PROGRAMAR

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

EDIÇÃO #46 - SETEMBRO 2014

ISSN 1647-0710

PARALELIZAÇÃO DE APLICAÇÕES COM

ANÁLISE

EM IOS IPHONE, IPAD E IPOD TOUCH
CURSO COMPLETO (3ª EDIÇÃO ATUALIZADA) **DESENVOLVIMENTO**

A PROGRAMAR

GESTÃO DE PROJETOS UTILIZANDO O REDMINE 2.4 NO DEBIAN WHEEZY

LUA LINGUAGEM DE PROGRAMAÇÃO PARTE 12

CLASSIFIQUE A APP

WINDOWS PHONE KIT
USAR/LER O FICHEIRO MANIFEST

COLUNAS

ENSINO DELTA
EMPRESA **COREDUMP**

NO CODE

VERSÃO MELHORADA **RASPBERRY PI**

USAR OU NÃO USAR? **JQUERY**

MÚLTIPLOS MONITORES PARA PROGRAMAR **USAR OU NÃO USAR**

COMUNIDADES

NETPONTO ESTENDENDO UMA APLICAÇÃO CRIADA NO APP STUDIO DA MICROSOFT

EDITORIAL

EQUIPA PROGRAMAR

ISSN
1 647-071 0

Tech Home Brew… Variables!

Quando escrevia este editorial, “choviam” informações via feed, sobre aquela que é de momento conhecida como CVE-2014-6271, a mais recente descoberta de uma vulnerabilidade numa das mais usadas shell’s a Bourne-again Shell (Bash).

Esta stream de informação que me inundava o software cliente de rss, fez-me pensar quanto tempo demoraria a aparecer um patch, para esta situação! Quanto tempo demoraria a resposta a uma situação classificada como de gravidade 10.

Então e sem surpresas, eis que aparece o primeiro patch que vi! Não oficial por sinal, utiliza mais um software feito em casa (home brew), como tantas outras grandes peças de software que são escritas! Isto para mim, fez-me pensar: “feito em casa, feito por dedicação, feito com “a nossa paixão”, num pensamento muito português, que nos caracteriza enquanto povo! Nas palavras do Prof. Dr. Domingos Xavier Viegas “Mestres do Desenrascanço”.

Não pude deixar de recordar o que li sobre o histórico HomeBrew Computer Club, de onde nasceram tecnologias que ainda agora usamos! Corria o ano de 1975, quando a 7 de Março, Gordon French, recebeu na sua garagem a primeira reunião do HomeBrew Computer Club. Um grupo de entusiastas das tecnologias, que pretendiam fazer algo mais com elas e trocar impressões sobre elas. Naquele espirito de HomeBrew, pessoas como John Draper também conhecido como (Captain Crunch), Stephen Gary Wozniak , Bob Marsh, entre outros, encontravam-se para trocar impressões, debater ideias, criar “coisas”, em casa, sem nenhuma indústria a apoiá-los, apenas pelo desafio e o entusiasmo!

Os anos passaram o tempo passou, o mundo mudou, a tecnologia evoluiu! Mas o espírito “HomeBrew” que esteve com as tecnologias quase desde o seu início, continua vivo!

Existem centenas de programas e peças de hardware “home brew” feitas por entusiastas apaixonados, profissionais e não profissionais das tecnologias, cuja criatividade e dedicação não tem limites, “fronteiras”, nem restrições.

Esta mesma edição acaba sendo uma “versão home brew” feita por todos os que nela escrevem e colaboram, com toda a dedicação, feita para todos quantos a lêem que são todos vós!

Resta-me ter esperança que nos muitos bytes que compõem esta edição, os “bytes livres do espirito homebrew”, sejam transmitidos nesta stream e a criatividade do verdadeiro homebrew, continue a crescer!

Até à próxima edição.

António Santos

ÍNDICE

TEMA DE CAPA

A PROGRAMAR

COLUNAS

ANÁLISES

COMUNIDADES

NO CODE

EVENTOS

9 de Outubro - PT.JUG
13 a 14 de Outubro - RubyConf
18 Outubro - SQLSaturday Porto 2014
Para mais informações/eventos: http://bit.ly/PAP_Eventos. Divulga os teus eventos para o email eventos@portugal-a-programar.pt

NOTICIAS

Em Portugal já é possível recuperar de um AVC com a ajuda de videojogos

O projeto NeuroRehab Lab está a ser desenvolvido no Instituto de Tecnologias Interativas da Universidade da Madeira e usa as novas tecnologias para acelerar o recobro. Desta forma a recuperação pode continuar em casa depois da alta médica.

O projeto nasceu em Espanha, mas o desenvolvimento está agora a ser feito na Madeira. A investigadora Ana Lúcia Faria, do Instituto de Tecnologias Interativas, juntou-se ao espanhol Sergi Bermudez Badia para criarem um sistema de recuperação motora e cognitiva para as pessoas que sofreram um AVC.

Através do recurso a videojogos e a ambientes de realidade virtual os pacientes executam várias tarefas do dia a dia num ambiente digital, como levantar dinheiro no multibanco ou ir aos correios, o que ajuda no processo de recuperação.

O sistema ajuda por exemplo no exercício do braço que foi afetado pelo Acidente Vascular Cerebral, além de tratar “os défices de memória, linguagem, funções executivas e atenção”.

"Na reabilitação tradicional, temos uma repetição de movimentos e de tarefas com vista a melhorar a plasticidade cerebral. Através da realidade virtual vamos poder oferecer, não só em espaços clínicos, mas também em casa, as mesmas metodologias de reabilitação, mas através de ambientes de simulação de atividades de vida diária", explicou Ana Lúcia Faria em declarações citadas pelo Jornal de Notícias.

Dada a forma como a recuperação é feita, os pacientes podem depois continuar o tratamento em casa, usando apenas o computador pessoal.
O objetivo do projeto que está a ser desenvolvido na Madeira passa por criar uma ferramenta gratuita que possa ser usada por pacientes e por profissionais de saúde.
"Neste momento, cerca de 30 doentes já foram abrangidos pelo projeto, tanto no serviço de medicina física e de reabilitação no Hospital Dr. Nélio Mendonça, como no Hospital João de Almada", revelou Ana Lúcia Faria. A investigadora acrescentou ainda que os pacientes fazem três tratamentos por semana durante um mês.

Escrito ao abrigo do novo Acordo Ortográfico
Tek.Sapo

Gmail já reconhece endereços com carateres não latinos e acentos

O Gmail passou a reconhecer endereços de correio eletrónico com carateres não latinos e com acentos, tudo por um serviço "mais global".
A novidade é dada a partir do blog oficial da Google e é possível graças a um protocolo de email lançado pela Internet Engineering Task Force em 2012, adotado agora pela gigante da Internet.

"A partir de agora, o Gmail (e muito em breve o Calendar) irá reconhecer endereços que contenham carateres acentuados ou não latinos. Isto significa que os utilizadores do Gmail poderão enviar e receber emails para e de pessoas que tenham tais carateres nos seus endereços de correio eletrónico", pode ler-se no post.

A Google acrescenta ainda que este é apenas um primeiro passo na estratégia de tornar o serviço verdadeiramente global e que a intenção é passar a aceitar que os utilizadores registem eles próprios endereços de correio eletrónico com carateres não latinos e com acentos a partir do Gmail.

Escrito ao abrigo do novo Acordo Ortográfico

Curso: Scrum gratuito para quem está a chegar à gestão de projetos

A oferta está disponível online e promete ensinar o básico sobre a metodologia ágil Scrum, uma das mais populares em todo o mundo na área da gestão de projetos.
O curso foi criado pela empresa brasileira MindMaster, que o ministra totalmente online e que oferece aulas, conteúdos de vídeo e outro material de apoio que o utilizador pode descarregar.

Entre os módulos integrados no curso pode encontrar os fundamentos do método ágil e os conceitos básicos para a sua implementação; os papéis e responsabilidades do Scrum no contexto das equipas de projeto; os processos e cerimónias do Scrum; ou a certificação Scrum Master.

Este módulo final foi criado para ajudar quem pretende avançar para a certificação, fornecendo dicas e mais detalhes sobre o processo.

Entre os conteúdos para download incluem-se um ebook, um guia oficial, um diagrama de processos Scrum, entre outros conteúdos. O curso está acessível a partir daqui.

Escrito ao abrigo do novo Acordo Ortográfico

TEMA DE CAPA

Paralelização de Aplicações com OpenMP

Paralelização de Aplicações com OpenMP

Introdução

O OpenMP é uma norma/API para programação paralela em sistemas de memória partilhada para as linguagens de programação C, C++ e Fortran, desenvolvida e mantida pelo OpenMP Architecture Review Board. Disponibiliza uma alternativa simples e portável a soluções de mais baixo nível como POSIX Threads, e é suportado por vários compiladores como o GCC ou ICC, e deverá chegar em breve ao Clang/LLVM.

O OpenMP distingue-se de outras soluções para suporte a paralelismo em memória partilhada pelo seu nível de abstracção, na medida em que o essencial das suas funcionalidades é obtido através de um conjunto de directivas do compilador que especificam de forma declarativa como é que diferentes partes do código podem ser executadas em paralelo. Adicionalmente, um compilador que não suporte OpenMP pode simplesmente ignorar estas directivas, continuando a aplicação a funcionar correctamente (embora de forma sequencial). Contudo, aplicações mais complexas podem também necessitar de chamadas a funções de mais baixo nível (disponibilizados pela API do OpenMP), e que tornam a compilação da aplicação dependente do OpenMP.

Neste artigo pretende-se mostrar como podem explorar paralelismo em memória partilhada recorrendo a um pequeno conjunto de funcionalidades do OpenMP, de modo a tirarem partido dos processadores com múltiplos núcleos disponíveis na generalidade dos PCs hoje em dia. Utilizar-se-á o C com linguagem de suporte neste artigo. Os programas serão compilados usando o GCC 4.7.2.

Conceitos Básicos

O OpenMP usa threads para obter paralelismo. Todas as threads usam o mesmo espaço de endereçamento de memória, sendo que os dados podem ser partilhados por todas as threads, ou privados de cada thread (cada thread tem uma cópia da variável à qual só ela acede). De notar que o OpenMP utiliza um modelo de memória de consistência relaxada, na medida em que cada threads pode ter uma cópia local temporária (por exemplo, em registos ou cache) cujo valor não necessita de ser constantemente consistente com o valor global visto por todas as outras threads (vários mecanismos são disponibilizados para que o programador possa forçar a consistência de memória em determinados pontos da aplicação).

É usado um modelo fork & join, em que temos inicialmente uma única thread (a master, ou principal), e depois temos pontos da aplicação em que um grupo de threads é criado (fork, no início de uma região paralela), e pontos da aplicação onde as threads são terminadas voltando a haver apenas uma thread (join, no final de uma região paralela), como mostrado na Figura 1. Ou seja, as regiões paralelas identificam os blocos de código onde o trabalho a ser executado é atribuído a várias threads. A norma do OpenMP prevê a possibilidade de termos regiões paralelas aninhadas, em que cada thread de um grupo de threads dá origem a um novo grupo de threads. No entanto, nem todas as implementações suportam regiões paralelas (e neste caso, cada região paralela aninhada será apenas executada por uma thread).

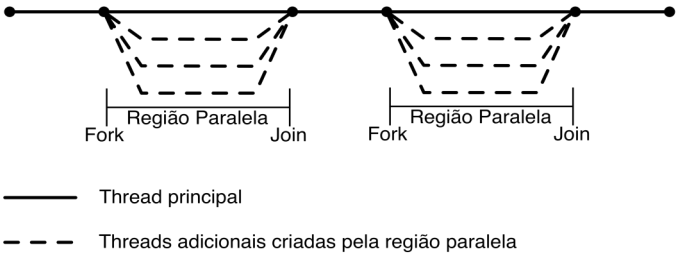


Figura 1: modelo *fork & join* do OpenMP

O acesso às funcionalidades do OpenMP é feito através de três tipos de mecanismos:

directivas de compilador

permitem criar regiões paralelas, distribuir o trabalho a ser realizado pelas várias threads, controlar o acesso a dados (especificando se são privados ou partilhados, por exemplo), assim como gerir a sincronização entre threads;

funções

permitem definir o número de threads a usar, obter informações sobre quantas threads estão a correr ou qual o identificador da thread em que estamos, determinar se se está numa zona paralela, gerir locks, aceder a funções de tempo (relógio), entre outras coisas;

variáveis de ambiente

permitem também definir o número de threads a usar, controlar o escalonamento, definir o máximo aninhamento de regiões paralelas, etc.

Nas próximas secções veremos como alguns alguns destes mecanismos podem ser usados, sendo dado especial foco às directivas de compilador.

Mecanismo do OpenMP

Nesta secção veremos como podemos usar as funcionalidades do OpenMP. O exemplos disponibilizados podem ser compilados usando a opção de compilação -fopenmp do GCC (caso estejam a usar outro compilador, devem recorrer à documentação do mesmo para determinar como activar as funcionalidades do OpenMP).

Criação de Threads

Na base do OpenMP está a directiva omp parallel. Esta directiva cria uma região paralela, i.e., indica que um determinado bloco de código será executada por várias threads (caso o OpenMP esteja activo). Como exemplo, considere o seguinte código.

```
int main(void) {
  #pragma omp parallel
  {
    printf("Hello world!\n");
  }
  return 0;
}
```

Caso execute este código (compilado com a opção do OpenMP activa), deverá ver a mensagem Hello world! aparecer no ecrã tantas vezes quanto o número de núcleos do CPU da sua máquina. Tendo em conta que não há sincronização entre as threads, é possível que o texto impresso pelas várias threads apareça misturado.

Aquilo que é executado em paralelo é delimitado pelo bloco que se segue à directiva omp parallel. Por exemplo, no seguinte código, temos mensagens que são impressas por todas as threads, e mensagens que são impressas apenas pela thread principal.

```
int main(void) {
  #pragma omp parallel
  {
    printf("Hello world!\n"); // todas as threads
    printf("All threads.\n"); // todas as threads
  }
  printf("Only master.\n"); // apenas thread
                                        //principal
  return 0;
}
```

De notar que a mensagem Only master., que apenas é impressa pela *thread* principal, deverá ser sempre a última mensagem a ser mostrada, na medida em há uma barreira implícita no final do bloco paralelo (a generalidade dos construtores têm barreiras implícitas no final, sendo que alguns mecanismos suportam a opção nowait para evitar estas barreiras; consulte a documentação para mais detalhes), *i.e.*, a *thread* principal não avança enquanto ainda houver *threads* a executar a região paralela.

Existem diversas formas de controlar explicitamente o número de *threads* que são criadas por uma região paralela. Por exemplo, podem simplesmente acrescentar a opção num_threads(3) no final do omp parallel (ficando com #pragma omp parallel num_threads(3)) para indicar que querem 3 *threads*.

Distribuição de Trabalho

O exemplo anterior não é particularmente útil na maior parte dos casos, na medida em que todas as *threads* executam o mesmo trabalho de forma redundante. Em exemplos reais, normalmente o que queremos é que cada *thread* execute uma parte diferente do trabalho. O *OpenMP* disponibiliza diversos mecanismos para tal.

Uma forma simples de distribuir trabalho é indicar que diferentes partes da região paralela podem ser executadas em paralelo, mas cada parte só deverá ser executada por uma *thread*. Para obter este comportamento podemos usar a directiva omp sections. Considere o seguinte bloco de código.

```
#pragma omp parallel
{
  #pragma omp sections
  {
    #pragma omp section
    {
      task11();  // apenas uma thread
      task12();  // apenas uma thread
    }
    #pragma omp section
    task2();  // apenas uma thread
    #pragma omp section
    task3();  // apenas uma thread
  }
  taskAll();  // todas as threads
}
```

Neste bloco de código teremos uma thread a executar tast11 e task12, outra thread a executar task2, outra thread a executar task3, e todas as threads a executar taskAll (visto que esta função já se encontra fora do bloco omp sections). Ou seja, a directiva omp sections define um conjunto de blocos de código em que cada um deles deve ser executado por apenas uma thread. Cada um dos blocos desse conjunto é identificado pela directiva omp section. Se houver mais blocos do que threads, uma mesma thread irá executar mais do que uma secção (realça-se, contudo, que a divisão de trabalho entre as threads nem sempre é equilibrada, podendo uma thread ficar com a maioria das tarefas). No final do bloco omp sections existe mais uma vez uma barreira implícita aplicada a todas as threads.

