cair-se em em graves erros que afetam a performance (principalmente no mau uso de seletores e na manipulação do DOM). Não quer isto dizer que não se deva usar bibliotecas (a velocidade de desenvolvimento aumenta muito, assim como a inteligibilidade do código)! Mas ao usar-se uma, deve saber-se o que se está a passar "por baixo".

O máximo que se pode dizer é que bibliotecas JavaScript promovem a escrita de código fácil de ler e modificar, oferecendo capacidades avançadas (ex.: classes), com isso aumentando muito a velocidade de desenvolvimento no geral.

Mito #5: O jQuery Mobile é o jQuery para mobile

O nome "jQuery Mobile" é muito enganador e é a causa deste mito. Leva a crer que o jQuery Mobile é a versão do jQuery mas para mobile. Na realidade é um complemento. O jQuery UI está para o desktop assim como o jQuery Mobile está para o mobile. Ambos dependem do jQuery. Ambos oferecem componentes gráficos (ex.: listas, seletor de data, tabs, etc.). Assim, o jQuery Mobile não é alternativa ao jQuery; é sim uma biblioteca de componentes para mobile tal como o jQuery UI o é para desktop.

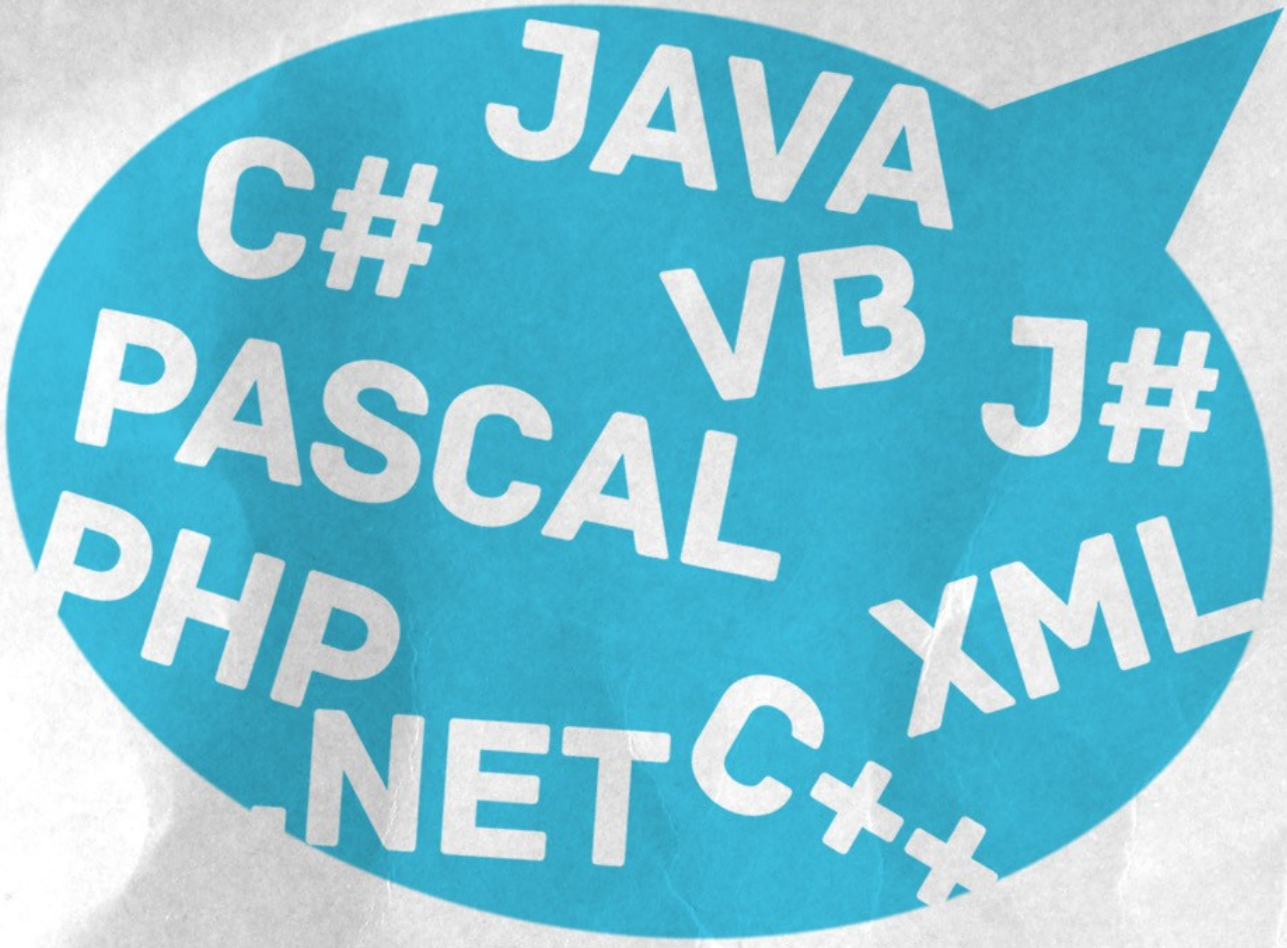


Para saber mais sobre o tema, aproveito para fazer publicidade ao livro "jQuery - A Sua Biblioteca JavaScript", em que é apresentada biblioteca jQuery, e debatidos alguns tópicos relacionados com performance.

AUTOR

Escrito por Luís Soares

Formado em Engenharia Informática e de Computadores no Instituto Superior Técnico (Licenciatura e Mestrado). Sou *web developer, _tendo já colaborado em projetos de telecomunicações e dos _media*. Gosto de linguagens de alto nível, de reutilizar código, de refactoring para simplificar. Gosto de ensinar. Escrevi um livro sobre jQuery (goo.gl/nw2Zb).
Os meus contactos estão em luissoares.com, para qualquer dúvida sobre o artigo ou outra informação.



ENTÃO, SÓ FALAS EM CÓDIGO?

TEMOS O REMÉDIO CERTO PARA TI!

portugal-a-programar.pt

A MAIOR COMUNIDADE PORTUGUESA DE PROGRAMAÇÃO, APARECE!

