PROGRAMAR

A REVISTA DE PROGRAMAÇÃO PORTUGUESA

Revista nº14 - Maio de 2008

www.portugal-a-programar.org

Network Scanning

Interacção Python/MySQL

REFLECTION E SUAS APLICAÇÕES

e ainda...

Techdays 2008

Engenharia de Software

Festival Nacional da Robótica

índice

equipa PROGRAMAR

Um poder demasiado grande para ser partilhado

Tudo nasce na ideia. É dela que o projecto surge, é sobre ela que o produto é concebido e no final testado. A ideia move a inovação e a tecnologia, mas existe uma entidade sem a qual todo e qualquer pensamento inovador morre por si, pois é dela dependente: o utilizador. Qualquer pedaço de software, site web ou qualquer outro produto é feito para ser usado por alguém, tem em vista ultimamente satisfazer a falha naquela necessidade específica do utilizador e é esse desejo que move quem concebeu o produto, ou deveria. Como reagir se a maneira como concebemos o nosso produto acaba por ser tida como não a melhor, como a que não resolve da melhor maneira as necessidades que deveríamos cobrir?

Fui informado há uns dias que o famoso cliente de instant messaging Pidgin (antigo Gaim) havia levado um duro golpe moral. Culpados de não cobrir as necessidades dos utilizadores ou as suas reivindicações, developers externos ao projecto Pidgin decidiram criar um fork deste, denominado FunPidgin, numa tentativa de apresentar um produto que realmente cobriria as necessidades especificas necessárias ou a criar num sistema de mensagens instantâneas. Tentado responder à pergunta que finalizou o primeiro parágrafo, e pelo que me foi dado a entender por uma mensagem escrita por um developer do Pidgin no blog oficial, a equipa desculpou-se tentando provar que de facto ouvia os utilizadores, no entanto, pouco mais que isto poderiam fazer, pois o Pidgin está sob licença GNU GPL que permite que isto mesmo seja feito.

Não me cabe, nem interessa, analisar se o que aconteceu foi merecido ou injusto, mas apenas constatar um facto: quem ganhou com tudo isto? Quem tem neste momento a possibilidade de escolher entre dois produtos semelhantes em que um deles se tentou aproximar ainda mais das necessidades requeridas? A resposta está no utilizador, a base do projecto que por vezes é esquecida e posta de parte, quando tudo começou nesta.

Quão maravilhoso é poder ter um sistema em que quem ganha, acima de tudo o resto, é o utilizador, pela competição cobrindo necessidades, pela cooperação melhorando-se cada vez mais, sem impedimentos ou restrições? Este é de facto um poder demasiadamente grande para ser partilhado embora esteja por aí para todos os que o queiram adoptar, visto de soslaio pelas alternativas que na mesma rua deste circulam. É o poder do software livre.

Governo alarga e-escolas

O Programa e-escolas vai ser alargado a federações, associações juvenis e de estudantes, abrangendo um universo de mais de meio milhão de jovens. Os primeiros computadores são entregues já hoje numa cerimónia nas instalações do Instituto Português da Juventude, no Parque das Nações.

Este novo alargamento do programa às associações juvenis e estudantis é feito no Dia do Associativismo Jovem e abrange as instituições que integram o Registo Nacional do Associativismo Jovem - RNAJ. Segundo dados do Ministério das Obras Públicas Transportes e Comunicações, que tutela esta iniciativa, existem mil associações juvenis e duzentas associações de estudantes, distribuídas por todo o país, representando um universo de mais de meio milhão de jovens.

Não são ainda conhecidos muitos detalhes desta nova iniciativa mas pela informação fornecida pelo ministério depreende-se que os jovens destas associações terão acesso aos computadores portáteis por um custo máximo de 150 euros e a ligações de banda larga móvel com uma redução de 5 euros em relação ao valor praticado pelos operadores, tal como acontece para os restantes escalões de utilizadores. O objectivo é para já entregar um portátil a cada associação, podendo este modelo ser alterado conforme a adesão à iniciativa.

Recorde-se que os operadores móveis se dispuseram a investir neste programa os fundos reservados para o desenvolvimento da Sociedade da Informação, sendo a TMN a empresa que colocou mais alto a fasquia, definindo que queria distribuir pelo menos 50 por cento dos 600 mil portáteis estimados no arranque da iniciativa, prevendo um investimento de 150 milhões de Euros. Já a Optimus apontava para os 116 milhões de euros, uma verba que previa ser suficiente para a entrega de 250 mil notebooks.