Uma outra forma muito útil de distribuir trabalho entre threads é dividindo as iterações de um ciclo for. Para tal temos a directiva omp for, usada no próximo exemplo.

```
double nums[10] = {1,2,3,4,5,6,7,8,9,10};
#pragma omp parallel for
for(int i = 0; i < 10; i++) {
  nums[i] = sqrt(nums[i]);
}
```

Este código computa a raiz quadrada dos elementos de um array. As iterações não têm dependências, pelo que podemos ter as várias iterações a serem executados em paralelo. É precisamente isso que a directiva omp for está a fazer. De referir que estamos a juntar as directivas omp parallel e omp for numa só, na medida em que ambas se aplicam ao mesmo bloco de código.

Mas nem sempre as iterações do ciclo são completamente independentes, o que pode requerer a utilização de funcionalidades mais complexas do OpenMP. Por exemplo, suponhamos que se pretende somar todos os elementos de um array. Tal pode ser feito de forma sequencial com o seguinte código.

```
double sum = 0;
double nums[SIZE];
// ...
// inicializar nums
// ...
for(int i = 0; i < SIZE; i++) {
  sum += nums[i];
}
```

Todas as iterações escrevem na variável sum, pelo que no caso de executarmos várias iterações em paralelo, teremos um *race condition*, que pode levar a resultados incorrectos. Existem várias maneiras de contornar este problema, nomeadamente através da utilização de diferentes variáveis para cada *thread*, como veremos mais adiante. Contudo, o *OpenMP* disponibiliza um mecanismo de alto nível para este tipo de situações, em que é feita uma operação de redução de um *array* (*i.e.*, os elementos do *array* são agrupados num só elemento, neste caso, a soma de todos os elementos). O mecanismo em causa é o reduction(<op>:<var>). Este mecanismo torna a variável <var> privada a cada *thread* dentro da região paralela (na generalidade dos casos, variáveis declaradas fora do bloco são partilhadas por omissão), e no final realiza uma junção de todas as cópias privadas da variável aplicando a operação <op>. Para o nosso caso, queremos que a variável sum seja privada a cada *thread*, e queremos que no final todas as cópias privadas sejam somadas. Assim, a versão paralela pode ser obtida como exemplificado no seguinte código.

```
double sum = 0;
double nums[SIZE];
// ...
// inicializar nums
// ...
#pragma omp parallel for reduction(+:sum)
for(int i = 0; i < SIZE; i++) {
  sum += nums[i];
}
```

Existem formas mais rudimentares de distribuir o trabalho por threads, usando a API do OpenMP. Pegando no exemplo anterior, a sua versão paralela pode ser também obtida através do código que se segue.

```
// variável que irá a
double sum = 0;
double nums[SIZE];
// ...
// inicializar nums
// ...
// criação de uma região paralela
#pragma omp parallel
{
  // declarada dentro da região paralela, logo
privada à thread
  double sumt = 0;
  // obter o id desta thread
  int id = omp_get_thread_num();
  // obter o número de threads
  int nthreads = omp_get_num_threads();
  for(int i = id; i < 100; i += nthreads) {
    sumt += nums[i];
  }
  // região crítica; apenas uma thread a executa
em cada momento
  #pragma omp critical
  sum += sumt;
}
```

Neste exemplo temos novamente todas as threads a executar a região paralela. Aqui a distribuição do trabalho é feita recorrendo à API do OpenMP. Obtendo o identificador de cada thread, e o número de threads a serem executados, podemos usar as estruturas condicionais para que cada thread execute trabalho diferente. No caso, for(int i = id; i < 100; i += nthreads) faz com que a thread id some os valores dos índices i tal que i % nthreads == id (sendo nthreads o número de threads a executar). Desta forma, temos as várias posições do array divididas, de forma cíclica, por todas as threads. A variável sumt está declarada dentro da região paralela, pelo que a mesma é local a cada thread. No final, todas as threads adicionam a sua soma à variável sum, que contém a soma total. Mais uma vez é necessário cuidado com as race conditions, visto que todas as threads escrevem na variável sum. A directiva omp critical é usada para evitar problemas. Esta garante que apenas uma thread está a executar aquele bloco da região paralela num determinado momento. De notar também que o acesso às funções da API do OpenMP requer a utilização do header omp.h.

Penso que todos os leitores irão concordar que a utilização do omp for e do reduction permite um código muito mais simples, e é a versão que se recomenda que seja utilizada. Por norma, apenas se recomenda fazer este tipo de divisão de trabalho manual quando não é possível fazê-lo usando apenas as directivas do compilador (o que pode ocorrer em casos mais complexos).

Sincronização de Threads

No exemplo anterior já vimos como podemos sincronizar a execução de threads através do omp critical. Existem mais alguns mecanismos que podem ser usados para sincronização:

omp atomic

é uma versão mais básica (mas também mais eficiente) do omp critical, que permite tornar as actualizações de variáveis atómicas (no exemplo anterior seria mais eficiente utilizar o omp atomic do que o omp critical);

omp master

indica que um bloco de código só será executado pela thread principal (de referir que, ao contrário da generalidade dos mecanismos do OpenMP, não existe uma barreira implícita no final destes blocos);

omp barrier

permite definir barreiras explícitas, onde as threads de uma região bloqueiam até que todas elas tenham atingido aquele ponto de execução;

omp flush

é usado para garantir que todas as threads têm uma visão consistente da memória naquele ponto (como indicado anteriormente, o OpenMP usa um modelo de memória de consistência relaxada);

omp ordered

permite definir que um determinado bloco dentro de um ciclo que está a correr em paralelo deverá ser executado na mesma ordem em que seria executado se estivesse a correr de forma sequencial.

Adicionalmente, podemos ainda sincronizar a execução de threads através das funções de gestão de locks disponibilizados pela API do OpenMP, que incluem o omp_init_lock (inicialização de um lock), omp_set_lock (adquire um lock, esperando até que tal seja possível), omp_unset_lock (liberta um lock), omp_test_lock (tenta adquirir um lock, mas não bloqueia caso não seja possível), e omp_destroy_lock (destrói o lock). Por exemplo, o omp critical usado no exemplo anterior podia ser substituído pela utilização de locks, como mostrado no próximo exemplo.

```
double sum = 0;
double nums[SIZE];
// ...
// inicializar nums
// ...
omp_lock_t sumlock;
omp_init_lock(&sumlock);
#pragma omp parallel
{
  // declarada dentro da região paralela, logo
  //privada à thread
  double sumt = 0;
  // obter o id desta thread
  int id = omp_get_thread_num();
  // obter o número de threads
  int nthreads = omp_get_num_threads();
  for(int i = id; i < SIZE; i += nthreads)
  {
    sumt += nums[i];
  }
  omp_set_lock(&sumlock);
  sum += sumt;
  omp_unset_lock(&sumlock);
}
omp_destroy_lock(&sumlock);
```

Mais uma vez, sempre que possível, aconselha-se a utilização dos mecanismos abstractos disponibilizadas OpenMP (critical, atomic, barrier, etc.) em vez da utilização das funções de mais baixo nível disponibilizados pela API.

Para terminar, convém referir que os mecanismos de sincronização devem ser evitados, pois os mesmos implicam overheads, muitas vezes ao ponto impedirem que a paralelização produza qualquer ganho de desempenho. Em geral é recomendável que se tente reorganizar o algoritmo de modo a evitar a necessidade destes mecanismos.

Tarefas Paralelas

Uma outra forma de definir blocos de código que podem ser executados em paralelo (por uma thread cada) é usando a directiva omp task. Esta directiva permite definir tarefas cuja a execução pode ser feita de forma assíncrona. A directiva de sincronização omp taskwait deve depois ser usada para bloquear a execução da aplicação até que a tarefa tenha sido concluída (quando necessário). Por exemplo, o seguinte código implementa o cálculo de um número de Fibonacci em paralelo.

```
int fibonacci(int n) {
  int fib1, fib2, result;
  if(n == 0 || n == 1) {
    result = n;
  }
  else {
    // cria uma tarefa assíncrona
    #pragma omp task shared(fib1)
    fib1 = fibonacci(n - 1);
    // cria uma tarefa assíncrona
    #pragma omp task shared(fib2)
    fib2 = fibonacci(n - 2);
    // espera pelas tarefas assíncronas lançadas
    #pragma omp taskwait
    result = fib1 + fib2;
  }
  return result;
}
```

Neste código é indicado que as chamadas recursivas à função fibonacci podem ser executadas de forma assíncrona. Antes de precisarmos de utilizar o resultados dessas chamadas recursivas, utilizamos o omp taskwait para esperarmos pela conclusão das tarefas.

Neste exemplo podemos também ver mais uma vez a utilização de mecanismos de controlo de escopo/partilha (já anteriormente tínhamos visto que o reduction podiam ser usado para tornar variáveis privadas às threads). Ao contrário da generalidade dos construtores, no omp task as variáveis declaradas anteriormente são privadas à task por omissão. Assim, se precisarmos de aceder ao seu valor mais tarde, como no caso das variáveis fib1 e fib2, é necessário tornar as variáveis partilhadas. Tal é feito através das op-

ções shared(fib1) e shared(fib2).

Conclusão

Neste artigo procurou-se mostrar como podem paralelizar aplicações de forma simples usando o OpenMP. Contudo, este disponibiliza apenas uma introdução ao tema, sendo que as funcionalidades do OpenMP vão muito para além das aqui abordadas, nomeadamente no que diz respeito ao escalonamento de tarefas, ou às formas de controlo de escopo/partilha de variáveis. Sugere-se ao leitor interessado que consulte a lista de referências disponibilizada no final caso deseje aprofundar os seus conhecimentos.

"Para além das linguagens suportadas oficialmente (C, C++ e Fortran), existem solução similares para outras linguagens como o Java"

Neste artigo também não houve grande preocupação com o desempenho da aplicação. Tirar partido da paralelização requer cuidados adicionais no que diz respeito à performance, nomeadamente para se fazer um bom uso da hierarquia de memória (por exemplo, para se evitarem problemas de false sharing), para se conseguir um bom balanceamento de carga, ou para evitar overheads excessivos com sincronização (como referido anteriormente).

Apesar do OpenMP ser destinado a sistemas memória partilhada, pode ser usado em conjunto com soluções de memória distribuída (como o MPI). Para além das linguagens suportadas oficialmente (C, C++ e Fortran), existem solução similares para outras linguagens como o Java, por exemplo.

Referências

Website do OpenMP: http://www.openmp.org/

Especificação do OpenMP: http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf

Tutorial de OpenMP: https://computing.llnl.gov/tutorials/openMP/

Tuturial prático de OpenMP: http://openmp.org/mp-documents/omp-hands-on-SC08.pdf

AUTOR

Escrito Por Rui Carlos Gonçalves

A PROGRAMAR

Gestão de projetos utilizando o Redmine 2.4 no Debian Wheezy

Lua – Linguagem de Programação – Parte 12

Como usar ler o ficheiro manifest usando o Cimbalino Windows Phone Toolkit (usando MVVM)

Classifique a App

Gestão de projetos utilizando o Redmine 2.4 no Debian Wheezy

A gestão de projetos pode ser descrita como o processo de planejamento, execução e controle de um projeto, desde o seu início até à sua conclusão, com prazo determinado e várias tarefas que devem ser realizadas para se atingir o objetivo final, com um determinado custo e qualidade, através de atividades realizadas utilizando recursos técnicos e humanos.

No processo de gestão, o principal objetivo é um produto final, de acordo com as especificações, com o melhor desempenho possível, sob uma perspectiva dinâmica. Assim, a gestão do projeto forma um ciclo dinâmico, onde tudo gira em torno do planejamento, da execução e do controle.

Atualmente, os projetos, principalmente na área de desenvolvimento de software, estão cada vez mais complexos-extrapolam prazos, aumentam custos, comprometem a qualidade do produto final por isso, um projeto deve ser bem gerenciado para minimizar ao máximo esses impactos. Para auxiliar o processo de gestão utilizamos ferramentas de software para gestão de projetos, que utilizam amplos conjuntos de funcionalidades, aderentes aos principais métodos e guias de boas práticas de gerenciamento de projetos utilizados no mercado.

O Redmine é uma poderosa e flexível aplicação web para gerenciamento de projetos. Desenvolvido usando o framework Ruby on Rails, é uma aplicação multi-plataforma e multi-banco de dados, que suporta os principais aspectos do processo de desenvolvimento de software. Além disso, possui código fonte aberto e sua licença está de acordo com os termos da GNU General Public License v2 (GPL).

Para esta instalação, utilizamos o Redmine 2.4 (versão estável no momento da escrita deste manual) e o sistema operacional Debian 7.3 (chamado de Wheezy, versão estável no momento desta escrita). Para a execução dos passos aqui descritos, é importante que você tenha conhecimentos básicos sobre o uso do Debian, Apache, Mysql e Linux em geral.

Para mantermos o foco, não explicamos todos os passos em detalhes. Os nomes de arquivos e caminhos neste documento estão sujeitos a mudanças, de acordo com a sua instalação.

Após aplicar todas as etapas de instalação descritas neste manual, você terá uma instância funcional do Redmine, de acordo com as especificações citadas abaixo. Se alguma etapa falhar, observe as versões dos softwares e das bibliotecas utilizadas, além da localização dos diretórios (estrutura dos diretórios) da instalação do sistema operacional em uso.

Convenções usadas neste documento:

- Fonte monospace: é usada para indicar entradas do utilizador ou saídas do sistema, bem como configurações de arquivos.
- ↩: é usado para indicar que uma linha não foi finalizada e continua abaixo.

Etapas de instalação

1. Instalação do Ruby, Rails, Gems and Passenger

1) Preparando o sistema

```
apt-get install gcc build-essential zlib1g ↩
zlib1g-dev zlibc ↩
libzlib-ruby libssl-dev libyaml-dev libcurl4-
openssl-dev ↩
apache2-mpm-prefork apache2-prefork-dev ↩
libapr1-dev libxslt-dev checkinstall
```

2) Baixando, construindo e instalando o Ruby

```
cd ~
wget ↩
http://ftp.ruby-lang.org/pub/ruby/1.9/ruby-1.9.3-
p448.tar.gz
tar xvfz ruby-1.9.3-p448.tar.gz
cd ruby-1.9.3-p448
./configure --enable-pthread --prefix=/usr/local
make && checkinstall --type=debian ↩
--install=yes --fstrans=no --pakdir='~'
```

3) Verificando se o Ruby está instalado e funcionando

```
ruby -v
```

Esperamos uma saída como esta:

```
ruby 1.9.3p448 (2013-06-27 revision 41675)
[x86_64-linux]
```

4) Fazendo com que o Ruby suporte o OpenSSL

```
cd ext/openssl/
ruby extconf.rb
make && checkinstall --type=debian
--install=yes ↩
--fstrans=no --pakdir='~'
```

5) O comando gem está instalado? O Ruby 1.9 vem com o RubyGems por padrão, então o comando gem já deve estar instalado. Se estiver corretamente instalado, o comando abaixo deverá apresentar uma saída com o número da versão como 1.8.2.x:

```
gem -v
```

Agora, nós podemos instalar o rdoc:

```
gem install rdoc
```

6) Instalando o Rails

```
gem install rails --no-ri --no-rdoc
```

Nota: Você pode estar recebendo a mensagem de erro "não existe arquivo para carregar- -- -zlib (LoadError)". Neste caso, você deve instalar zlib primeiro:

```
cd ruby-1.9.3-p448/ext/zlib/
ruby extconf.rb
make
make install
```

7) Instalando o Passenger (servidor de aplicação)

```
gem install passenger
passenger-install-apache2-module
```

8) Configurando o Apache
Abra o arquivo /etc/apache2/mods-available/passenger.load com o seu editor de texto preferido e digite o conteúdo abaixo numa única linha (lembre-se de fazer os ajustes dos caminhos, se necessário). Salve as modificações no arquivo:

```
LoadModule passenger_module↩
 /usr/local/lib/ruby/gems/1.9.1/gems/
                                passenger-4.0.37/↩
 buildout/apache2/mod_passenger.so
```

Abra o arquivo /etc/apache2/mods-available/passenger.conf e digite o conteúdo abaixo (lembre-se de fazer os ajustes dos caminhos, se necessário). Salve as modificações no arquivo:

```
PassengerRoot ↩
 /usr/local/lib/ruby/gems/1.9.1/gems/passenger-
4.0.37
PassengerRuby /usr/local/bin/ruby
PassengerDefaultUser www-data
```

9) Ativando o módulo Passenger

```
a2enmod passenger
```

2. Instalando o Redmine

1) Baixando o Redmine

Para a nossa instalação, baixámos a última versão estável (Redmine 2.4). Você pode encontrar essa versão no website do Redmine (http://www.redmine.org/projects/redmine/wiki/Download). Descompacte o arquivo em /opt/redmine utilizando os comandos de descompactação o unzip, gunzip ou similar.