COLUNAS

C# - Resolução de Sobrecarga de Método

C# - RESOLUÇÃO DE SOBRECARGA DE MÉTODO

O enigma desta edição é-nos trazido por Jon Skeet.

Dado o seguinte código:

```
class X
{
    static int M
      (Func<int?, byte> x, object y) { return 1; }
    static int M
         (Func<X, byte> x, string y) { return 2; }

    const int Value = 1000;

    static void Main()
    {
        var a = M(X => (byte)X.Value, null);

        unchecked
        {
            Console.WriteLine(a);
            Console.WriteLine(M(X => (byte)
                                X.Value, null));
        }
    }
}
```

Qual é o resultado da sua execução?

Resultado

O resultado da execução é:

```
1
2
```

O que se passa aqui? Porque é que simplesmente passar a expressão para um bloco **unchecked** causa um comportamento diferente?

Explicação

A expressão em que nos devemos concentrar é esta:

```
M(X => (byte)X.Value, null)
```

Trata-se apenas de uma chamada a um método usando uma expressão lambda, usando resolução de sobrecarga de método (*method overload resolution*) para determinar qual a sobrecarga a chamara e o tipo de inferência para determinar o tipo de argumentos para o método.

Simplificando a descrição da resolução de sobrecarga de método, seguem-se os seguintes passos:

- Determina-se que sobrecargas (overloads) ão aplicáveis (isto é, quais fazem sentido em termos dos argumentos fornecidos e os correspondentes parâmetros)
- Compara-se as sobrecargas aplicáveis entre si em termos de qual “a melhor”
- Se uma sobrecarga é “melhor” que todas as outras, usar essa
- Se não há sobrecargas aplicáveis, ou nenhuma é melhor que as outras, então a chamada é inválida e leva a um erro de compilação

Pode haver a tentação de saltar diretamente para a segunda opção, assumindo que *ambas* as sobrecargas são válidas em ambos os casos.

Sobrecarga 1 – um parâmetro simples

Primeiro olhemos para a primeira sobrecarga: aquela onde o primeiro parâmetro é do tipo Func<int?, byte>. O que significa a expressão lambda X => (byte)X.Value quando convertida para aquele tipo de *delegate*? É sempre válida?

A parte manhosa é perceber o que o nome-simples X significa como parte de X.Value dentro da expressão lambda. Aqui a parte importante da especificação é o começo da seção §7.6.2 (nomes simples):

Um nome simples está na forma I ou na forma I<A_1, …, A_K>, onde I é um único identificador e <A_1, …, A_K> é uma lista opcional de tipos argumento. Quando não for especificada uma lista de tipos argumento, considera-se que K é zero. O nome simples é avaliado e classificado do seguinte modo:

- Se K é zero e o nome simples aparece dentro de um bloco e se o espaço de declaração de variáveis do bloco (ou blocos incluídos) (§3.3) contem uma variável local, parâmetro ou constante e classificado como variável ou valor.

(A especificação continua com outros casos.) Então X refere-se ao parâmetro da expressão lambda, que é do tipo int? – portanto X.Value refere-se ao valor do parâmetro subjacente.

Sobrecarga 2 – um pouco mais de complexidade

E então a segunda sobrecarga? Nesta o tipo do primeiro parâmetro de M é Func<X, byte>, portanto está-se a tentar converter a mesma expressão lambda para esse tipo. Aqui a mesma parte da seção §7.6.2 é usada, mas também a seção §7.6.4.1 é envolvida para determinar o significado da expressão de acesso a membro X.Value:

No acesso a um membro da forma E.I, se E é um único identificador, e o significado de E como nome-simples (§7.6.2) é uma constante, campo, propriedade, variável local ou parâmetro com o mesmo tipo que o significado de E como nome de tipo (§3.8), então ambos os significados possíveis de E são permitidos. Os dois significados possíveis de E.I nunca são ambíguos, uma vez que I deve necessariamente ser um membro do tipo E em ambos os casos. Por outras

palavras, a regra apenas permite acesso aos membros estáticos de E onde, caso contrário, um erro de compilação teria ocorrido.

Não é bem explícito aqui, mas a razão de que não pode ser ambíguo é porque não podem existir dois membros com o mesmo nome no mesmo tipo (para além da sobrecarga de métodos). Pode existir sobrecarga de métodos que são ambos estáticos e de instância, mas então as regras normais de sobrecarga de métodos são aplicadas.

Sendo assim, X.Value neste caso, não envolve em nada a utilização do parâmetro chamado X. Em vez disso, é a constante chamada Value dentro da classe X. Note-se que isto acontece apenas porque o tipo do parâmetro é o mesmo que se o tipo a que o nome do parâmetro se refere. (não é bem o mesmo que os nome serem os mesmos. Se uma diretiva using introduzir um alias, como using Y = X; então a condição pode ser satisfeita com nomes diferentes.)

Mas então que diferença faz **unchecked**?

```
Func<int?, byte> foo = X => (byte)X.Value;
Func<X, byte> bar = X => (byte)X.Value;
```

De volta a sobreposições

```
var a = M(X => (byte)X.Value, null);
```

Recursos

- Perfil do Jon Skeet no StackOverflow
- Blogue do Jon Skeet
- A tale of two puzzles

> ***"Trata-se apenas de uma chamada a um método usando uma expressão lambda, usando resolução de sobrecarga de método (method overload resolution) para determinar qual a sobrecarga a chamara e o tipo de inferência para determinar o tipo de argumentos para o método."***

AUTOR

Escrito por Paulo Morgado
Bacharel em Engenharia Electrónica e Telecomunicações (Sistemas Digitais) pelo Instituto Superior de Engenharia de Lisboa e Licenciado em Engenharia Informática pela Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa exerce variadas funções relacionadas com o desenvolvimento, distribuição e manutenção de software há mais de 10 anos. Participa em diversas comunidades nacionais e internacionais (pontoNETpt, NetPonto, SharePointPT, SQLPort, Portugal-a-Programar, CodeProject, CodePlex, etc.). Pelo seu contributo para com estas comunidades, a Microsoft premeia-o com o prémio MVP (C#) desde 2003. É ainda co-autor do livro "LINQ Com C#" da FCA. http://PauloMorgado.NET/ - @PauloMorgado

Media Partners da Revista PROGRAMAR

Análises

Gestão de Sistemas e Redes em Linux - 3 Edição

JavaScript (2. Edição Atualizada)

Gestão de Sistemas e Redes em Linux - 3 Edição

Título: Gestão de Sistemas e Redes em Linux - 3ª Edição

Autor: Jorge Granjal

Editora: FCA - Editora de Informática, Lda.

Páginas: 520

ISBN: 978-972-722-784-6

O livro que me foi apresentado para análise tem o título "Gestão de Sistemas e Redes em Linux – 3ª Edição" e tem como público-alvo docentes ou estudantes que pretendam aprofundar conhecimentos na área de administração de sistemas e redes, ou profissionais com responsabilidades na administração de redes informáticas de média e grande dimensão. O seu autor, Jorge Granjal, é Professor na Faculdade de Ciências e Tecnologia da Universidade de Coimbra, onde desenvolve também atividades de investigação no grupo de Comunicações e Telemática do Centro de Informática e Sistemas da Universidade de Coimbra. A primeira edição deste livro foi lançada em fevereiro de 2010, e esta mais recente, a terceira edição, publicada em setembro de 2013.

Este livro começa por apresentar um cenário de aplicação e sobre este, propõe um conjunto de atividades no âmbito da gestão de sistemas e da gestão de serviços de rede. Além disso, dedica alguns capítulos à área de segurança e por último a ferramentas para a monitorização e gestão de problemas. Na parte da gestão de sistemas, o leitor poderá encontrar atividades como a configuração de níveis de execução e serviços, gestão de utilizadores e quotas, configuração TCP/IP, gestão de logs, escalonamento de tarefas, operações sobre o *Kernel*, gestão de módulos e *Boot loaders*. Na parte de gestão de serviços de rede inclui-se a configuração do sistema como router e configuração de firewall (com o IPTables e o Squid) e a configuração de serviços nomeadamente o DNS, o NTP, DHCP, LDAP, Email (POP, IMAP e SMTP), WWW (HTTP e Webmail) e o NFS. Na parte dedicada à segurança, são propostas atividades com sistemas VPN e um IDS (Snort), e a parte final é dedicada à monitorização e gestão de problemas com ferramentas como o MRTG, Nágios e o RT.

Numa análise mais detalhada, verifica-se que o cenário de aplicação apresentado pode considerar-se equivalente a uma topologia de rede empresarial de média dimensão, com servidores distribuídos entre duas redes privadas e uma DMZ. Como sistema operativo para os servidores presentes nesta topologia de rede, é escolhido o CentOS do projeto open-source derivado dos produtos da empresa RedHat. Aqui, acrescenta-se que a versão do sistema operativo utilizada nesta edição é a 6.4 (versão mais atual do CentOS à data de publicação) e que entretanto, em dezembro de 2013, foi lançada a versão 6.5. Mesmo assim, pela análise rápida às *Major Changes* desta última versão, é de presumir os exemplos de configurações apresentados não se alterem. Da mesma forma, se o leitor pretender aplicar os conteúdos apresentados a outras distribuições Linux como o Debian, Ubuntu, Fedora (ou outras), este manual também poderá útil, uma vez que os conceitos abordados podem ser transpostos para estas distribuições. Em relação às atividades apresentadas, considera-se que estas abrangem um conjunto alargado de tecnologias, serviços e protocolos de rede, sobre as quais são explicadas as configurações básicas e, nalguns casos, configurações mais avançadas que se enquadram no cenário inicial. Verifica-se também que a maior parte destas atividades são "estanques", ou seja, permite que o leitor possa escolher os serviços que pretende configurar, sem a obrigatoriedade de seguir uma sequência predefinida, sendo que, nos casos em que tal é exigido, é indicado pelo autor. Além disso, para profissionais que já estão nas áreas de administração de redes/sistemas podem ser necessárias configurações mais específicas que podem exigir que se faça uma leitura adicional. Para isto, consideram-se úteis as referências apresentadas no final de cada capítulo, que apontam para *links* onde são encontrados tutoriais que podem servir para extrapolar o caso de estudo apresentado. É de referir igualmente que no capítulo onde são referidas as ferramentas de monitorização e gestão de problemas, poder-se-iam incluir também outras plataformas, como é exemplo do Webmin (http://www.webmin.com/) e o Cacti (http://www.cacti.net/) que permitem integrar, configurar e monitorizar múltiplos serviços e podem constituir-se como um auxílio importante na área da gestão de sistemas e redes.

Em suma, considera-se que este livro constitui um bom manual, com referências atualizadas e úteis para profissionais na área de administração de sistemas e redes, ou para estudantes que pretendam explorar a área da gestão de sistemas e redes e pretendam no futuro aplicar estes conhecimentos em cenários reais.

AUTOR

Escrito por Pedro Pinto

Pedro Pinto - Licenciado em Engenharia Electrotécnica e de Computadores, Mestre em Redes e Serviços de Comunicação e doutorando na FEUP. As suas áreas de interesse são: Routing, QoS e Segurança. Atualmente é docente no Instituto Politécnico de Viana do Castelo (IPVC) onde leciona unidades curriculares na área das redes de computadores. .

JavaScript (2. Edição Atualizada)

Título: JavaScript (2.ª Edição Atualizada)

Autor: Luís Abreu / João Paulo Carreiro

Editora: FCA

Páginas: 208

ISBN: 978-972-722-785-3

Ao começar a ler este livro a primeira coisa que me saltou à mente foi “ok, vou ler sobre JavaScript… Vamos ver o que este titulo singelo e esta capa minimalista me reservam”. Com o ler das primeiras páginas comecei a perceber que me reservavam algo bem mais amplo do que um texto extremamente tecnicista, apenas acessível a quem já está familiarizado com a linguagem, a programação para web, os seus conceitos e o paradigma da programação orientada a objectos.

Página a página, capitulo a capitulo, pude constatar que o livro é de uma leitura interessante e pouco maçadora, com exemplos que ilustram o que é explicado, não requerendo grande conhecimento prévio sobre o tema.