2) Mais preparos no sistema

Nota: A instalação do pacote libmagickwand-dev acompanha muitas outras instalações de outros pacotes (dependências/recomendações). Se você encontrar um erro no pacote de instalação, você pode tentar limpar (executar o purge), em seguida, instalar os pacotes *-dev ou instalar os pacotes separadamente. Utilizando o comando gem install.

```
gem install bundler mysql2
apt-get install libmagick9-dev libmysqlclient-dev
cd /opt/redmine
bundle install --without postgresql
```

3) Criando o banco de dados
Antes de efetuarmos esta etapa, instalamos o MySQL versão 5.5:

```
apt-get install mysql-server mysql-client
```

Os comandos abaixo foram executados no cliente MySQL para a criação do banco de dados, do usuário redmine e definição de privilégios, respectivamente.

```
create database redmine character set utf8;
create user 'redmine'@'localhost' identified by
'XXX';
grant all privileges on redmine.* to
'redmine'@'localhost';
```

4) Configurando a conexão com o banco de dados Abra o arquivo /opt/redmine/config/database.yml com o seu editor de texto preferido e digite o conteúdo abaixo. Salve as modificações no arquivo:

```
production:
        adapter: mysql2
        database: redmine
        host: localhost
        username: redmine
```

5) Gerando uma sessão secreta de armazenamento

```
rake generate_secret_token
```

6) Preparando o banco de dados / Criando tabelas

```
RAILS_ENV=production rake db:migrate
```

7) Definindo as permissões para o sistema de arquivos

```
cd /opt/redmine
mkdir tmp tmp/pdf public/plugin_assets
chown -R www-data:www-data files log tmp ↩
 public/plugin_assets
chmod -R 755 files log tmp public/plugin_assets
```

8) Testando a sua instalação do Redmine

```
ruby script/rails server webrick -e production
```

Agora acessamos http://localhost:3000 e observamos o Redmine em ação.

```
ln -s /opt/redmine/public /var/www/redmine2
```

3. Configurando o Apache e o Passenger

Abrimos o arquivo /etc/apache2/sites-available/redmine com

```
 RailsEnv production
 RackBaseURI /redmine2
 Options -MultiViews
```

o seu editor de texto preferido e digite o conteúdo abaixo.

Salve as modificações no arquivo:

```
a2ensite redmine
```

Agora ativamos o novo site:

Reiniciamos o Apache e testamos a nossa nova URL: http://localhost/redmine2

Referências

NIGGEMANN, Jan. Redmine Installation Instructions. 2014. Disponível em: http://hz6.de/wp/wp-content/uploads/2013/02/installing_redmine.pdf. Acesso em: 25 de fevereiro de 2014.

ROLDÃO, Victor Sequeira. Gestão de projetos: uma perspectiva integrada. São Carlos: EdUFSCar, 2004.

AUTOR

Escrito por Francisco Anderson Freire Pereira

É técnico de Tecnologia da Informação, membro interino do LabEPI (Laboratório de Elementos do Processamento da Informação) e aluno do 7º período do curso Bacharelado em Sistemas de Informação da Universidade Federal do Rio Grande do Norte - Centro de Ensino Superior do Seridó (UFRN CERES/Caicó) - Brasil.

Lua – Linguagem de Programação – Parte 12

Como informado na décima parte desta série de artigos sobre apresentação da linguagem Lua, esporadicamente seriam mostrados outros artigos complementares com recursos que anteriormente não foram abordados. Cumprindo o prometido, apresenta-se neste artigo recursos relacionados a definição de pausa (aguardar o acionamento da tecla <Enter> para continuar o fluxo do programa), definição e uso da função sleep () com o objetivo de aguardar um certo tempo em segundos e a ordenação decrescente de elementos em uma matriz do tipo vetor.

Os detalhes aqui descritos são assuntos que surgiram durante os cursos de linguagem Lua ministrados.

Pausa no Programa

A realização de pausa na execução de um programa Lua que aguarde o acionamento da tecla <Enter> pode ser definida com a instrução io.read '*l'.

```
-- inicio do programa P11_01
  io.write("Entre o 1o. valor: ")
  A = tonumber(io.read())
  io.write("Entre o 2o. valor: ")
  B = tonumber(io.read())
  X = A + B
  io.write("Soma = ", X, "\n\n")
  io.write("Acione <Enter>... ")
  io.read '*l'
-- fim do programa P11_01
```

Em seguida escreva o código do programa em um editor de texto, gravando-o com o nome P11_01.lua e execute-o com a linha de comando lua 5.1 P11_01.lua.

O programa P11_01.lua além do uso do recurso de pausa que aguarda o uso da tecla <Enter> faz uso da função tonumber () para a leitura de um valor numérico.

A leitura de valores numéricos na linguagem Lua pode ser efetuada com o uso da forma io.read("*number") ou com o uso da forma tonumber(io.read()). A diferença entre os dois métodos de entrada é que o segundo faz a limpeza do buffer do teclado após a entrada dos valores.

Função sleep()

A linguagem Lua não possui definida uma função que efetue uma pausa controlada por tempo, mas por meio da função clock() da biblioteca io é possível criar tal função. Note o código seguinte.

```
-- inicio do programa P11_02
  function sleep(TMP)
    local I = os.clock()
    while (os.clock() - I <= TMP) do
    end
  end

  function pause()
    io.read '*l'
  end

  FAT = 1
  CONT = 1

  for CONT = 1, 5, 1 do
    FAT = FAT * CONT
  end

  sleep(4) -- 4 segundos
  print(FAT)

  io.write("\nAcione <Enter>... ")
  pause()
-- fim do programa P11_02
```

Em seguida escreva o código do programa em um editor de texto, gravando-o com o nome P11_02.lua e execute-o com a linha de comando lua 5.1 P11_02.lua.

A função definida sleep() recebe como valor de parâmetro o limite de tempo de espera em segundos e marca o tempo inicial para a variável I a partir do tempo atual. O ciclo while consome o tempo de execução dentro do limite estabelecido para TMP enquanto o tempo atual descontado do valor inicial de tempo em I seja menor ou igual ao valor definido para TMP.

Ordenação Decrescente/Descendente

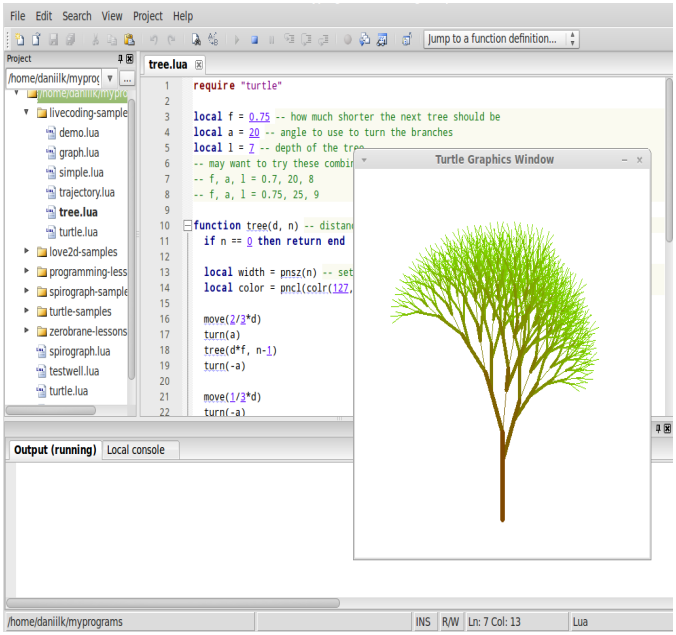
Anteriormente, no artigo 7, foi abordado o uso da função sort () da biblioteca table para a colocação de elementos em ordem crescente ou ascendente, mas nada foi dito sobre como proceder para a definição de uma ordem de disposição inversa.

Neste sentido o uso da função sort() é efetuado de maneira diferenciada. Observe o programa exemplo a seguir.

```
-- inicio do programa P11_03
  local VETOR = {}
  for I = 1, 5 do
    NOME = io.read()
    table.insert(VETOR, NOME)
  end
  for I = 1, 5 do
    print("["..I.."] = "..VETOR[I])
  end
  table.sort(VETOR,
    function(A, B) return A > B end)
  print()

  for I = 1, 5 do
    print("["..I.."] = "..VETOR[I])
  end

-- fim do programa P11_03
```

LUA – LINGUAGEM DE PROGRAMAÇÃO – PARTE 12

Em seguida escreva o código do programa em um editor de texto, gravando-o com o nome P11_03.lua e execute-o com a linha de comando lua 5.1 P11_03.lua.



Após entrar cinco nomes serão mostrados os nomes na ordem de entrada e ordenados de forma descendente.

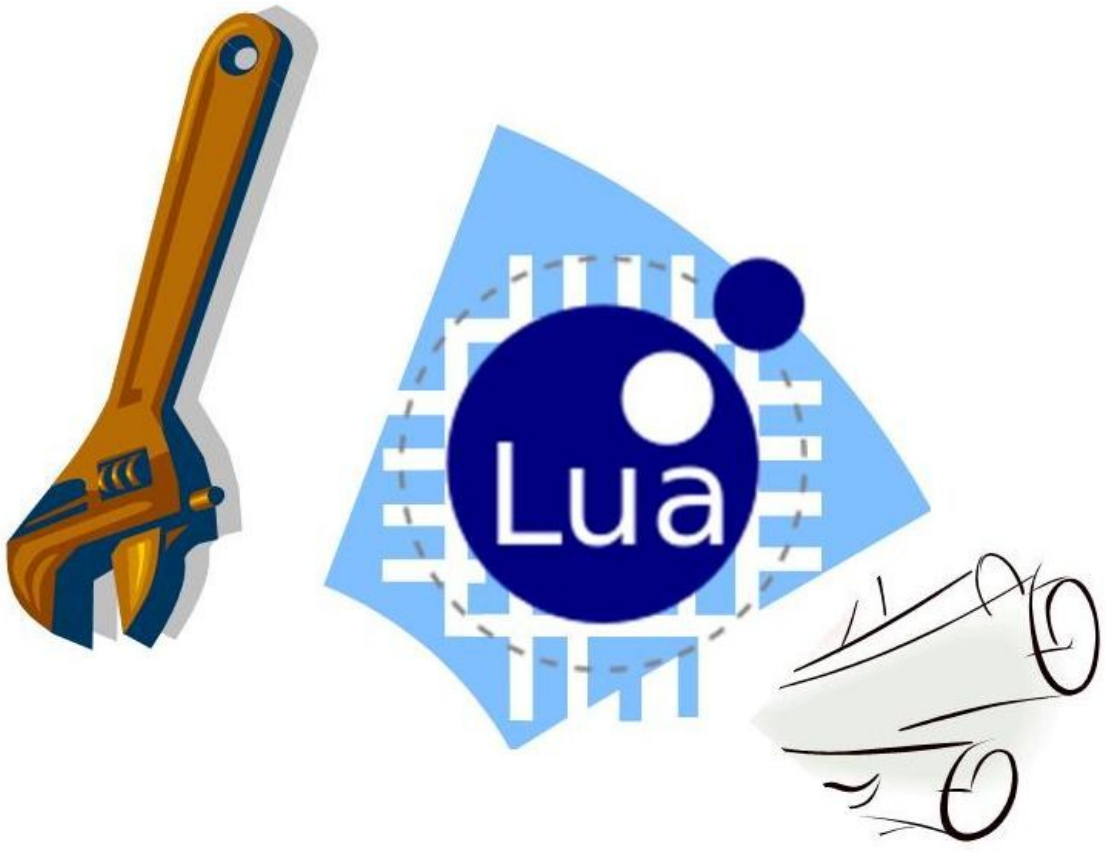
Para conseguir ordenar os elementos do vetor em ordem decrescente faz-se uso no segundo parâmetro de sort() de um recurso baseado em uma função em estilo lambda: function(A, B) return A > B end que recebe dois parâmetros e retorna o maior entre os dois valores fornecidos.

Conclusão

Neste artigo foi apresentado o uso de recursos para a definição de pausa, ciclos de tempo em segundos e ordenação de vetor em sentido decrescente/descendente.

Em outras ocasiões serão apresentados outras possibilidades de uso da linguagem Lua.

Até lá.



AUTOR

Escrito por Augusto Manzano

Como usar ler o ficheiro manifest usando o Cimbalino Windows Phone Toolkit

(usando MVVM)

Introdução

Cimbalino Windows Phone Toolkit é um conjunto de itens úteis e poderosos para ajudar na implementação de aplicações *Windows Phone*, especialmente nas aplicações que implementem o padrão *MVVM*. O projeto base do *toolkit* contém serviços para implementação do padrão de *MVVM*, conversores, classes auxiliares, métodos de extensões.

O Cimbalino.Phone.Toolkit.Background - é o projeto do *toolkit* contendo serviços para implementação do padrão de *MVVM* compatível com *background agents*. Este inclui o IApplicationManifestService - que representa a interface para o serviço com a capacidade de ler o ficheiro *manifest* da aplicação. A implementação é ApplicationManifestService.

Ecrã do Exemplo

Construindo o exemplo

O código fonte pode ser obtido em Exemplo ApplicationManifestService (**inclui exemplo para os vários targets**).

Os pacotes estão disponíveis em Nuget Package Manager (para ambos os "targets") e podem ser instalados.

O Exemplo

Registando serviços

Devemos começar por registar cada serviço no ViewModelLocator, como podemos ver de seguida:

```
///
/// This class contains static references to all
///the view models in the
/// application and provides an entry point for
///the bindings.
///
public class ViewModelLocator
{
    ///
    /// Initializes a new instance of the ViewMo-
delLocator class.
    ///
    public ViewModelLocator()
    {
        ServiceLocator.SetLocatorProvider(() =>
                                SimpleIoc.Default);

        if (!SimpleIoc.Default.IsRegistered())
        {
            SimpleIoc.Default.Register
                    <IApplicationManifestService,
                   ApplicationManifestService>();
        }
        SimpleIoc.Default.Register();
    }

    public MainViewModel MainViewModel
    {
        get
        {
            return ServiceLoca-
tor.Current.GetInstance();
        }
    }

    public static void Cleanup()
    {
        // TODO Clear the ViewModels
    }
}
```

Implementando a ViewModel

Em seguinda devemos implementar o MainViewModel, como podemos ver de seguida:

```
///
    /// This class contains properties that the
main View can data bind to.
    ///
    public class MainViewModel : ViewModelBase
    {
        ///
        /// The public application url.
        ///
        private readonly string _appUrl;
        ///
        /// The application manifest.
```

```
        ///
        private readonly ApplicationManifest
                                _applicationManifest;

        ///
        /// Initializes a new instance of the
        ///class.
        ///
        ///
        /// The email Compose Service.
        ///
        ///
        /// The application Manifest Service.
        ///
        ///
        /// The marketplace review service
        ///
        ///
        /// The share Link Service.
        ///
        public MainViewModel
                        (IApplicationManifestService
                         applicationManifestService)
        {
            _applicationManifest =
                         applicationManifestService.
                           GetApplicationManifest();
            _appUrl = string.Concat("http://
windowsphone.com/s?appid= This link is external to
     TechNet Wiki. It will open in a new window. ",
                  _applicationManifest.App.ProductId);
        }

        ///
        /// Gets the title.
        ///
        public string Title
        {
            get
            {
                return
                    _applicationManifest.App.Title;
            }
        }

        ///
        /// Gets the author.
        ///
        public string Author
        {
            get
            {
                return
                   _applicationManifest.App.Author;
            }
        }

        ///
        /// Gets the version.
        ///
        public string Version
        {
            get
            {
                return
                  _applicationManifest.App.Version;
            }
        }

        ///
        /// Gets the description.
        ///
        public string Description
        {
            get
            {
                return

                    _applicationManifest.App.Description;
                }
            }

            ///
            /// Gets the product ID.
            ///
            public string ProductID
            {
                get
                {
                    return

                      _applicationManifest.App.ProductId;
                }
            }

            ///
            /// Gets the publisher.
            ///
            public string Publisher
            {
                get
                {
                    return
                  _applicationManifest.App.Publisher;
                }
            }

            ///
            /// Gets the capabilities.
            ///
            public IList Capabilities
            {
                get
                {
                    return
  _applicationManifest.App.Capabilities.ToList();
                }
            }
        }
    }
```

Implementando a View

Para conetar a view model com a página devemos adicionar a ViewModelLocator no App.xaml:

```
<vm:ViewModelLocator x:Key="Locator"
                     :IsDataSource="True"  =""/>
```

e adicionar o binding na página principal:

```
DataContext="{Binding MainViewModel,
Source={StaticResource Locator}}"
```