Achei particularmente interessante a forma como são apresentados os conceitos de objectos e os objectos, ao longo do livro, não só no capitulo 2º como mais tarde no capitulo quarto onde são apresentados aspectos como a herança entre classes. Pareceu-me acessível até para quem nunca lidou com o paradigma da programação orientada a objectos, no entanto perfeitamente explícito para que quem lê entenda como aplicar o paradigma em JavaScript.

Ao longo de todo o livro o leitor é de certa forma “levado” numa aprendizagem progressiva e constantemente aprazível da linguagem JavaScript, sem “constrangimentos”, ou “dependência excessiva de determinadas frameworks”.

Pareceu-me um livro de leitura acessível e grande pertinência, numa altura em que o JavaScript cresce cada vez mais em popularidade e utilidade tanto na programação para ambientes web, como na programação para outros ambientes , como o caso das aplicações Windows Store em que a aplicação pode ser toda desenvolvida em linguagens de markup e programação tradicionalmente usadas para web, até às aplicações que requerem grandes interacções com grandes volumes de dados oriundos de sistemas disponíveis na web, como o caso de alguns sistemas SIG.

Confesso que ao longo do livro me recordei de alguns projectos em que trabalhei e utilizei bastante javascript tanto mais porque constantemente estava a recorrer a arrays de JSON como a AJAX, para efectuar transferências de dados de algum volume e colocar os dados de forma “compreensível ao utilizador” no lado cliente-side. Recordei-me particularmente disso, porque na altura a informação que encontrava disponível na língua de Camões sobre JavaScript era pouca e de pobre qualidade.

No final do livro, fiquei com a sensação de que me “sabia a pouco”, que gostaria de ter lido um pouco mais, mas compreendi que sairia do âmbito essencial deste livro, maiores explicações sobre paradigmas e tecnologias.

Em suma, posso dizer que sou um leitor satisfeito, que recomenda este livro a quem pretende aprender como autodidacta, como aos bloggers, que pretendam aprender a personalizar mais os templates que usam nos vossos blogs, mas especialmente a quem estuda nos diversos níveis de ensino, em especial nos cursos com elevada componente tecnológica, em que este livro poderá ser útil não apenas como um “farol de referência”, mas como um guia para se aprofundar o conhecimento, que muitas vezes peca pela falta de profundidade com que é transmitido.

Resta-me desejar uma boa leitura a todos os que decidirem ler este livro e parabenizar os autores, pela qualidade do texto que além de aprazível, se revela desafiante, entusiasmante e cativante, características não muito comuns neste tipo de livros e que tanto requerem dos seus autores.

AUTOR

Escrito por Sara Freixo

Licenciada na Escola Superior de Educação do Porto, iniciou-se no mundo da programação em 1996, tendo acompanhado a evolução da tecnologia desde então. Participou em alguns projectos de desenvolvimento multimédia, sendo o seu fascínio pela programação cada vez maior. É neste momento WebDeveloper em part-time, estando envolvida em alguns projectos de software Open-Source, com especial enfoque em projectos de desenvolvimento Web.

COMUNIDADES

Comunidade NetPonto – Integrar o Facebook numa aplicação Windows Phone

INTEGRAR O FACEBOOK NUMA APLICAÇÃO WINDOWS PHONE

Este artigo explica como integrar o Facebook numa aplicação Windows Phone.

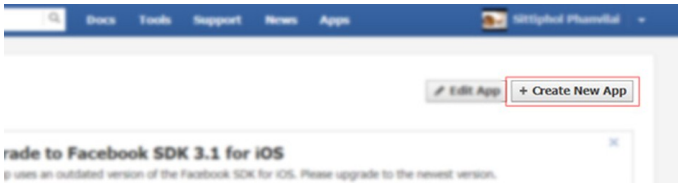
Introdução

Facebook é neste momento a rede social mais popular no mundo. Poderá obter benefícios da integração do Facebook na sua aplicação, por exemplo, ganhar mais utilizadores, publicar coisas em nome da sua aplicação, etc Este artigo irá mostrar-lhe como integrar facilmente o Facebook na sua aplicação do Windows Phone 8.

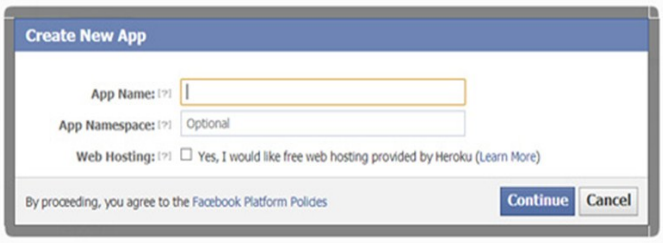
Criar aplicação Facebook

Primeiro que tudo, é necessário criar uma aplicação Facebook no site do Facebook. O "link" de referência é https://developers.facebook.com/apps.

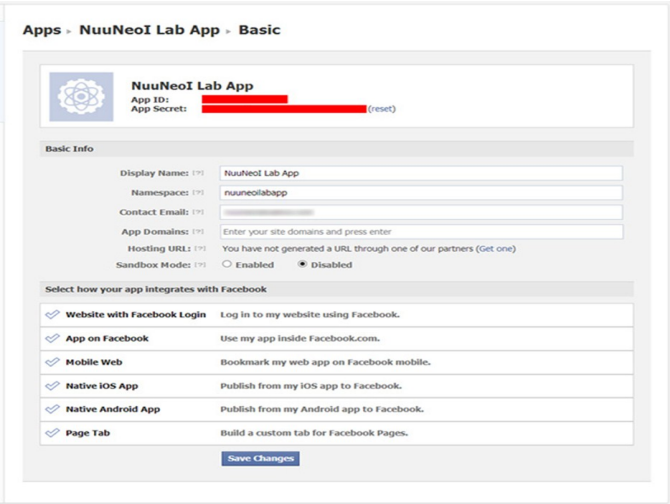
Clique no botão **"Create New App"**



Introduza o Nome da aplicação e o "namespace" (opcional)



Anote o "App ID" e o "App Secret" que será preciso no desenvolvimento da aplicação Windows Phone.

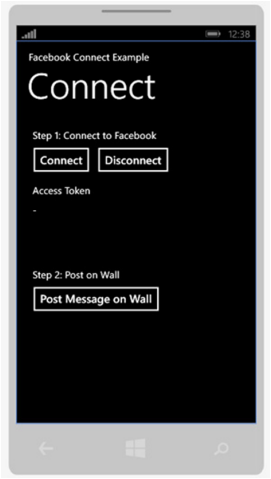


Configure o Projeto

Prepare as páginas

Coloque os botões **Connect/Disconnect** assim como **Post Message on Wall** na MainPage.

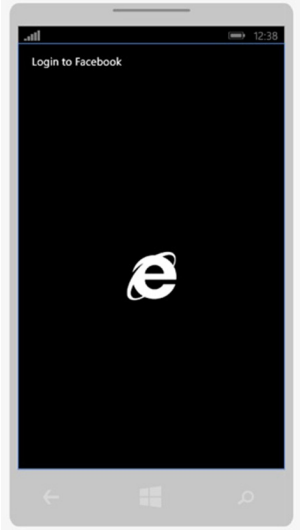
Para conectar a aplicação ao Facebook, é preciso pressionar o botão "Connect", a aplicação irá navegar para a próxima página "ConnectPage".



A página ConnectPage é apenas uma página em XAML com um "WebBrowser".

ConnectPage.xaml

```
<phone:WebBrowser x:Name="mWebBrowser"
      HorizontalAlignment="Left"
      VerticalAlignment="Top"
      Height="696"
      Width="480"
      IsScriptEnabled="True"
      Navigated="WebBrowser_Navigated"/>
```



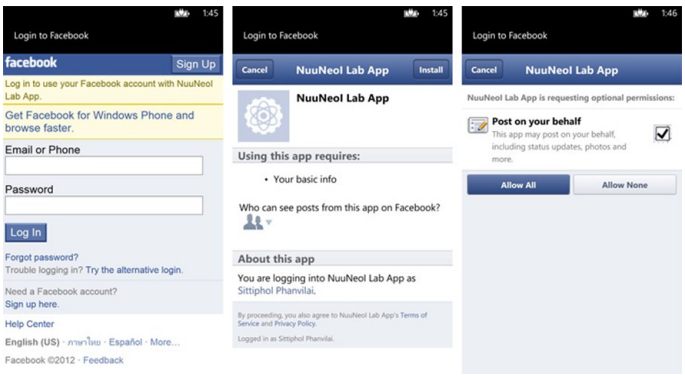
Adicionar bibliotecas

É preciso adicionar 4 pequenos ficheiros que encontram abaixo. Para o primeiro, **FacebookClient.cs**, é preciso adicionar o **App ID** e o **App Secret** às variáveis **appId** e **clientSecret**.

```
// FacebookUtils/FacebookClient.cs

using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading.Tasks;
using System.IO.IsolatedStorage;

namespace FacebookUtils
{
    public class FacebookClient
    {
        private static FacebookClient instance;
        private string accessToken;

        private static readonly
           IsolatedStorageSettings appSettings =
     IsolatedStorageSettings.ApplicationSettings;

        private String appId = "xxx";
        private String clientSecret = "xxx";
        private String scope = "publish_stream";

        public FacebookClient()
        {
            try
            {
                accessToken = (string)appSettings
                                    ["accessToken"];
            }
            catch (KeyNotFoundException e)
            {
                accessToken = "";
            }
        }

        public static FacebookClient Instance
        {
            get
            {
                if (instance == null)
                 instance = new FacebookClient();
                return instance;
            }
            set
            {
                instance = value;
            }
        }

        public string AccessToken
        {
            get
            {
                return accessToken;
            }
            set
            {
                accessToken = value;
                if (accessToken.Equals(""))
                    appSettings.Remove
                                    ("accessToken");
                else
                    appSettings.Add
                    ("accessToken", accessToken);
            }
        }

        public virtual String GetLoginUrl()
        {
            return "https://m.facebook.com/dialog/
         oauth?client_id=" + appId +"&redirect_uri=
                  https://www.facebook.com/connect/
                  login_success.html&scope="+ scope
                                  + "&display=touch";
        }

        public virtual String
              GetAccessTokenRequestUrl(string code)
        {
            return "https://graph.facebook.com/
                   oauth/access_token?client_id=" +
                                appId + "&redirect_uri=
                  https://www.facebook.com/connect/
               login_success.html&client_secret=" +
                     clientSecret + "&code=" + code;
        }

        public virtual String
                            GetAccessTokenExchangeUrl(
                                    string accessToken)
        {
            return "https://graph.facebook.com/
           oauth/access_token?client_id=" + appId +
                 "&client_secret=" + clientSecret +
                   "&grant_type=fb_exchange_token&
                 fb_exchange_token="+ accessToken;
        }

       public void PostMessageOnWall(string
         message, UploadStringCompletedEventHandler
                                            handler)
        {
            WebClient client = new WebClient();
            client.UploadStringCompleted +=
                                            handler;
            client.UploadStringAsync(new Uri
    ("https://graph.facebook.com/me/feed"), "POST",
      "message=" + HttpUtility.UrlEncode(message) +
                                  "&access_token=" +
              FacebookClient.Instance.AccessToken);
        }

        public void ExchangeAccessToken
        (UploadStringCompletedEventHandler handler)
        {
            WebClient client = new WebClient();
            client.UploadStringCompleted +=
                                            handler;
            client.UploadStringAsync(new Uri
                            (GetAccessTokenExchangeUrl
    (FacebookClient.Instance.AccessToken)), "POST",
                                                "");
        }

    }
}

// FacebookUtils/ResponseData.cs

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FacebookUtils
{
    public class ResponseData
    {
        public string id { get; set; }
```

```
        public ErrorData error { get; set; }
    }

    public class ErrorData
    {
        public int code { get; set; }
        public int error_subcode { get; set; }
    }
}

// Tools/UriToolKits.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Text.RegularExpressions;

namespace Tools
{
    public static class UriToolKits
    {
        private static readonly Regex
           QueryStringRegex = new Regex(@"[\?&](?
                <name>[^&=]+)=(?<value>[^&=]+)");

        public static IEnumerable<KeyValuePair
          <string, string>> ParseQueryString(this
                                        string uri)
        {
            if (uri == null)
                throw new ArgumentException
                                            ("uri");

            var matches =
                   QueryStringRegex.Matches(uri);
            for (var i = 0; i < matches.Count;
                                                i++)
            {
                var match = matches[i];
                yield return new KeyValuePair
     <string, string>(match.Groups["name"].Value,
                       match.Groups["value"].Value);
            }
        }
    }
}

// Tools/KeyValuePairUtils.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Tools
{
    public static class KeyValuePairUtils
    {
        public static TValue GetValue<TKey,
      TValue>(this IEnumerable<KeyValuePair<TKey,
                        TValue>> pairs, TKey key)
        {
            foreach (KeyValuePair<TKey, TValue>
                                        pair in pairs)
            {
                if (key.Equals(pair.Key))
                    return pair.Value;
            }

            throw new Exception("the value is not
                        found in the dictionary");
        }
    }
}
```