A MainPage.xaml será algo do género:

```
<phone:PhoneApplicationPage
  x:Class="CimbalinoSample.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/
       xaml/presentation    xmlns:x="http://
                            schemas.microsoft.com/
         winfx/2006/xaml    xmlns:d="http://
                            schemas.microsoft.com/
      expression/blend/2008   xmlns:mc="http://
                          schemas.openxmlformats.org/
  markup-compatibility/2006.   xmlns:phone="
                    clr-namespace:Microsoft.Phone.
                  Controls;assembly=
                                  Microsoft.Phone"
  xmlns:shell="clr-namespace:Microsoft.
```

```
            Phone.Shell;assembly=Microsoft.Phone"
DataContext="{Binding MainViewModel,
    Source={StaticResource Locator}}"
FontFamily="{StaticResource
                        PhoneFontFamilyNormal}"
FontSize="{StaticResource PhoneFontSizeNormal}"
Foreground="{StaticResource
                        PhoneForegroundBrush}"
Orientation="Portrait"
SupportedOrientations="Portrait"
shell:SystemTray.IsVisible="True"
mc:Ignorable="d">

<!--  LayoutRoot is the root grid where all page
                        content is placed  -->
<Grid x:Name="LayoutRoot"
                      Background="Transparent">
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition Height="*" />
  </Grid.RowDefinitions>


  <!--  TitlePanel contains the name of the
                  application and page title  -->
  <StackPanel x:Name="TitlePanel"
              Grid.Row="0"
              Margin="12,17,0,28">
    <TextBlock Margin="12,0"
               Style="{StaticResource
                        PhoneTextTitle2Style}"
               Text="Cimbalino Sample" />
  </StackPanel>

  <!--  ContentPanel - place additional content
                                      here  -->
  <Grid x:Name="ContentPanel"
        Grid.Row="1"
        Margin="12,0,12,0" >
    <TextBlock TextWrapping="Wrap"
       VerticalAlignment="Top" Height="50">
      Title: <Run Text="{Binding Title}"/>
    </TextBlock>
    <TextBlock TextWrapping="Wrap"
       VerticalAlignment="Top" Margin="0,50,0,0">
      Author: <Run Text="{Binding Author}"/>
    </TextBlock>
    <TextBlock TextWrapping="Wrap"
       VerticalAlignment="Top"Margin="0,100,0,0">
      Version: <Run Text="{Binding Version}"/>
    </TextBlock>
    <TextBlock TextWrapping="Wrap"
      VerticalAlignment="Top" Margin="0,150,0,0">
      Description: <Run Text="{Binding
                                  Description}"/>
    </TextBlock>
    <TextBlock TextWrapping="Wrap"
       VerticalAlignment="Top"
                            Margin="0,300,0,0">
     ProductID: <Run Text="{Binding ProductID}"/>
    </TextBlock>
    <TextBlock TextWrapping="Wrap"
                      VerticalAlignment="Top"
                                  Margin="0,350,0,0">
       Publisher: <Run Text="{Binding Publisher}"/>
      </TextBlock>
      <TextBlock TextWrapping="Wrap"
                                  Text="Capabilities:"
                      VerticalAlignment="Top"
                                  Margin="0,400,0,0"/>
      <ListBox DisplayMemberPath="Name"
                                   Margin="0,450,0,0"
                 ItemsSource="{Binding
 Capabilities}"/>
    </Grid>

  </Grid>

</phone:PhoneApplicationPage>
```

Conclusão

Em conclusão, podemos constatar que através dos *toolkits* usados é muito simples obter a informação do ficheiro *manifest* e estes ajudam na implementação do padrão de desenvolvimento *MVVM*, simplificando o desenvolvimento das aplicações.

AUTOR

Escrito Por Sara Silva

Licenciada em Matemática – Especialidade em Computação, pela Universidade de Coimbra e é Microsoft Certified Profissional 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.

Classifique a App

Classificação é um processo importante no desenvolvimento de Apps. Utilizadores terão preferência por uma App com (boa) classificação. Este artigo tem como objetivo demonstrar como, em Apps desenvolvidas em C# para Windows 8 e Windows Phone 8, sugerir ao utilizador que classifique a App e envie o seu feedback após um dado número de utilizações.

O método consiste em gravar o número de vezes que o utilizador abre a aplicação, e, à quinta vez, sugerir ao utilizador que classifique a App. O código está desenvolvido para poder ser usado numa App Universal[1], mas poderá também ser utilizado em aplicações individuais.

Namespaces:

```
using System;                 //Uso Uri Email
using System.Threading.Tasks;//Funções Assín-
cronas
using Windows.System;         // Launcher para
abertura Store
using Windows.ApplicationModel.Store; // App ID
using Windows.UI.Popups;      // Caixas Mensagem
```

Iremos usar as seguintes variáveis:

Chaves que irão guardar informação do contador de aberturas e estado da Classificação:

```
public static string chaveContadorAberturas =
            "CLASSIFIQUE_APP_CONTADOR_ABERTURAS";
public static string chaveClassificada =
                  "CLASSIFIQUE_APP_CLASSIFICADA";
```

Variaveis que irão conter valor lidos e valores a serem gravados:

```
public static int contadorAberturas = 0;
public static bool classificada = false;
```

Para review, iremos usar uma função:

```
public async static void AppReview()
{
}
```

Começamos por verificar nas definições da App se foi classificada, usando um função GetSetting:

```
classificada = GetSetting<bool>
                      (chaveClassificada, false);
```

Função GetSetting:

```
public static T GetSetting<T>(string key,
                                  T defaultValue)
{
      if
        (Windows.Storage.ApplicationData.Current.
         LocalSettings.Values.ContainsKey(key))
      {
          return (T)
                 Windows.Storage.ApplicationData.
              Current.LocalSettings.Values[key];
      }
      else
      {
          return defaultValue;
      }
}
```

Usamos o Windows.Storage.ApplicationData para obter valores previamente guardados nas definições da App. Se a chave não existir, no caso de a App ser aberta pela primeira vez, devolve o valor por defeito, que deve ser introduzido na chamada da função. No caso da variável classificada, o valor por defeito será false.

Após leitura do valor, iremos verificar se a App foi classificada:

```
if (!classificada)
{
}
```

Se a App não tiver sido classificada, iremos ler contador aberturas:

```
contadorAberturas = GetSetting<int>
                        (chaveContadorAberturas, 0);
```

Incrementamos a abertura atual:

```
contadorAberturas ++;
```

Se a App tiver sido aberta 5 vezes[2], chamamos função review:

```
if (contadorAberturas == 5)
{
     await Review();
}
```

Por fim, guardamos o novo número de aberturas e estado da classificação, usando a função StoreSettings, similar à função GetSettings, mas usada para gravar definições:

```
StoreSetting(chaveContadorAberturas,
                                    contadorAberturas);
StoreSetting(chaveClassificada, classificada);
```

Função StoreSetting:

```
public static void StoreSetting(string key, object
value)
{
     Windows.Storage.ApplicationData.Current.
                LocalSettings.Values[key] = value;
}
```

Função AppReview completa:

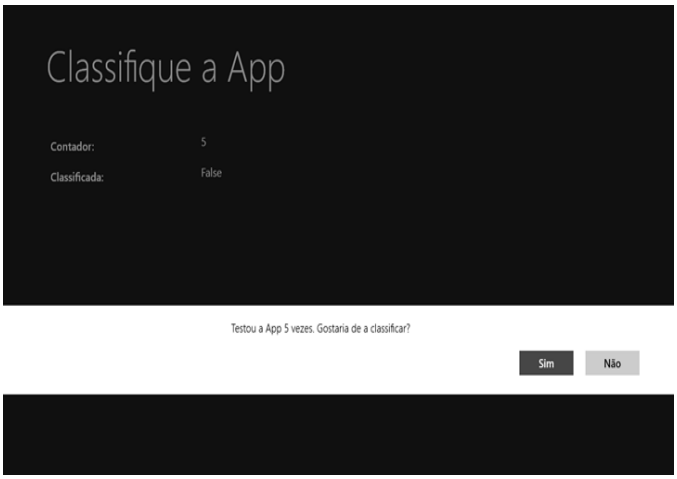
```
public async static void AppReview()
{
    //Variaveis que irão conter valor lidos e
    //valores a serem gravados
      classificada = GetSetting<bool>
                        (chaveClassificada, false);

      if (!classificada)
      {
           contadorAberturas = GetSetting<int>
```

```
(chaveContadorAberturas, 0); //Valor por defeito 0,
//se não tiver nenhum valor (caso app abertura
//primeira vez)
            contadorAberturas ++; // Acrescenta ao
                          //contador abertura atual

            if (contadorAberturas == 5)
            {
                  await Review();
            }

            StoreSetting(chaveContadorAberturas,
                                    contadorAberturas);
            StoreSetting(chaveClassificada,
                                       classificada);
      }
}
```

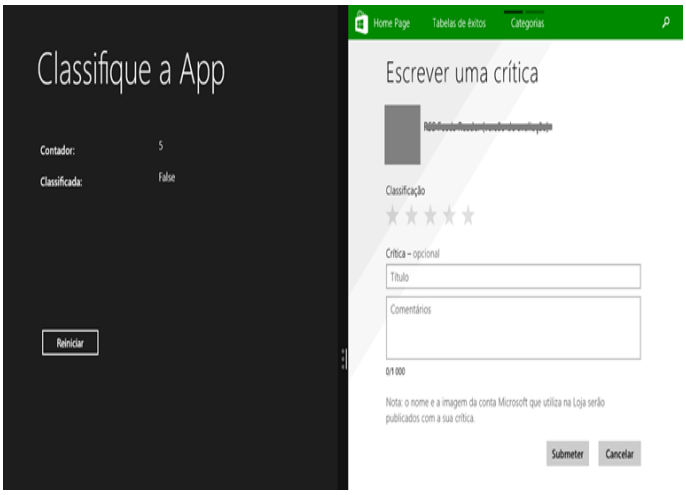
Função Review:

```
public static async Task<bool> Review()
{
   classificada = true;
   var messageDialog = new MessageDialog("Testou a
         App 5 vezes. Gostaria de a classificar?");
   messageDialog.Commands.Add(new UICommand
                                              ("Sim"));
   messageDialog.Commands.Add(new UICommand
                                              ("Não"));
   var messageResult = await
                          messageDialog.ShowAsync();

   if (messageResult.Label == "Sim")
   {

#if WINDOWS_PHONE_APP
    await Launcher.LaunchUriAsync(new
          Uri("ms-windows-store:reviewapp?appid=" +
                      CurrentApp.AppId.ToString()));
#else
    await Launcher.LaunchUriAsync(new Uri
                  ("ms-windows-store:review?PFN=" +
                      CurrentApp.AppId.ToString()));
#endif
    }

    else
    {

      var messageDialogFeedback = new MessageDialog
       ("Gostaria de nos dar o seu feedback sobre a
                                              App?");
       messageDialogFeedback.Commands.Add(new
                                  UICommand("Sim"));
       messageDialogFeedback.Commands.Add(new
                                 UICommand("Não "));
       var messageResultFeedback = await
                 messageDialogFeedback.ShowAsync();

       if (messageResultFeedback.Label == "Sim")
       {

            var mailTo = new Uri("mailto:?
  to=email@domain.com&subject=Classifique a
            App - App Feedback&body=Escreva aqui o
                                       seu feedback");

            await Windows.System.Launcher
                              .LaunchUriAsync(mailTo);
        }
     }

     return true;
   }
```

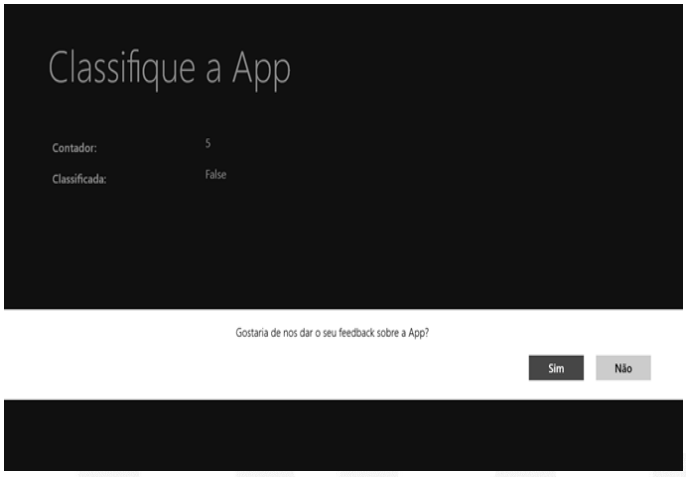
Na função Review, apresentamos uma mensagem para sugerir classificação.



Caso o utilizador queria classificar a App, obtemos o ID da nossa App[3], usando CurrentApp.AppId, e redirecionamos o utilizador para a Página da App na Windows Store, usando o protocolo Review[4].

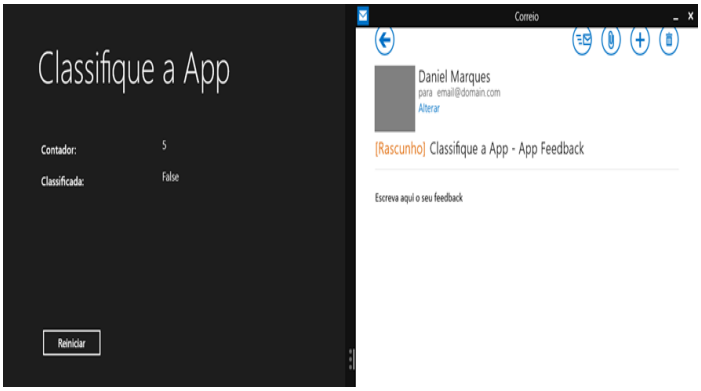


Caso o utilizador não queira classificar a App, convidamos a dar o seu feeback.



CLASSIFIQUE A APP

Caso o utilizador queira dar o seu Feedback, redirecionamos para um novo email (que será aberto no editor email predefinido), preenchendo com "De:" com email suporte, e Assunto.



Agora basta fazer uma chamada à função AppReview no LoadState da App. Como inseri a função no ficheiro na classe comum App, estou a chamar a função da seguinte forma:

```
App.AppReview();
```

Se usar numa App única, basta chamar função, não Universal:

```
AppReview();
```

[1] Apps desenvolvidas com target para Windows 8 e Windows Phone 8, com código partilhado. Mais informações: http://msdn.microsoft.com/en-us/library/windows/apps/dn609832.aspx

[2] Novas Apps Universais para Windows 8.1 e Windows Phone 8.1 usam Windows.Storage.ApplicationData. Para versões anteriores do Windows Phone, é usada função IsolatedStorageSettings.

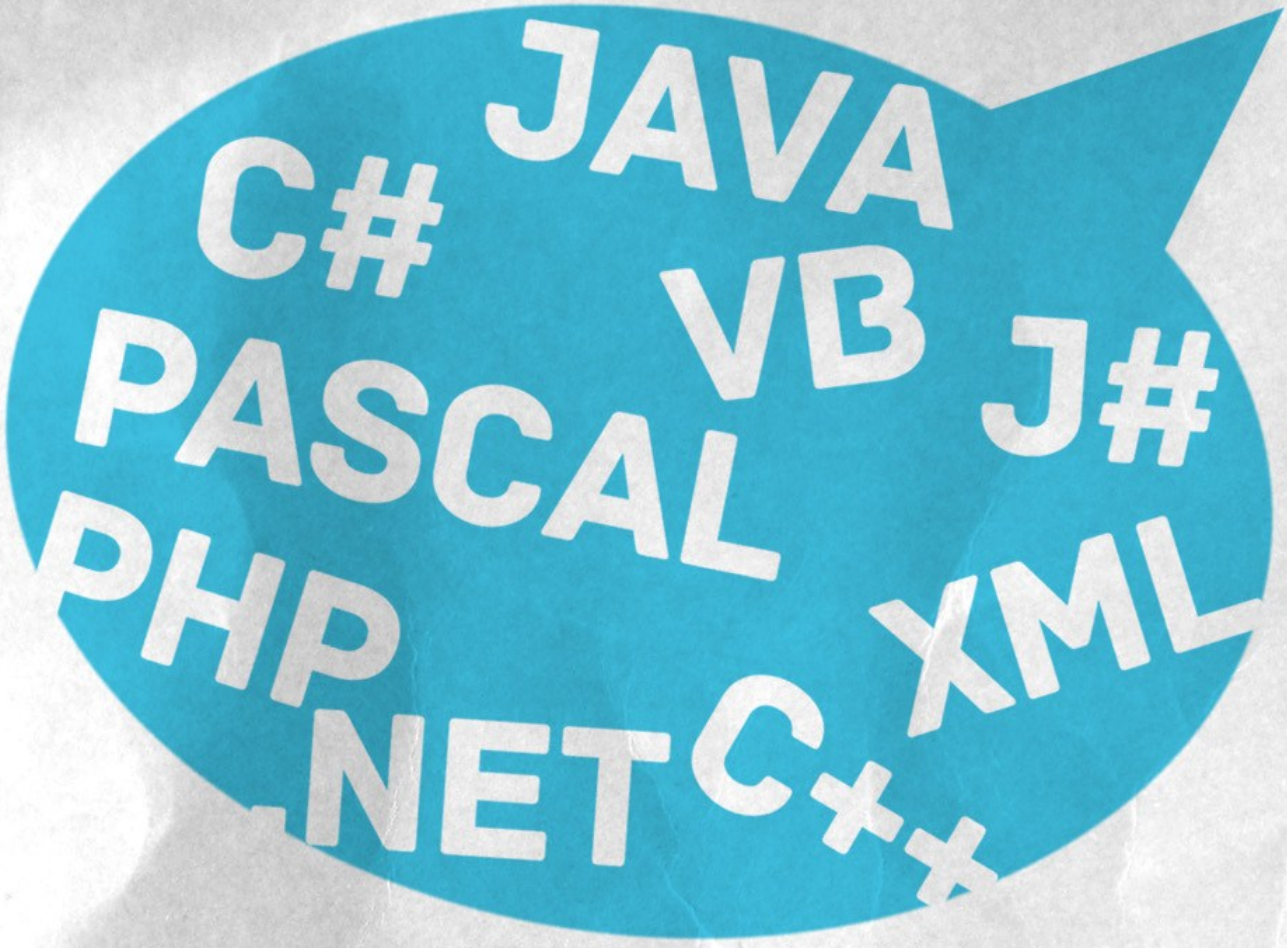
[3] O ID de uma App é atribuído durante publicação na Store.