Lidando com a janela de login do Facebook



Uma vez que o botão **Connect** é pressionado, a aplicação irá navegar para a **ConnectPage**. Uma vez que a página é inicializada, é preciso atribuir o "source" ao "WebBrowser" usando o url preparado pelo FacebookClient. Por favor note, que é necessário limpar "cookies" antes de atribuir o "source" para limpar registos efetuados anteriormente.

```
public ConnectPage()
{
            InitializeComponent();

            // Clear Cookie to remove current
            //logged in user data
            mWebBrowser.ClearCookiesAsync();

            // Go to Login url
            mWebBrowser.Source = new Uri
          (FacebookClient.Instance.GetLoginUrl());
}
```

Agora o que é preciso é verificar o url atual do "WebBrowser" usando o evento "Navigated". Se o processo de "logging" foi realizado com sucesso o url final será "http://www.facebook.com/connect/login_success.html" ou "https://www.facebook.com/connect/login_success.html". Uma vez obtido este resultado, é necessário extrair o parâmetro "code" através de "query string" para assim procedermos ao próximo passo.

```
private void WebBrowser_Navigated(object sender,
NavigationEventArgs e)
{
    String uri = e.Uri.ToString();

    if (uri.StartsWith("https://www.facebook.com/
                        connect/login_success.html")
                            || uri.StartsWith("
                    http://www.facebook.com/connect/
                            login_success.html"))
    {
     // Remove junk text added by facebook from url
        if (uri.EndsWith("#_=_"))
        uri = uri.Substring(0, uri.Length - 4);

      String queryString = e.Uri.Query.ToString();

        // Acquire the code from Query String
        IEnumerable<KeyValuePair<string, string>>
               pairs = UriToolKits.ParseQueryString
                                        (queryString);
        string code = KeyValuePairUtils.GetValue
```

```
(pairs, "code");

        // Get access_token from code using
        //Asynchronous HTTP Request
        WebClient client = new WebClient();
        client.DownloadStringCompleted +=
            new DownloadStringCompletedEventHandler
                    (AccessTokenDownloadCompleted);
        client.DownloadStringAsync(new Uri
                        (FacebookClient.Instance.
                GetAccessTokenRequestUrl(code)));
    }
}
```

O parâmetro "code" é necessário para as trocas do token de acesso que é o parâmetro real que é preciso quando se pretende usar a conta do Facebook via Facebook App. Para trocar o código para o token de acesso, é necessário fazer um "POST request" ao servidor usando o "WebClient" como é mostrado de seguida- Uma vez que os dados estão disponíveis, extrai-se o parâmetro "access_token" e guarda-se o valor, regressando novamente à MainPage.

```
void AccessTokenDownloadCompleted(object sender,
                DownloadStringCompletedEventArgs e)
{
  string data = e.Result;
  data = "?" + data;

 // Acquire access_token and expires timestamp
 IEnumerable<KeyValuePair<string, string>> pairs =
                UriToolKits.ParseQueryString(data);
 string accessToken = KeyValuePairUtils.GetValue
                            (pairs, "access_token");
 string expires = KeyValuePairUtils.GetValue(pairs,
                                        "expires");

 // Save ac1cess_token
 FacebookClient.Instance.AccessToken = accessToken;

 // Back to MainPage
 var rootFrame = Application.Current.RootVisual as
                                PhoneApplicationFrame;
 if (rootFrame != null)
 rootFrame.GoBack();
}
```

Escrever no Wall

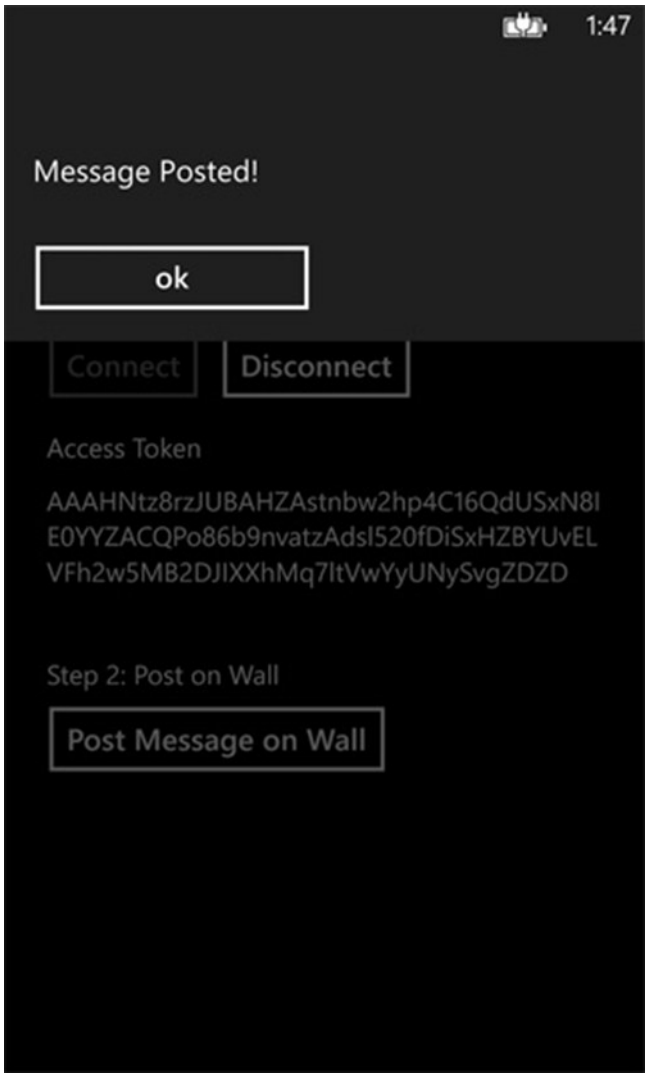
Na realidade o access_token é a chave para conectar ao Facebook. Uma vez obtido isto, é possível fazer qualquer coisa com a API. Por agora, apenas irei mostrar como escrever uma mensagem no "wall". Para o fazer simplesmente chamo o método FacebookClient.Instance.PostMessageOnWall

```
FacebookClient.Instance.PostMessageOnWall
(TextToPost, new UploadStringCompletedEventHandler
                    (PostMessageOnWallCompleted));
```