[4] Mais detalhes sobre protocolos em: http://msdn.microsoft.com/en-us/library/windows/apps/hh974767.aspx

"Utilizadores terão preferência por uma App com (boa) classificação. Este artigo tem como objetivo demonstrar como, em Apps desenvolvidas em C# para Windows 8 e Windows Phone 8, sugerir ao utilizador que classifique a App e envie o seu feedback após um dado número de utilizações."

AUTOR

Escrito por Daniel Marques



ENTÃO, SÓ FALAS EM CÓDIGO?

TEMOS O REMÉDIO CERTO PARA TI!

portugal-a-programar.pt

A MAIOR COMUNIDADE PORTUGUESA DE PROGRAMAÇÃO, APARECE!

COLUNAS

Core Dump [13] - Ensino Delta Empresa

CORE DUMP [13] - Ensino Delta Empresa

A distância entre ensino e indústria é um tema real que merece atenção constante por parte da nossa sociedade.

Assistimos recentemente no nosso país a um ajustamento forçado, mas necessário, do número de vagas do ensino superior em relação às necessidades empresariais do país. Neste ajustamento, a área de tecnologia terá sido das menos afetadas. No entanto esse facto não garante que os recém licenciados venham preparados para aquilo que são as reais necessidades das empresas.

Antes de mais, **temos de ter em atenção que o objetivo do ensino superior não é, nem deve ser, totalmente alinhado com as necessidades das empresas**. A razão pelo qual faço esta afirmação prende-se com o facto de que, se os recém licenciados souberem apenas fazer o que as empresas necessitam, então terão sido privados de um número considerável de exposições a diferentes formas de pensar, de resolver problemas, de diferentes tecnologias e inúmeros desafios. Seriam como Chaplin no filme "*Tempos Modernos*", apenas saberiam "apertar porcas" e dificilmente poderiam pensar ou executar de forma diferente, limitando assim a evolução e a inovação. Ou seja, **o ensino superior deve ter o seus próprios objetivos e seguir o seu próprio caminho de forma a que a investigação, experimentação e inovação não sejam limitadas pelo pensamento único de "formar pessoas para as empresas"**.

Filme: Tempos Modernos. United Artists.

No entanto, e porque **um dos objetivos do ensino superior é garantir aos seus recém licenciados um mínimo de conhecimento e competências que lhes permita ingressar no mercado de trabalho de forma produtiva**, torna-se imprescindível que os currículos abordem tópicos que farão parte da realidade profissional dos seus formandos.

Na outra ponta, consumidora dos recém formados, estão as organizações, que, infelizmente, não raras vezes recrutam recém licenciados que consideram "mal preparados". Esta situação força as empresas a investir algum tempo na formação dos seus novos profissionais até que estes comecem efetivamente a serem produtivos. **Esta realidade ficou mais exposta aquando da entrada de Bolonha, onde algumas empresas optaram por deixar de recrutar recém licenciados e passaram apenas a recrutar mestrados.**

Esta queixa por parte das empresas não é uma falha do ensino, é apenas uma consequência dos conhecimentos académicos e do seu alinhamento, ou não, com as necessidades empresariais aquando do recrutamento. **Não nos podemos esquecer que todos os ensinamentos académicos devem ser vistos como ferramentas úteis para todo o percurso profissional das pessoas e que se tornam mais valias para as organizações.** Ou seja, não podemos ver este desalinhamento inicial como um problema dado que o conhecimento e mais valia dos recém licenciados não se esgota na contratação inicial. Como costumo dizer, isto não é uma corrida de 100 metros, é uma maratora.

Como qualquer ponto de equilibrio, este entre os objetivos académicos e as necessidades das empresas não é fácil de descobrir, requerendo uma análise constante e um esforço de aproximação entre universidades e empresas. Este esforço tem vindo a ser feito ao longo dos últimos anos, e **atualmente temos algumas estratégias inovadoras e interessantes, como programas de estágio especiais logo após o primeiro ano para alunos com médias superiores a 15 valores**.

Este tipo de abordagem é extremamente interessante, pois dá aos alunos uma motivação extra e uma aprendizagem mais diversificada e valiosa, ao mesmo tempo que permite às empresas conhecer e criar relações com os melhores alunos para uma potencial oferta de primeiro emprego.

Este é, claramente, um tópico de discussão para o qual não há respostas certas, pelo que a procura do ponto de equilibrio deverá ser constante e a criação de estratégias para incluir componentes que preparem melhor os alunos para a vida profissional deverá ser uma prioridade.

AUTOR

Escrito por Fernando Martins
Faz parte da geração que se iniciou nos ZX Spectrum 48K. Tem um Mestrado em Informática e mais de uma década de experiência profissional nas áreas de Tecnologias e Sistemas de Informação. Criou a sua própria consultora sendo a sua especialidade a migração de dados.

Media Partners da Revista PROGRAMAR

Análises

Desenvolvimento em iOS iPhone, iPad e iPod Touch – Curso Completo (3. Edição Atualizada)

Review

Desenvolvimento em iOS iPhone, iPad e iPod Touch – Curso Completo (3. Edição Atualizada)

Título: Desenvolvimento em iOS iPhone, iPad e iPod Touch – Curso Completo (3.ª Edição Atualizada)

Autor: Nuno Fonseca, Catarina Reis, Catarina Silva, Luis Marcelino, Vitor Carreira

Editora: FCA - Editora de Informática, Lda.

Páginas: 448

ISBN: 978-972-722-786-0

O livro que me foi apresentado para análise intitulado *Desenvolvimento em iOS iPhone, iPad e iPod Touch - Curso Completo (edição revista e atualizada para iOS 7)* tem como público-alvo todo o leitor interessado em começar a desenvolver aplicações móveis para as plataformas *iOS* da *Apple* ou, para leitores mais experientes, "afinar" alguns aspectos específicos de programação em *Objective-C* para *iOS*. De qualquer forma, é assumido neste livro que o leitor tenha conhecimentos prévios de programação, mais especificamente da linguagem *C*, e dos conceitos inerentes à programação orientada aos objetos (*POO*).

Desde o primeiro capítulo os autores optam por uma abordagem prática, aliás, passo a citar, "extremamente prática", desafiando o leitor a desenvolver uma aplicação móvel *iOS* (*app*) denominada "O Meu Diário" como forma de apresentar o funcionamento base de uma aplicação *iOS* tipo bem como ter uma perspectiva inicial sobre o respectivo ambiente integrado de desenvolvimento (*IDE*)---o *Xcode*.

Este livro começa assim por apresentar uma breve resenha história da plataforma *iOS* e respectivo *kit* de desenvolvimento de software (*SDK*), contextualizando o legado da linguagem *C* e da empresa *NeXT* (segunda empresa criada por Steve Jobs logo após ter sido afastado da *Apple* em 1985), e dos dispositivos móveis associados: o *iPhone*, *iPod Touch* e *iPad*. Logo desde o primeiro capítulo, o leitor aprende o que são *IBOutlets* e *IBActions* e para que servem, e à medida que o leitor vai criando a sua primeira *app*. Os autores aproveitam para também explicar como funciona e está organizado todo o ambiente de desenvolvimento *Xcode*. É também aqui logo apresentado (no primeiro capítulo) uma breve referência ao padrão arquitetural *Model-View-Controller* (*MVC*) que qualquer aplicação *iOS* deve seguir, aproveitando os autores para introduzir o conceito de *Storyboards*. Assim, o leitor no fim do primeiro capítulo tem uma *app* a funcionar e o conhecimento de alguns dos mais importantes conceitos no desenvolvimento *iOS*.

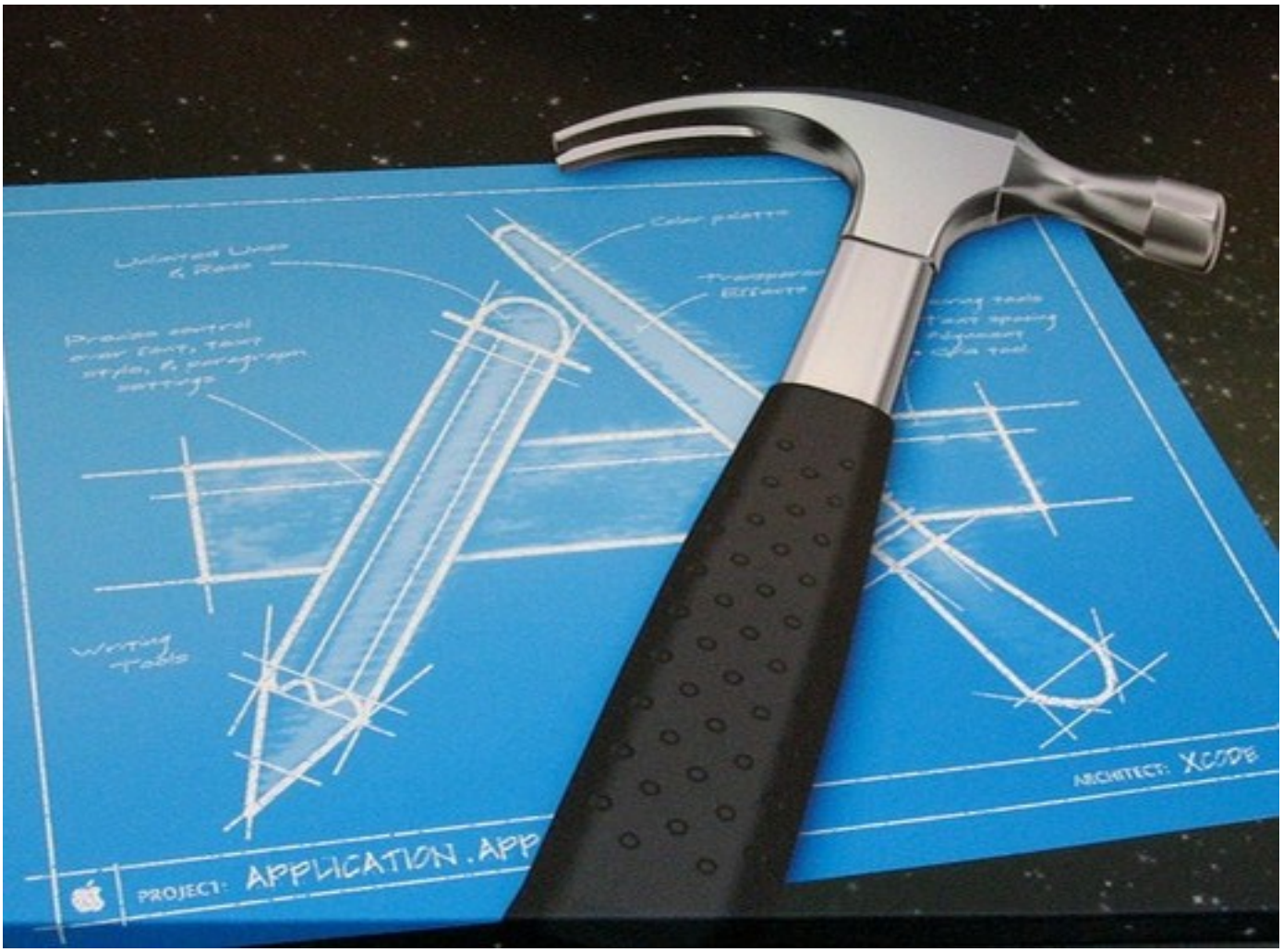
O segundo e terceiro capítulos apresentam, respectivamente, os conceitos Base e Avançados da linguagem de programação *Objective-C* (linguagem de programação nativa para a plataforma *iOS*). Para o leitor iniciado o segundo capítulo é de leitura obrigatória sugerindo-se, no entanto, que consulte apenas o terceiro capítulo à medida que vai precisando. Para o leitor mais experiente, o terceiro capítulo é sem dúvida uma fonte bastante rica e profícua de informação. É aqui apresentado de forma bastante aprofundada e exaustiva a questão dos mecanismos de Gestão de Memória da linguagem de programação *Objective-C* para *iOS*, onde o Ciclo de Vida de uma *app iOS* é introduzido e os conceitos de protocolos e *delegates* bem explicados (estes dois últimos conceitos também bastante importantes e a saber, mesmo para o leitor iniciado, dado tratar-se em mais um dos padrões de desenho muito usado em *iOS*). Ainda neste capítulo são aprofundados outros conceitos, já apresentados no capítulo anterior, até um nível de exaustão exemplar.

A principal *framework* utilizada no desenvolvimento de uma qualquer aplicação para a plataforma *iOS Cocoa Touch* constituída pelas *frameworks Foundation* + *UIKit* (sobre as quais assentam todas as restantes *frameworks* da plataforma *iOS*), é apresentada em termos das suas principais classes a saber, respectivamente, nos capítulo 4 e 5.
A *framework Foundation* fornece as principais classes para a representação de cada tipo de dados elementares e a *framework UIKit* fornece as classes necessárias para construir e gerir toda a interação com o utilizador de uma qualquer aplicação *iOS*. Cada classe é apresentada segundo os seus principais objectivos e métodos, e sempre acompanhada de um pequeno exemplo de utilização. As principais melhorias introduzidas desde a versão 6 do *iOS* são sempre focadas e explicadas. O Ciclo de Vida e a anatomia de uma qualquer aplicação *iOS*, bem como os principais padrões de desenho já atrás referenciados e apresentados, são aqui explicados ao pormenor (capítulo 5). A implementação prática da *app* "O Meu Diário" é aqui retomada e algumas questões pertinentes que a *Apple* muito valoriza e obriga os programadores *iOS* a seguir em termos de *guidelines* e boas práticas, relacionadas com Usabilidade e toda a eXperiência de Utilização (UX), são expostas e explicadas.

Os principais controladores de navegação bem como a técnica de *storyboards*, esta última introduzida desde a versão *iOS 5*, são exemplificados como devem ser utilizados no capítulo 6, e mais uma vez como novas funcionalidades a acrescentar na *app* "O Meu Diário". De modo similar, no capítulo 7, são apresentadas mais duas *frameworks* utilizadas em qualquer aplicação baseada em localização quer para ter acesso à informação gerada pelos sensores de localização quer para apresentar visualmente as localizações num mapa ---*Core Location* e *MapKit*. Os cuidados a ter com a rotação da interface gráfica bem como a leitura dos sensores de movimento são aflorados no capítulo 8. As principais técnicas

de armazenamento local de informação (Persistência) e na nuvem (o *iCloud*), bem como a *framework* específica *Core Data* (recomendada só para situações em que existe um volume significativo de dados a persistir), são apresentados em dois capítulos separados. Todas as questões relacionadas com o Multimédia (Som e Imagem, Desenho, Gestos e Animação), Execução Concorrente (onde a *API Grand Central Dispatcher* introduzida na versão 5 do *iOS*) e Serviços de Rede, técnica imprescindível a saber com a proliferação dos serviços de *Cloud-Computing* para *mobile* (conhecidos por *mobile Backend as a Service---mBaaS*) e com a utilização massiva das redes sociais, são também descritos com o máximo rigor seguindo uma estratégia muito prática e sempre com bons exemplos. Por fim, o livro termina com uma explicação de como uma *app* deve ser preparada para ser submetida com sucesso na *App Store* bem como todo o processo muito rígido que *Apple* nos obriga deve ser seguido.

Em suma, este livro é sem dúvida uma excelente referência e um manual completo para quem quer desenvolver aplicações para a plataforma *iOS*, com um português simples, apelativo e cientificamente correto. Todos as principais fases no desenvolvimento *iOS* são abordadas com o máximo rigor, explicadas e logo exemplificadas, onde o leitor tem o prazer de "aprender fazendo", aconselhando veementemente a leitura desta obra para qualquer profissional e estudantes da área.



AUTOR

Escrito por Vitor Ferreira

Vítor Manuel ferreira, Licenciado em Engenharia Electrónica e Telecomunicações pela Universidade de Aveiro (1994), Mestre em Engenharia Electrónica e Telecomunicações pela Universidade de Aveiro - área de Processamento de Sinal (1999), Licenciado em Ensino da Música pela Universidade de Aveiro - com profissionalização em Clarinete (2006) é Doutorando em Multimédia em Educação na Universidade de Aveiro. As suas áreas de interesse são: Computação móvel/ubíqua aplicada, mobile-learning e iOS. Atualmente é docente no Instituto Politécnico de Viana do Castelo (IPVC) onde leciona unidades curriculares na área dos Sistemas Operativos, Redes e Tecnologias de Informação.

COMUNIDADES

Comunidade NetPonto – Autenticação usando a conta do Facebook, Google e Microsoft numa app de WP8.0

Autenticação usando a conta do Facebook, Google e Microsoft numa app de WP8.0

Este artigo tem como objectivo mostrar como conectar uma aplicação Windows Phone 8.0 à conta do Facebook, Microsoft e Google, para que com isto seja validada a autenticação na aplicação. O artigo irá portanto, focar-se nas funcionalidades de *login* e *logout*.

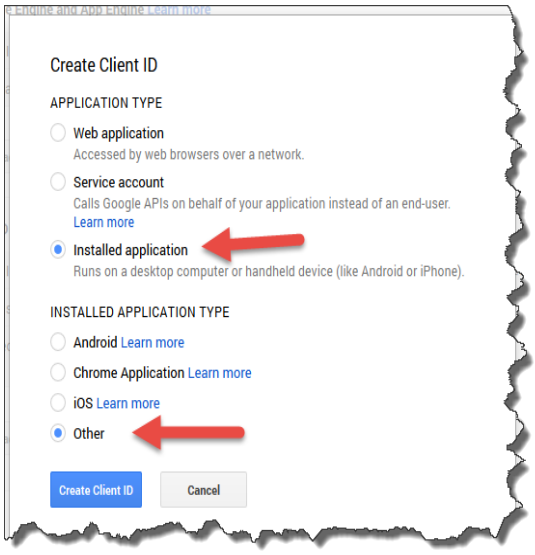
O exemplo do artigo usa MVVM Light Toolkit e Cimbalino Windows Phone Toolkit, no entanto está fora do âmbito do artigo explicar como se usa estes *toolkits*, para mais informações por favor consulte as seguintes referências: Artigos sobre MVVM Light e Artigos sobre Cimbalino.

No exemplo iremos usar as seguintes referências:

- Facebook SDK for Windows Phone (http://facebooksdk.net/docs/phone/)
- Google APIs Auth Client e Google APIs OAuth2 Client (https://www.nuget.org/packages/Google.Apis.Auth/ e https://www.nuget.org/packages/Google.Apis.Authentication/1.6.0-beta)
- Live SDK (http://msdn.microsoft.com/en-US/onedrive/dn630256) (para aceder à conta Microsoft)

De salientar que cada *provider* requer que seja atribuído um *app id/client id/client secret,* informação esta que é obtida nas seguintes referências:

Para a conta do Google deve aceder à referência https://console.developers.google.com/project e de seguida deve criar um novo projecto (*APIs & auth* > *credentials*) com as seguintes características:



- Para a conta do Facebook deve aceder à referência https://developers.facebook.com/ e criar uma nova *app*.
- Para o Live SDK, deve aceder à referência https://account.live.com/developers/applications/index e de seguida criar uma nova *app* ou usar uma *app* já existente.

Antes de começar, deve-se definir a classe Constants com as várias keys, sem esta informação o exemplo não irá funcionar!

```
public class Constants
  {
      public const string FacebookAppId = "";

      public const string GoogleClientId = "";

      public const string GoogleTokenFileName =
                          "Google.Apis.Auth.OAuth2.
                      Responses.TokenResponse-user";

      public const string GoogleClientSecret = "";

      public const string MicrosoftClientId = "";

      ...
  }
```

De seguida iremos ver como nos podemos conectar às várias contas. Para ajudar a abstrair a autenticação, foi criada uma classe SessionService cujo principal objectivo é gerir as operações de *login* e *logout* para um determinado *provider*. Isto é interessante porque na página LoginView são definidos os vários botões de autenticação e para cada um deles é atribuído um Command que irá lançar a acção e é enviado um CommandParamater que contém o nome do *provider* que será utilizado no SessionService e desta forma a página LoginView e a classe LoginViewModel serão mais simples e claras. Outra questão também relevante, é o facto que, depois da autenticação para cada provider, poder haver a necessidade de fazer um pedido de autorização no meu *backend* para aceitar o utilizador e com o SessionService apenas tenho que adicionar essa validação uma vez, em vez de ter que o fazer em cada *provider* (o que na realidade não faz muito sentido).

As classes criadas foram:

- FacebookService contém todo o código necessário para efetuar a autenticação usando uma conta Facebook;
- MicrosoftService contém todo o código necessário para efetuar a autenticação usando uma conta Microsoft;

- GoogleService contém todo o código necessário para efetuar a autenticação usando uma conta Google;
- SessionService chama os métodos de login e logout para cada provider;
- LoginView representa a página para efetuar a autenticação;
- LoginViewModel representa a view model da página LoginView.

Vejamos agora o código para cada classe.

A classe FacebookService será:

```
public class FacebookService : IFacebookService
    {
        private readonly ILogManager _logManager;
        private readonly FacebookSessionClient
                            _facebookSessionClient;

        public FacebookService(ILogManager
                                        logManager)
        {
            _logManager = logManager;
            _facebookSessionClient = new
  FacebookSessionClient(Constants.FacebookAppId);
        }

        public async Task LoginAsync()
        {
            Exception exception;
            Session sessionToReturn = null;
            try
            {
                var session = await
                _facebookSessionClient.LoginAsync
                   ("user_about_me,read_stream");
                sessionToReturn = new Session
                {
                    AccessToken =
                             session.AccessToken,
                    Id = session.FacebookId,
                    ExpireDate = session.Expires,
                    Provider =
                       Constants.FacebookProvider
                };

                return sessionToReturn;
            }
            catch (InvalidOperationException)
            {
                throw;
            }
            catch (Exception ex)
            {
                exception = ex;
            }
            await _logManager.LogAsync
                                    (exception);
            return sessionToReturn;
        }

        public async void Logout()
        {
            Exception exception = null;
            try
            {
                _facebookSessionClient.Logout();

                // limpa os cookies do browser
                await new WebBrowser().
                                ClearCookiesAsync();
            }
            catch (Exception ex)
            {
                exception = ex;
            }
            if (exception != null)
            {
                await _logManager.LogAsync
(exception);
            }
        }
    }
```

Um leitor mais atento, deve ter reparado que a seguir de fazer *logout*

```
_facebookSessionClient.Logout();
```

foi feita a limpeza dos *cookies* do *browser*, isto não é mais do que um *workaround*, para que da próxima vez que se tente autenticar com a conta do Facebook, o *browser* não use a conta anterior.

A classe GoogleService será:

```
public class GoogleService : IGoogleService
    {
        private readonly ILogManager _logManager;
        private readonly IStorageService
                                    _storageService;
        private UserCredential _credential;
        private Oauth2Service _authService;
        private Userinfoplus _userinfoplus;

        public GoogleService(ILogManager
        logManager, IStorageService storageService)
        {
            _logManager = logManager;
            _storageService = storageService;
        }

        public async Task LoginAsync()
        {
            Exception exception = null;
            try
            {
                // Oauth2Service.
                                //Scope.UserinfoEmail
                _credential = await
    GoogleWebAuthorizationBroker.AuthorizeAsync(new
                                        ClientSecrets
                {
                    ClientId =
                          Constants.GoogleClientId,
                    ClientSecret =
                       Constants.GoogleClientSecret
                }, new[] { Oauth2Service.
                   Scope.UserinfoProfile }, "user",
                            CancellationToken.None);

                var session = new Session
                {
                    AccessToken =
                     _credential.Token.AccessToken,
                    Provider =
                          Constants.GoogleProvider,
                    ExpireDate =

        _credential.Token.ExpiresInSeconds != null
                             ? new DateTime
        (_credential.Token.ExpiresInSeconds.Value)
```

```
                        : DateTime.Now.AddYears(1),
                    Id = string.Empty
                };

                return session;
            }
            catch (TaskCanceledException
                            taskCanceledException)
            {
                throw new InvalidOperationException
        ("Login canceled.", taskCanceledException);
            }
            catch (Exception ex)
            {
                exception = ex;
            }
            await _logManager.LogAsync(exception);
            return null;
        }

        public async Task GetUserInfo()
        {
            _authService = new Oauth2Service(new
                    BaseClientService.Initializer()
            {
                HttpClientInitializer =
                                        _credential,
                ApplicationName =
                     AppResources.ApplicationTitle,
            });
            _userinfoplus = await
  _authService.Userinfo.V2.Me.Get().ExecuteAsync();

            return _userinfoplus;
        }

        public async void Logout()
        {
            await new WebBrowser().
                                ClearCookiesAsync();
            if (_storageService.FileExists
                    (Constants.GoogleTokenFileName))
            {
                _storageService.DeleteFile
                    (Constants.GoogleTokenFileName);
            }
        }
    }
```

Mais uma vez teve-se que criar um *workaround* para efetuar o *logout*, isto porque o GoogleWebAuthorizationBroker não contém um método de *logout* para ser chamado. A solução passa por limpar os *cookies* no *browser* e apagar o ficheiro com os dados da sessão que é guardado no *storage*.

A classe MicrosoftService será:

```
public class MicrosoftService : IMicrosoftService
    {
        private readonly ILogManager _logManager;
        private LiveAuthClient _authClient;
        private LiveConnectSession _liveSession;

      ///
      /// Defines the scopes the application needs.
      ///
        private static readonly string[] Scopes =
  { "wl.signin", "wl.basic", "wl.offline_access" };

      ///
      /// Initializes a new instance of the  class.
      ///
      ///
      /// The log manager.
      ///
    public MicrosoftService(ILogManager logManager)
        {
            _logManager = logManager;
        }

        ///
        /// The login async.
        ///
        ///
        /// The  object.
        ///

      public async Task LoginAsync()
        {
            Exception exception = null;
            try
            {
                _authClient = new LiveAuthClient
                     (Constants.MicrosoftClientId);
                var loginResult = await
                _authClient.InitializeAsync(Scopes);
                var result = await
                     _authClient.LoginAsync(Scopes);
                if (result.Status ==
                LiveConnectSessionStatus.Connected)
                {
                    _liveSession =
                                loginResult.Session;
                    var session = new Session
                    {
                        AccessToken =
                        result.Session.AccessToken,
                        ExpireDate =
                   result.Session.Expires.DateTime,
                        Provider =
                       Constants.MicrosoftProvider,
                    };

                    return session;
                }
            }
            catch (LiveAuthException ex)
            {
                throw new InvalidOperationException
                            ("Login canceled.", ex);
            }
            catch (Exception e)
            {
                exception = e;
            }

            await _logManager.LogAsync(exception);
            return null;
        }

        ///
        /// The logout.
        ///

      public async void Logout()

        {
            if (_authClient == null)
            {
                _authClient = new LiveAuthClient
                     (Constants.MicrosoftClientId);
                var loginResult = await
                _authClient.InitializeAsync(Scopes);
            }

            _authClient.Logout();
        }
    }
```

A classe SessionService será:

```
public class SessionService : ISessionService
    {
     private readonly IApplicationSettingsService
                                _applicationSettings;
     private readonly IFacebookService
     _facebookService;
     private readonly IMicrosoftService
     _microsoftService;
     private readonly IGoogleService
     _googleService;
     private readonly ILogManager
     _logManager;
        public SessionService
(IApplicationSettingsService applicationSettings,
            IFacebookService facebookService,
            IMicrosoftService microsoftService,
            IGoogleService googleService,
                           ILogManager logManager)
        {
            _applicationSettings =
                               applicationSettings;
            _facebookService = facebookService;
            _microsoftService = microsoftService;
            _googleService = googleService;
            _logManager = logManager;
        }

        public Session GetSession()
        {
            var expiryValue = DateTime.MinValue;
            string expiryTicks =
 LoadEncryptedSettingValue("session_expiredate");
            if (!string.IsNullOrWhiteSpace
                                     (expiryTicks))
            {
                long expiryTicksValue;
                if (long.TryParse(expiryTicks,
                            out expiryTicksValue))
                {
                    expiryValue = new DateTime
                              (expiryTicksValue);
                }
            }

            var session = new Session
            {
                AccessToken =
      LoadEncryptedSettingValue("session_token"),
                Id = LoadEncryptedSettingValue
                                   ("session_id"),
                ExpireDate = expiryValue,
                Provider =
    LoadEncryptedSettingValue("session_provider")
            };

            _applicationSettings.Set
                    (Constants.LoginToken, true);
            _applicationSettings.Save();
            return session;
        }

        private void Save(Session session)
        {
            SaveEncryptedSettingValue
          ("session_token", session.AccessToken);
            SaveEncryptedSettingValue
                        ("session_id", session.Id);
            SaveEncryptedSettingValue
                           ("session_expiredate",
                session.ExpireDate.Ticks.ToString
                 (CultureInfo.InvariantCulture));
            SaveEncryptedSettingValue
("session_provider", session.Provider);
            _applicationSettings.Set
                      (Constants.LoginToken, true);
            _applicationSettings.Save();
        }

        private void CleanSession()
        {
            _applicationSettings.Reset
                                ("session_token");
            _applicationSettings.Reset
                                  ("session_id");
            _applicationSettings.Reset
                               ("session_expiredate");
            _applicationSettings.Reset
                                ("session_provider");
            _applicationSettings.Reset
                               (Constants.LoginToken);
            _applicationSettings.Save();
        }

        public async Task LoginAsync(string
                                            provider)
        {
            Exception exception = null;
            try
            {
                Session session = null;
                switch (provider)
                {
                    case
                        Constants.FacebookProvider:
                        session = await
                     _facebookService.LoginAsync();
                        break;
                    case
                       Constants.MicrosoftProvider:
                        session = await
                    _microsoftService.LoginAsync();
                        break;
                    case Constants.GoogleProvider:
                        session = await
                       _googleService.LoginAsync();
                        break;
                }
                if (session != null)
                {
                    Save(session);
                }

                return true;
            }
            catch (InvalidOperationException e)
            {
                throw;
            }
            catch (Exception ex)
            {
                exception = ex;
            }
            await _logManager.LogAsync(exception);

            return false;
        }

        public async void Logout()
        {
            Exception exception = null;
            try
            {
                var session = GetSession();
                switch (session.Provider)
                {
                    case
                        Constants.FacebookProvider:
                         _facebookService.Logout();
```

```
                        break;
                    case
                       Constants.MicrosoftProvider:
                        _microsoftService.Logout();
                        break;
                    case Constants.GoogleProvider:
                        _googleService.Logout();
                        break;
                }
                CleanSession();
            }
            catch (Exception ex)
            {
                exception = ex;
            }
            if (exception != null)
            {
                await _logManager.LogAsync
                                        (exception);
            }
        }

        private string LoadEncryptedSettingValue
                                        (string key)
        {
            string value = null;

            var protectedBytes =
             _applicationSettings.Get<byte[]>(key);
            if (protectedBytes != null)
            {
                byte[] valueBytes =
    ProtectedData.Unprotect(protectedBytes, null);
                value = Encoding.UTF8.GetString
                (valueBytes, 0, valueBytes.Length);
            }

            return value;
        }

        private void SaveEncryptedSettingValue
                        (string key, string value)
        {
            if (!string.IsNullOrWhiteSpace(key)
              && !string.IsNullOrWhiteSpace(value))
            {
                byte[] valueBytes =
                     Encoding.UTF8.GetBytes(value);

                // Encrypt the value by using the
                //Protect() method.
                byte[] protectedBytes =
           ProtectedData.Protect(valueBytes, null);
                _applicationSettings.Set(key,
                                        protectedBytes);
                _applicationSettings.Save();
            }
        }
    }
```

Uma vez que as classes relacionadas com a autenticação estão criadas, agora passamos para a parte da interface com o utilizador e para isso iremos criar o LoginViewModel e a página LoginView.