PostMessageOnWall é um método assíncrono é preciso lidar com o resultado depois de tudo. De notar que o "access_token" tem uma data limite. Caso expira, é preciso trocar por um novo "access_token" antes de fazer algo.

```
void PostMessageOnWallCompleted(object sender,
UploadStringCompletedEventArgs e)
{
    if (e.Cancelled)
        return;
    if (e.Error != null)
    {
        MessageBox.Show("Error Occurred: " +
                                    e.Error.Message);
        return;
    }
    System.Diagnostics.Debug.WriteLine(e.Result);

    string result = e.Result;
    byte[] data = Encoding.UTF8.GetBytes(result);
    MemoryStream memStream = new MemoryStream
                                                (data);
    DataContractJsonSerializer serializer = new
  DataContractJsonSerializer(typeof(ResponseData));
    ResponseData responseData = (ResponseData)
                serializer.ReadObject(memStream);

    if (responseData.id != null && !
                        responseData.id.Equals(""))
    {
        // Success
        MessageBox.Show("Message Posted!");
    }
    else if (responseData.error != null &&
                    responseData.error.code == 190)
    {
        if (responseData.error.error_subcode ==
                                                463)
        {
            // Access Token Expired, need to get
            //new token
            FacebookClient.Instance.
                                ExchangeAccessToken
             (new UploadStringCompletedEventHandler
                    (ExchangeAccessTokenCompleted));
        }
        else
        {
            // Another Error with Access Token,
                    //need to clear the Access Token
          FacebookClient.Instance.AccessToken = "";
            SetLoggedInState(false);
        }
    }
    else
    {
        // Error
    }
}

void ExchangeAccessTokenCompleted(object sender,
                UploadStringCompletedEventArgs e)
{
    // Acquire access_token and expires timestamp
    IEnumerable<KeyValuePair<string, string>> pairs
          = UriToolKits.ParseQueryString(e.Result);
    string accessToken = KeyValuePairUtils.GetValue
                            (pairs, "access_token");

    if (accessToken != null && !accessToken.Equals
                                                ("",))
    {
        MessageBox.Show("Access Token Exchange
                                            Failed");
        return;
    }

    // Save access_token
    FacebookClient.Instance.AccessToken =
                                        accessToken;
    FacebookClient.Instance.PostMessageOnWall
                                (TextToPost, new
                UploadStringCompletedEventHandler
                    (PostMessageOnWallCompleted));
}
```

INTEGRAR O FACEBOOK NUMA APLICAÇÃO WINDOWS PHONE



E é isto que iremos obter no "Wall" do Facebook!

Parabéns. A sua aplicação está conectada ao Facebook! Para ter mais funcionalidade poderá modificar o FacebookClient.cs. Para obter mais informações sobre a API consulte https://developers.facebook.com/docs/reference/api/

Código Fonte

O código fonte para o exemplo pode ser obtido em FacebookConnect.zip

> ***Poderá obter benefícios da integração do Facebook na sua aplicação, por exemplo, ganhar mais utilizadores, publicar coisas em nome da sua aplicação [...]***

Este artigo é tradução do artigo Integrate Facebook to Your Windows Phone Application, escrito por Spaso Lazarevic, como contributo à comunidade portuguesa para a Nokia Developer Wiki.

AUTOR

Escrito Por Sara Silva

Licenciada em Matemática – Especialidade em Computação, pela Universidade de Coimbra e é Microsoft Certified Professional Developer. Atualmente o seu foco de desenvolvimento incide em Windows Phone e Windows 8 Store Apps. O seu Blog **é** www.saramgsilva.com e o twitter é @saramgsilva.

No Code

O Impacto das Novas Tecnologias nas Crianças com Necessidades Educativas Especiais

O Impacto das Novas Tecnologias nas Crianças com Necessidades Educativas Especiais

Numa altura em que o mundo em que vivemos está em mudança constante, trazendo transformações positivas e negativas, a sociedade que nos envolve torna-se assim mais complexa, devido à enorme diversidade de culturas, de economias e de causas sociais. Não basta simplesmente ser alfabetizado, ou seja, aprender meramente a descodificar, é necessário que as crianças sejam capazes de exercer as práticas sociais.

A escola é portadora desse papel, sendo responsável por proporcionar à criança a oportunidade de trabalhar, de criar, de imaginar e de interagir em todas as aprendizagens. Ao professor cabe o papel de mediador essencial para consolidar essas aprendizagens. Como sabemos, a criança aprende melhor aquilo que faz sentido para ela e é através de uma cooperação íntima que podemos transformar a nossa prática diária num ambiente de verdadeira aprendizagem, onde tenhamos como base o respeito mútuo.

E é nesta base que as novas tecnologias, cada vez mais desenvolvidas podem ajudar a que a educação pré-escolar possa contribuir para uma maior igualdade de oportunidades. A educação para os média é uma das vertentes da área da Formação Pessoal e Social e da área do Conhecimento do Mundo, tornando-se indiscutivelmente uma questão de numerosas possibilidades e novos caminhos para a educação. É possível que as crianças expandam os seus horizontes e minimizem as suas diferenças.