A classe LoginViewModel será:

```
public class LoginViewModel : ViewModelBase
    {
        private readonly ILogManager _logManager;
        private readonly IMessageBoxService
                                        _messageBox;
        private readonly INavigationService
                                _navigationService;
        private readonly ISessionService
                                        _sessionService;
        private bool _inProgress;

        public LoginViewModel(INavigationService
                                    navigationService,
            ISessionService sessionService,
            IMessageBoxService messageBox,
            ILogManager logManager)
        {
            _navigationService = navigationService;
            _sessionService = sessionService;
            _messageBox = messageBox;
            _logManager = logManager;
            LoginCommand = new RelayCommand
                                        (LoginAction);
        }

        public bool InProgress
        {
            get { return _inProgress; }
            set { Set(() => InProgress, ref
                            _inProgress, value); }
        }

        public ICommand LoginCommand { get; private
                                            set; }

        private async void LoginAction(string
                                        provider)
        {
            Exception exception = null;
            bool isToShowMessage = false;
            try
            {
                InProgress = true;
                var auth = await
                  _sessionService.LoginAsync(provider);
                if (!auth)
                {
                    await _messageBox.ShowAsync
(AppResources.LoginView_LoginNotAllowed_Message,
                        AppResources.
                                MessageBox_Title,
                        new List
                        {
                            AppResources.Button_OK
                        });
                }
                else
                {
                    _navigationService.NavigateTo
(new Uri(Constants.MainView, UriKind.Relative));
                }

                InProgress = false;
            }
            catch (InvalidOperationException e)
            {
                InProgress = false;
                isToShowMessage = true;
            }
            catch (Exception ex)
            {
                exception = ex;
            }
            if (isToShowMessage)
            {
                await _messageBox.ShowAsync
                  (AppResources.LoginView_AuthFail,
            AppResources.ApplicationTitle, new List
                        { AppResources.Button_OK });
            }
            if (exception != null)
            {
                await _logManager.LogAsync
```

```
(exception);
            }
        }
    }
```

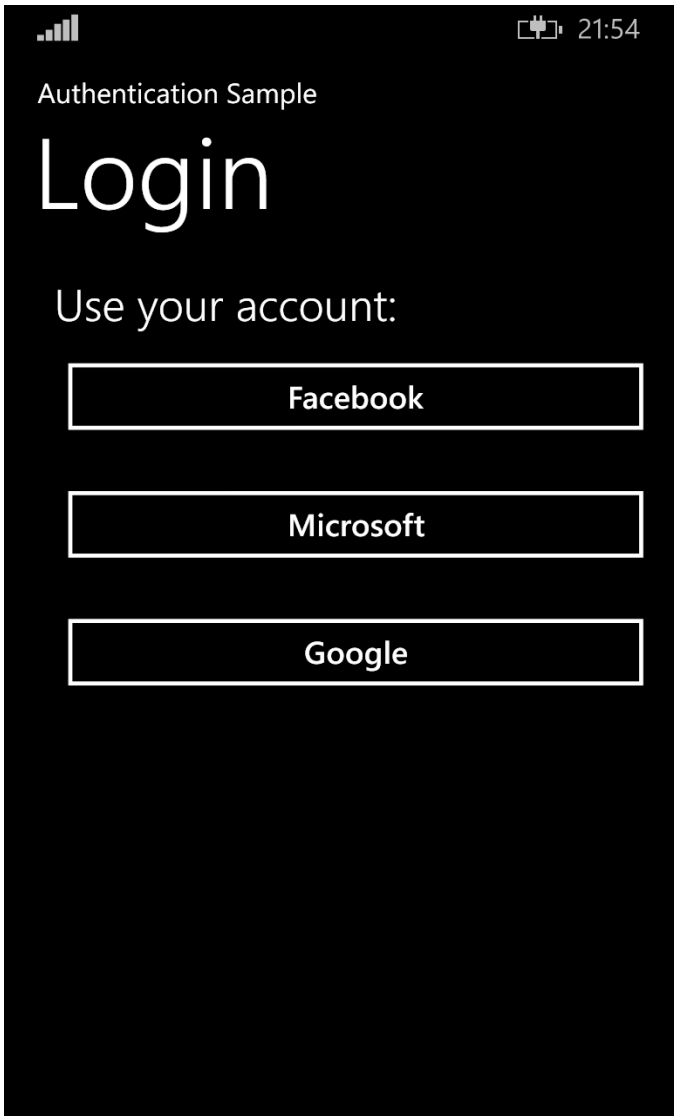
De salientar que no método LoginAction o parâmetro recebido é o valor do CommandParameter definido no LoginCommand (isto é definido na UI).

A página LoginView.xaml será:

```
<phone:PhoneApplicationPage
x:Class="AuthenticationSample.WP80.
                                Views.LoginView"
      xmlns="http://schemas.microsoft.com/
                    winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/
                                winfx/2006/xaml"
      xmlns:Command="clr-namespace:GalaSoft.
   MvvmLight.Command;assembly=GalaSoft.MvvmLight.
                                    Extras.WP8"
      xmlns:controls="clr-namespace:Facebook.
        Client.Controls;assembly=Facebook.Client"
      xmlns:d="http://schemas.microsoft.com/
                            expression/blend/2008"
      xmlns:i="clr-namespace:System.Windows.
                  Interactivity;assembly=System.
                            Windows.Interactivity"
      xmlns:mc="http://
                     schemas.openxmlformats.org/
                      markup-compatibility/2006"
      xmlns:phone="clr-namespace:Microsoft.
         Phone.Controls;assembly=Microsoft.Phone"
      xmlns:shell="clr-namespace:Microsoft.
            Phone.Shell;assembly=Microsoft.Phone"
      xmlns:converters="clr-namespace:Cimbalino.
                        Phone.Toolkit.Converters;
                assembly=Cimbalino.Phone.Toolkit"
      Orientation="Portrait"
      SupportedOrientations="Portrait"
      shell:SystemTray.IsVisible="True"
      mc:Ignorable="d">
  <phone:PhoneApplicationPage.DataContext>
    <Binding Mode="OneWay"
          Path="LoginViewModel"
          Source="{StaticResource Locator}" />
  </phone:PhoneApplicationPage.DataContext>
  <phone:PhoneApplicationPage.Resources>
    <converters:BooleanToVisibilityConverter
           x:Key="BooleanToVisibilityConverter"/>
  </phone:PhoneApplicationPage.Resources>
  <phone:PhoneApplicationPage.FontFamily>
    <StaticResource ResourceKey=
                       "PhoneFontFamilyNormal" />
  </phone:PhoneApplicationPage.FontFamily>
  <phone:PhoneApplicationPage.FontSize>
    <StaticResource ResourceKey=
                        "PhoneFontSizeNormal" />
  </phone:PhoneApplicationPage.FontSize>
  <phone:PhoneApplicationPage.Foreground>
    <StaticResource ResourceKey=
                        "PhoneForegroundBrush" />
  </phone:PhoneApplicationPage.Foreground>
  <!--  LayoutRoot is the root grid where all
                       page content is placed  -->
  <Grid x:Name="LayoutRoot"
                        Background="Transparent">
    <Grid.RowDefinitions>
      <RowDefinition Height="Auto" />
      <RowDefinition Height="*" />
    </Grid.RowDefinitions>

    <!--  TitlePanel contains the name of the ap-
plication and page title  -->
    <StackPanel x:Name="TitlePanel"
          Grid.Row="0"
          Margin="12,17,0,28">
      <TextBlock Margin="12,0"
            Style="{StaticResource
                              PhoneTextNormalStyle}"
            Text="{Binding
               LocalizedResources.ApplicationTitle,
                    Mode=OneWay,
                    Source={StaticResource
                              LocalizedStrings}}" />
      <TextBlock Margin="9,-7,0,0"
            Style="{StaticResource
                              PhoneTextTitle1Style}"
            Text="{Binding
                LocalizedResources.LoginView_Title,
                    Mode=OneWay,
                    Source={StaticResource
                              LocalizedStrings}}" />
    </StackPanel>

    <!--  ContentPanel - place additional content
                                          here  -->
    <Grid x:Name="ContentPanel"
          Grid.Row="1"
          Margin="24,0,0,-40">
      <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
      </Grid.RowDefinitions>
      <TextBlock Grid.Row="0"
            Style="{StaticResource
                              PhoneTextTitle2Style}"
            Text="{Binding
           LocalizedResources.LoginView_UserAccount,
                    Mode=OneWay,
                    Source={StaticResource
                              LocalizedStrings}}" />
      <Button Grid.Row="1"
            Margin="10"
            Command="{Binding LoginCommand}"
            CommandParameter="facebook"
            Content="Facebook" />
      <Button Grid.Row="2"
            Margin="10"
            Command="{Binding LoginCommand}"
            CommandParameter="microsoft"
            Content="Microsoft" />
      <Button Grid.Row="3"
            Margin="10"
            Command="{Binding LoginCommand}"
            CommandParameter="google"
            Content="Google" />
    </Grid>
    <Grid Visibility="{Binding InProgress,
                          Converter={StaticResource
                    BooleanToVisibilityConverter}}"
          Grid.Row="0"
          Grid.RowSpan="2">
      <Rectangle
            Fill="Black"
            Opacity="0.75" />
      <TextBlock
            HorizontalAlignment="Center"
            VerticalAlignment="Center"
            Text="{Binding
           LocalizedResources.LoginView_AuthMessage,
                    Mode=OneWay,
                    Source={StaticResource
                              LocalizedStrings}}" />
      <ProgressBar IsIndeterminate="True"
               IsEnabled="True" Margin="0,60,0,0"/>
```

```
    </Grid>

  </Grid>
</phone:PhoneApplicationPage>
```

E o resultado final será:



O código fonte do exemplo apresentado pode ser obtido aqui.

Em conclusão, podemos verificar que é extremamente simples criar uma aplicação que use autenticação usando a conta Facebook, Google ou Microsoft, facilitando assim o desenvolvimento da aplicação, pelo facto de não ter necessidade de criar uma autenticação personalizada e por outro lado o utilizador poderá usar a conta que à partida já tem, evitando assim criar mais uma conta nova.

AUTOR

Escrito Por Sara Silva

Licenciada em Matemática – Especialidade em Computação, pela Universidade de Coimbra e é Microsoft Certified Profissional 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.

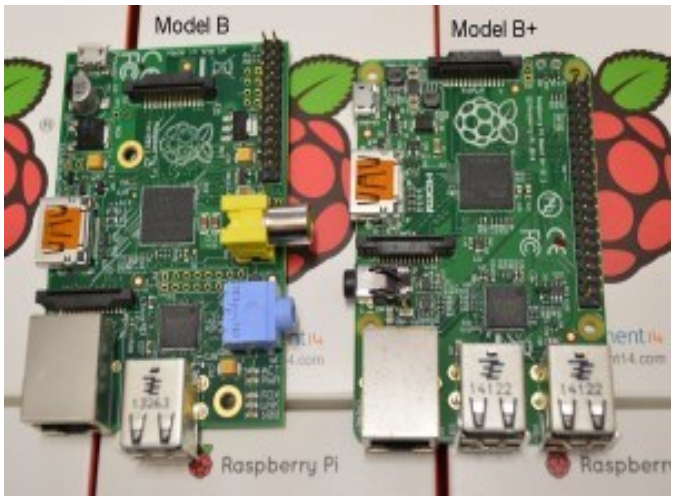
No Code

Raspberry Pi em versão melhorada

jQuery: usar ou não usar?

Raspberry Pi em versão melhorada

Há algumas edições atrás falámos do *Raspberry Pi*, o micro computador que ganhava cada vez mais adeptos. Pois bem, desta vez o *Raspberry Pi* voltou com uma versão melhorada de si mesmo. Falamos, caro leitor, do modelo *B+*.



Para aqueles que não se recordam, as primeiras versões deste "brinquedo" (versão *A* e versão *B*) começaram a ser comercializadas em 2012, nascidas de uma colaboração entre a *Universidade de Cambridge* e a *Fundação Raspberry Pi*. Em 2014 vemos então esta parceria lançar uma versão melhorada do modelo *B*. Usando estas duas versões para comparativo, elas "pouco" diferem uma da outra; uma das diferenças que salta à vista é o facto do *B+* ter agora quatro entradas *USB*, sendo que o seu homólogo tinha apenas duas.

Falando por experiência própria, quando o ano passado adquiri o modelo *B*, foi realmente algo desolador na primeira utilização, verificar que depois de ligar o rato e o teclado por *USB*, deixava de ter entrada para usar, por exemplo, uma simples *pen*. Claro que isto rapidamente foi ultrapassado com um pequeno investimento num *hub USB* que me permitiu usar, o rato, o teclado, o adaptador do *wifi* e uma *pen* ao mesmo tempo. O *Raspberry Pi B+* vem dar resposta a esta necessidade do utilizador comum.

Outra das diferenças que salta à vista do utilizador mais comum é o facto da entrada *SD* ter dado lugar à *mini SD*, o que permite que nesta versão não fiquemos um com terço do cartão *SD* de "fora" como acontecia na versão anterior. Outro dos melhoramentos é o facto da entrada de 26 *pins GPIO* ter dado lugar a uma entrada com 40 *pins*. Tendo em conta que os primeiros 26 continuam iguais, todos os projectos feitos utilizando a versão *B* são compatíveis com esta versão melhorada. Esta novidade permite uma maior versatilidade nos projectos informáticos e electrónicos a que frequentemente vemos o *Pi* associado.

Function/ GPIO	J8 Pin		Function/ GPIO
3.3V	1	2	5.0V
GPIO2	3	4	5.0V
GPIO3	5	6	0V
GPIO4	7	8	GPIO14
0V	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	0V
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	0V
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
0V	25	26	GPIO7
(GPIO0) ID_SD	27	28	ID_SC (GPIO1)
GPIO5	29	30	0V
GPIO6	31	32	GPIO12
GPIO13	33	34	0V
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
0V	39	40	GPIO21

Nesta versão, as melhorias do áudio também não passaram despercebidas, sendo que o circuito áudio dispõe de uma fonte de alimentação de baixo ruído, facilitando ainda mais o uso desta funcionalidade (a versão anterior por defeito trazia o áudio desligado, sendo que o utilizador é que reabilitava essa função). E por falar em fontes de alimentação, o novo *B+* tem um menor consumo de energia. Esta é inclusive uma das novas características mais destacadas pelos seus criadores. Segundo a *Fundação*, o novo *Pi* tem um consumo de energia reduzido na ordem dos 0,5W e 1W. Esta melhoria é sentida principalmente nos projectos que utilizam baterias, pois ganham alguma vantagem na sua autonomia.

Melhorias à parte, continuamos a ter o mesmo processador, o *ARM1176JZFS*, que é já nosso conhecido da família *ARM11*. Recordando o leitor, este processador utiliza um *pipeline* de 8 estágios, o que permite ao processador

procurar uma ou mais instruções além da próxima a ser executada, sendo que essas instruções são colocadas numa fila de espera dentro do processador onde aguardam a altura da sua execução.