Na parte prática que ilustra este artigo, criou-se um jogo interactivo cujo objectivo geral se centra sobretudo no desenvolvimento da autonomia. O jogo é extremamente simples, baseado apenas em páginas HTML, opção que permitiu e garantiu a jogabilidade do jogo mesmo que este seja acedido por computador, *tablet* ou por um *smartphone*.

Foi com base num diagnóstico de Síndrome Miasténico Congénito numa criança do sexo feminino que se baseou e orientou o presente artigo. Tendo em conta que a criança apresenta graves dificuldades ao nível da comunicação, o objectivo passou por construir um jogo que permita que a criança alcance um maior nível de autonomia, comunicando e interagindo com os outros.

(Para os leitores que tenham curiosidade, o jogo desenvolvido está acessível a todos os educadores e crianças que o queiram jogar. Para jogar basta apenas aceder ao site: ***www.abrincartambemseaprende.webuda.com)***

Numa altura em que os avanços tecnológicos são diários, depende da nossa vontade enquanto educadores conseguir estreitar a interação da criança com os dispositivos electrónicos.

O jogo procura solidificar duas sequências do nosso dia-a-dia, que apesar da sua simplicidade, por vezes se tornam difíceis para as crianças com N.E.E. (*dependendo da sua patologia*).

Ilustração 1 - Página Inicial do Jogo

Ao longo da construção desta ferramenta foram tomadas algumas decisões tais como as cores a usar, as sequências, o porquê do smile triste e contente assim como os aplausos finais.

O porquê das cores

O conceito de cor é um conceito abstracto e, para que uma criança seja capaz de identificar uma cor, é necessário ter milhares de referências para a ajudar a chegar ao conceito. Através dos exemplos, é importante que a criança compreenda que a cor não tem forma, tamanho, não tem contornos, pode ter diferentes texturas, formatos… Quantos mais exemplos distintos da mesma cor se mostrar a uma criança, mais facilmente a criança compreenderá o conceito da cor.

O porquê da sequência tomada

A sequência é uma das atividades mais importantes para se trabalhar com crianças do pré-escolar. Nela está a base de muitos fundamentos da Linguagem e da Matemática. Neste caso, demos grande importância às sequências temporais. Quando a criança adquire o conceito de sequência temporal, ela já pode compreender pequenas estruturas linguísticas nas histórias, como começo, meio e fim.

Daí a escolha estes dois jogos – comer e o lavar as mãos – e a sua sequência. É de extrema importância que a criança saiba o que faz a seguir e o seu porquê.

O porquê do smile triste/ do smile contente / dos aplausos

A mensagem visual (imagem) é mais simples, mais universal e retracta de uma forma mais real os acontecimentos do que a mensagem escrita. É muito mais entendível se visualizarmos imagens de um determinado acontecimento do que se o relatarmos por escrito. Desta forma escolhemos uma imagem que a criança conhece. Quando acerta aparece o Smile contente, sinal que conseguiu realizar a tarefa. Quando não acerta surge o Smile triste, símbolo que terá que tentar novamente. As palmas servem como um reforço positivo. Chegou ao fim, alcançou o objectivo da tarefa de ensino aprendizagem e é aplaudida por isso.

Ilustração 2 - Exemplo Sequência Alimentação

Como já foi referido caso a criança escolha a imagem correcta é apresentado o Smile Contente incentivando-a a continuar o jogo. A criança é assim encaminhada para uma nova página *HTML,* que mostra em tamanho mais pequeno a imagem já escolhida correctamente no passo anterior, e mostra as imagens que ainda lhe falta ordenar para terminar o jogo.

Caso a criança erre, é então mostrada a imagem do Smile Triste e a criança é reencaminhada para a página HTML em que se encontrava anteriormente.

Neste jogo é possível utilizarmos dispositivos *touch*, nomeadamente o *tablet*, em que a criança pode utilizar as próprias mãos para jogar. Quando a mobilidade da criança está afectada, esta possibilidade pode ser uma mais-valia pois por vezes torna-se difícil o manuseamento do ponteiro do rato no ecrã de um computador.

É, importante ainda referir, que qualquer criança pode jogar este jogo, pois mesmo sem ter necessidades educativas especiais, este jogo ajuda à compreensão sequencial dos passos que tem que executar na tarefa da alimentação e na tarefa da higiene.

Como breve conclusão ou reflexão na fase final deste artigo, é fundamental entender a dificuldade, de modo a promover não uma educação específica de pessoas com dificuldades intelectuais e de desenvolvimento mas gritar bem alto que não há pedagogias especiais. O que há é uma única pedagogia com diferentes grupos de aprofundamento.

Este caminho precisa de diálogo, cooperação e articulação entre todos os agentes intervenientes para que exista êxito, inovação, reabilitação… É urgente continuar a desenvolver e é neste ponto que as novas tecnologias, assim como todos os programadores têm uma voz activa pois poderão ajudar a minimizar as diferenças.

> ***"O jogo procura solidificar duas sequências do nosso dia-a-dia, que apesar da sua simplicidade, por vezes se tornam difíceis para as crianças com N.E.E. (dependendo da sua patologia)."***

Paulo Freire um dia escreveu: "Só existe saber na invenção, na reinvenção, na busca inquieta, impaciente, permanente, que os homens fazem no mundo, com o mundo e com os outros".

AUTOR

Escrito por Virgínia Belo Barata

Mestre em Educação de Infância e Ensino do 1º Ciclo do Ensino Básico / Pós Graduação e Especialização em Ensino Especial

Elege o melhor artigo desta edição

Revista PROGRAMAR

http://bit.do/ProgramarED44_V

Veja também as edições anteriores da Revista PROGRAMAR

43ª Edição - Dezembro 2013

42ª Edição - Setembro 2013

41ª Edição - Junho 2013

40ª Edição - Abril 2013

39ª Edição - Fevereiro 2013

38ª Edição - Dezembro 2012

e muito mais em ...

www.revista-programar.info

DUVIDAS?

IDEIAS?

AJUDAS?

PROJECTOS?

portugal-a-programar.org