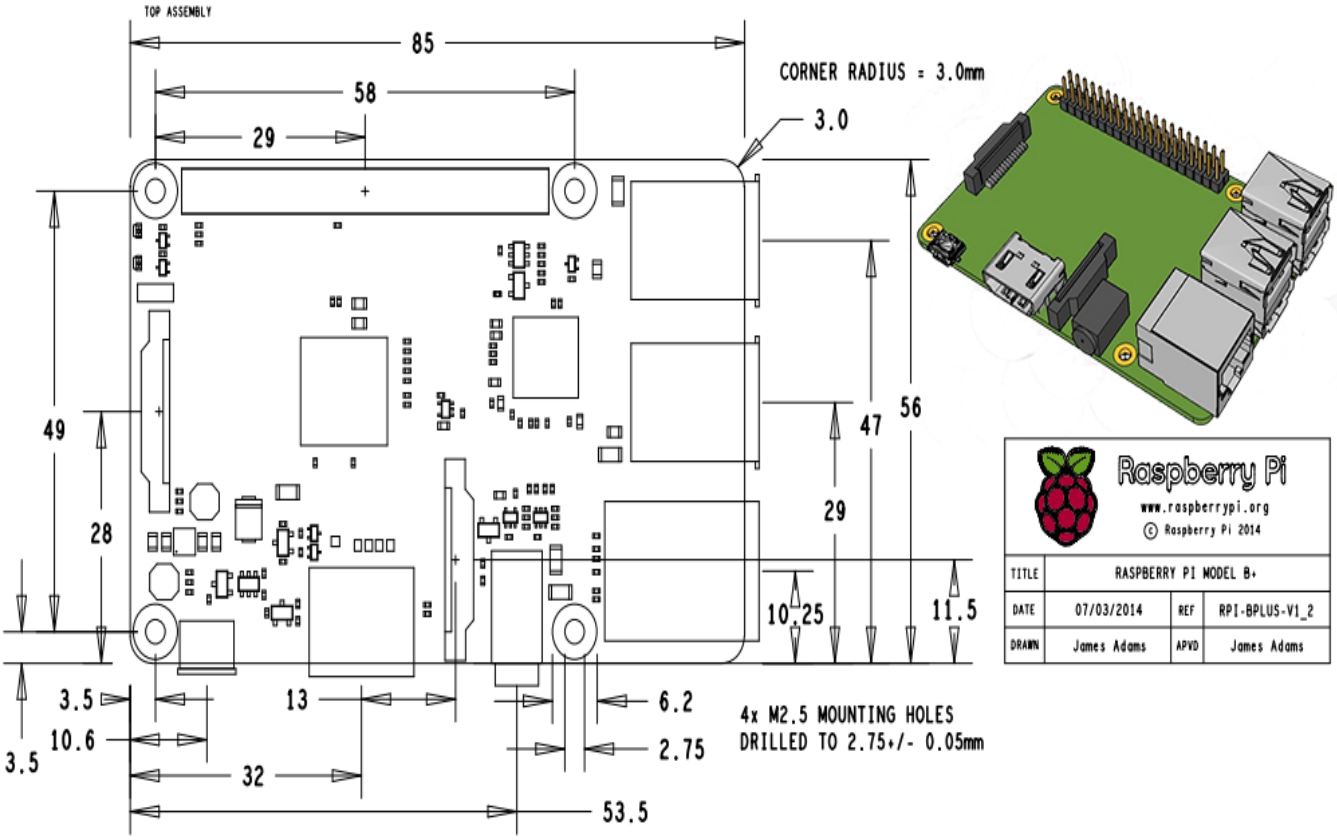
Recapitulando… a nova versão do *Pi* foi melhorada, é um facto! Mas apesar de este continuar a ser um projecto interessante, serei a única a ter esperado um pouco mais desta nova versão? O modelo *B+* continua a ter um *CPU* de 700MHz e 512MB de memória. Admito que quando ouvi falar que seria lançada uma versão melhorada pensei automaticamente que melhorariam também um pouco a memória e o processador deste projecto.

Isto porque a *Humming Board*, criada em 2011 pela *Solid Run*, um dos concorrentes directos do *Raspberry Pi*, teve numa das suas versões iniciais logo um processador de 1GHz (versão base com 512MB de memória e versão intermédia com 1GB). Esta concorrente é de facto um pouco mais cara que o *Pi* mas tem conseguido ganhar adeptos e conquistar um lugar de topo nos rivais do mini computador mais famoso destes últimos anos. A *Humming Board* pode suportar um processador quad-core e até 2GB de *RAM*.

A nível de sistema operativo é indiferente qual o modelo *Raspberry* que utiliza, pois os três modelos (versão *A*, *B* e *B+*) são compatíveis com qualquer um dos disponíveis. Os mais utilizados continuam a ser as versões dos sistemas operativos *Raspbian* e o *Noobs*.

Características à parte, a verdade é que o *Raspberry Pi* continua fiel a si próprio, sempre no intuito de levar a todos, de forma relativamente simples e económica, o contacto pelas tecnologias incentivando à aprendizagem da programação logo desde os primeiros anos de vida. Objectivo esse que tem sido alcançado.

Resta dizer que a *Fundação Raspberry* adiantou que a versão melhorada do *Raspberry Pi* modelo *B*, o *B+* estará no mercado pelo mesmo preço da versão anterior.



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.

jQuery: usar ou não usar?

Esta é uma questão frequente quando se começa um projeto web. Origina por vezes discussões acesas. A resposta rápida é que sim; provavelmente o que se está a desenvolver justifica o seu uso.

Quando se pondera a utilização de uma qualquer framework ou biblioteca (em qualquer projeto de software) deve-se questionar se os ganhos compensam o peso extra; se a complexidade do problema a resolver o justifica. De forma grosseira e geral, a questão é:

- partido que vamos tirar da biblioteca/framework > 1 ?
- peso trazido pela mesma

Assim, há que considerar:

1. Dificuldade de instalação e de aprendizagem;
2. Peso antes de runtime: o peso extra no projeto pode atrasar tempos iniciais (ex: deploys, downloads);
3. Peso em runtime: gastos extra de memória (RAM) e de processador: colocar uma camada à frente, por mais "fina" que seja, no cômputo geral, pode ser significativo.
4. Liberdade do uso: a instalação e/ou adição de uma biblioteca não deve forçar o seu uso.
5. Abstração indevida: ao se fazer tudo em alto nível, perde-se a noção do que se passa "por baixo", o que é imprudente.

Em software, a complexidade não se elimina, apenas se transfere. Ao usar jQuery, está a reduzir a complexidade de programação, mas por outro lado, a aumentar a complexidade para o browser, que tem de realizar mais operações, gastando mais tempo e memória. Quanto às considerações acima mencionadas:

1. **Dificuldade de instalação e de aprendizagem**: para começar a usar jQuery basta incluir um ficheiro JavaScript; as suas instruções de código são coerentes e simples e portanto de aprendizagem muito natural;
2. **Peso antes de runtime**: o download é reduzido: o jQuery é apenas um ficheiro com cerca de 90KB (versão minified). Além disso, o uso de CDN (servidores que permitem ao browser reutilizar ficheiros em cache) tornam isso quase irrelevante.
3. **Peso em runtime**: na grande maioria dos casos, o preço a pagar são apenas alguns milisegundos extra (poupam-se horas de desenvolvimento); é certo que se devem aplicar boas práticas (ex. caching de variáveis, boa utilização de seletores, ...) mas são quase as mesmas que em JavaScript puro. Onde o uso direto de JavaScript for determinante em termos de velocidade, pode-se sempre fazê-lo…
4. **Liberdade do uso**: após ser incluida, a biblioteca jQuery só é usada se se quiser; se nunca se usar, o único peso extra é o do seu download (se se usar um CDN, nem isso).
5. **Abstração indevida**: assumindo que ninguém deve aprender jQuery sem estar fluente em JavaScript e que o código jQuery é aberto e disponível (para não falar das ferramentas de desenvolvimento dos browsers), não há desculpa para se fazer as coisas sem se ter noção dos seus impactos… De qualquer forma, mais tarde ou mais cedo, o programador irá lidar com os problemas de baixo nível.

Algumas das vantagens que o jQuery traz, como o suporte cross-browser (onde se incluem browsers mobile), podem ser colmatadas com microframeworks e/ou pollyfills (ou scripts do próprio programador)... Mas as principais vantagens: coerência, simplicidade, expansibilidade via plugins (ex: jQuery UI), suporte da comunidade, uma API fluida, constantes atualizações, entre outras... ficam pelo caminho.

Para os fãs, poupa muitas horas de trabalho (ex: mudar a cor dos elementos com uma classe pode passar de 5 linhas para 1). Tanto para um amador que quer fazer um site para o tio, como para um profissional desenvolver um backoffice, o jQuery pode ser de grande ajuda. Para os puristas, o peso, muitas vezes, não compensa o seu uso. Contudo, os ganhos para o programador são elevados sendo difícil negar o seu uso. Podemos até argumentar que ajudou a democratizar o desenvolvimento web, permitindo a todos "fazer sites".

Se fôssemos a pensar nos "6 ou 7 milisegundos" ganhos pelo uso direto de uma linguagem, nunca se tinham inventado linguagens de alto nível como o Java ou o C# e ainda estaríamos a programar em Assembly. Num mundo de sites e aplicações web tendencialmente mais complexas, o programador tem de se munir de ferramentas que lhe permitam evitar reinventar a roda, focando-se mais nos problemas do negócio e menos nos dos computadores.

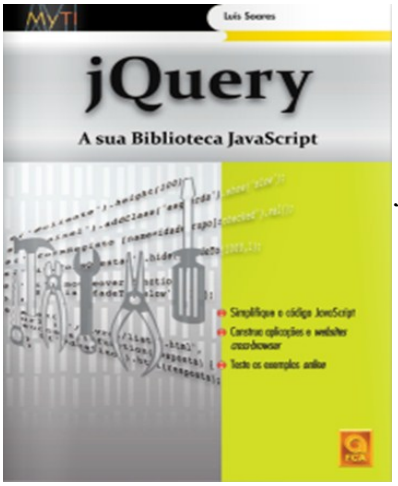
Em conclusão, afirmar que nunca se deve usar jQuery ou outras bibliotecas é claramente um extremo, e portanto, deve

gerar algum ceticismo... afirmar que se deve usar sempre também é exagerado. Cada caso é um caso... e raramente há soluções universais.

> ***Quando se pondera a utilização de uma qualquer framework ou biblioteca (...)) deve-se questionar se os ganhos compensam o peso extra; se a complexidade do problema a resolver o justifica.***

Mas há que ser pragmático: na maioria dos projetos web as vantagens do jQuery suplantam de longe as desvantagens: até para as coisas mais simples pode ser uma lufada de ar fresco, poupando muitas linhas de código e algumas dores de cabeça.

> ***Para os puristas, o peso, muitas vezes, não compensa o seu uso. Contudo, os ganhos para o programador são elevados sendo difícil negar o seu uso.***



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.

USAR OU NÃO USAR MÚLTIPLOS MONITORES PARA PROGRAMAR

Vivemos numa era multitarefa, onde muito frequentemente se usam várias aplicações em simultâneo no nosso ambiente de trabalho computacional. Apesar de atualmente existirem monitores de elevada resolução e tamanho, nem sempre estes conseguem suprir todas as necessidades de um profissional que necessita de otimizar e agilizar todas as ferramentas de desenvolvimento de modo a rentabilizar o seu tempo.

Quem programa certamente já sentiu na pele a falta de espaço no ambiente de trabalho para ter alguma documentação e IDE visíveis em simultâneo e é aí mesmo que um sistema com 2 ou mais monitores mostra as suas reais vantagens.

Apesar de eu não ser um programador profissional (ainda) possuo um sistema de duplo monitor o que me leva a sentir realmente todas as vantagens que um sistema desta natureza tem. Geralmente uso o IDE em full-screen num dos monitores, reservando o segundo monitor para o browser, documentação e um sem fim de outras coisas.

Desta forma consigo ter sempre presente o IDE e todo o código que estou a escrever no momento, sem que necessite de prescindir de uma página web aberta, do iTunes com a minha playlist musical, ou seja, consigo maximizar a informação disponível visivelmente em tempo real, o que acaba por agilizar o trabalho.

Para quem usa um computador portátil com um ecrã relativamente pequeno (13.3", 14" ou mesmo 15.6") para programar é muito mais produtivo acoplar um monitor externo expandindo a área de trabalho e assim agilizando a sua utilização.

Pessoalmente acho mais cómodo e agradável a utilização de um sistema de múltiplos monitores, mas no entanto os críticos da segurança defendem que não é produtivo e que exige um maior esforço físico ao utilizador, resultando num maior cansaço visual.

Gostos não se discutem, e é por esse mesmo motivo que o assunto das vantagens e desvantagens em usar um sistema de duplo monitor para programar é complicado de se discutir, pois cada um tem os seus hábitos aos quais mais se adapta.

“Para quem usa um computador portátil com um ecrã relativamente pequeno (13.3", 14" ou mesmo 15.6") para programar é muito mais produtivo acoplar um monitor externo”

A minha opinião vale o que vale, mas posso afirmar que existem realmente vantagens em usar um sistema de duplo monitor para programar, eu já experimentei e fiquei rendido. Certamente que não fui o único!

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/

PROJECTO EM DESTAQUE NA COMUNIDADE P@P: TAP BALLZ

Tap Ballz é um jogo feito por dois jovens portugueses Joel Belo e João Araújo (WIP Games). Este jogo é bastante simples. É um jogo onde temos de carregar nas bolas que vão descendo no ecrã na cor pedida. Cada vez o jogo fica mais difícil, aumentando a velocidade e tornando-se um verdadeiro desafio! O objectivo é obter a melhor pontuação possível.

Este jogo destaca-se por ser um jogo de raciocínio rápido no qual uma tela cheia de bolinhas coloridas é apresentada no display e o jogador deve apenas pressionar as que da cor que for indicada. Assim, será necessário ter reflexos rápidos para acertar apenas a cor correta e evitar perder pontos.

Quando se toca nas bolinhas da cor certa, aumenta um pouco a barra do contador, por isso, quanto mais vezes acertar, maior será a partida.

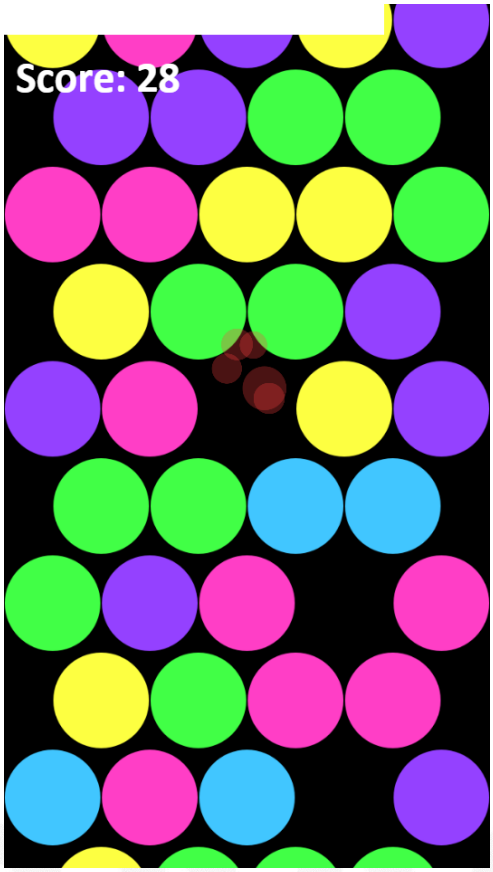
O jogo esteve no top 20 das novidades na categoria puzzle, na GameReactor, durante a terceira semana de Dezembro, e assume um formato free-to-play. Este é o segundo jogo da WIPGameStudios, seguindo-se a Super Balls, para PC .

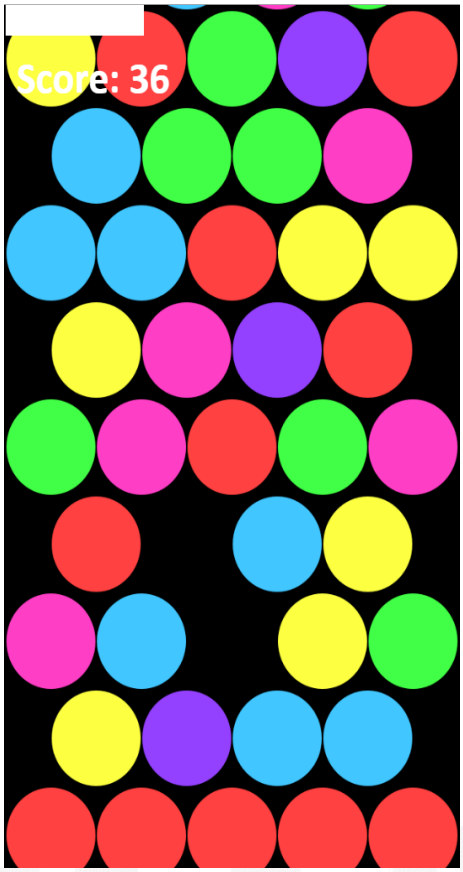
Apesar de ser relativamente simples o jogo tem um carácter bastante interessante e acaba tornando-se viciante para os jogadores.

Exige bastante atenção de quem joga e consegue ser bastante exigente em termos de concentração, no entanto consegue manter o carácter simplista e engraçado que o torna mais apelativo até para jogadores menos habituais.

Não deixa de ser um pequeno projecto de jogo, com fortes potencialidades e que nos mereceu o destaque nesta edição.

(Sara Santos)





Elege o melhor artigo desta edição

Revista PROGRAMAR

http://bit.do/ProgramarED46_V

Veja também as edições anteriores da Revista PROGRAMAR

45ª Edição - Maio 2014

44ª Edição - Fevereiro 2014

43ª Edição - Dezembro 2013

42ª Edição - Setembro 2013

41ª Edição - Junho 2013

40ª Edição - Abril 2013

e muito mais em ...

www.revista-programar.info

DUVIDAS?

IDEIAS?

AJUDAS?

PROJECTOS?

portugal-a-programar.org