Até agora já foram entregues cerca de 150 mil portáteis no programa e-escolas e o programa já recebeu perto de 250 mil inscrições.

Centros indianos pagam a funcionários para quebrar CAPTCHAs

A Trend Micro detectou que os cibercriminosos encontraram uma forma de contornar os sistemas CAPTCHA. No caso dos acessos serem levados a cabo por robots, a probabilidade dos controlos de segurança serem quebrados é de 30 a 35 por cento. Contudo, a empresa descobriu agora que existem centros na Índia que pagam entre 2,5 a 3,75 euros por dia aos seus colaboradores para que estes enganem as validações de segurança CAPTCHA, o que acaba por fazer com que a taxa de sucesso nas tentativas de entrada em páginas protegidas por estes sistemas seja de 100 por cento. Desta forma, os utilizadores maliciosos conseguem aumentar o número de contas que têm disponíveis para distribuírem as suas ameaças.

De acordo com a Trend Micro, o processo malicioso detectado consiste no acesso do programa robot a uma página de inscrição de um serviço. É preenchido o formulário com dados aleatórios e, quando aparece a verificação CAPTCHA, o programa envia a mensagem a um terminal sedeado na Índia. A partir daí, os utilizadores pagos decifram os caracteres e reenviam a informação para o programa que insere o código e abre uma conta a partir da qual pode enviar spam para muitos utilizadores.

Fedora 9 lançado

No passado dia 13 de Maio, a 9ª versão da distribuição Fedora patricionada pela Red Hat foi lançada ao público.

Entre as novidades encontramos a versão 2.22 do GNOME e o novo KDE4, melhoramentos no NetworkManager, a Beta 5 no novíssimo Firefox 3, o OpenJDK6 (a versão do SDK do Java da Sun lançada sobre uma licença grátis e opensource) e suporte ao novo filesystem ext4.

O download está disponível através do site do projecto: http://fedoraproject.org/get-fedora

Reflexão

Antes de começar por explicar de que se trata a reflexão, talvez seja melhor perceber o que é a Meta Programação. A Meta Programação consiste na escrita de programas que manipulam ou escrevem outros programas. Pode parecer um conceito um pouco esquisito à primeira vista, mas softwares como sejam debuggers ou softwares de profile são exemplos clássicos de meta programas largamente utilizados. Os meta programas são escritos em meta linguagens, enquanto que os programas manipulados são escritos em linguagem objecto.

A reflexão não é mais nem menos que, a capacidade de uma linguagem ter como meta linguagem ela própria. Ou seja, com a reflexão uma entidade tem a capacidade de manipular a representação da sua estrutura e comportamento durante a sua própria execução.

A reflexão pode existir em dois graus diferentes mas que podem coexistir, a Introspecção e a Intercessão.

A Introspecção é a capacidade que um sistema tem de observar a sua própria estrutura e comportamento. Como exemplo de introspecção podemos ter a determinação do tipo de um objecto em run time.

Por outro lado, a Intercessão é a capacidade do sistema de modificar a sua estrutura e comportamento. Um exemplo é querermos mudar a classe de uma determinada instância.

Ao usarmos a reflexão, seja introspecção ou intercessão, queremos poder interagir com os componentes do sistema como se fossem objectos normais da nossa linguagem. Queremos poder dizer "quais são os métodos da classe X?" ou "de que classe é esta instância?". Isto é conseguido porque existe o conceito de Reificação.

A reificação trata-se da transposição de um conceito abstracto para um objecto acessível para o programador. Se pretendemos perguntar a uma classe quais os seus métodos, então temos de ter disponível um objecto que represente uma classe, e assim por diante.

O Java começou sem nenhumas capacidades de reflexão, na evolução para a versão 1.1 foi introduzido o package `java.lang.reflect`[1], que passou a possibilitar um mecanismo de reflexão para a linguagem ainda sem grande capacidade. Essa capacidade tem vindo a ser aumentada com o lançamento de novas versões da linguagem. Actualmente o Java permite introspecção estrutural, mas é muitíssimo limitado em termos de intercessão. Esta limitação existe sobretudo porque quando uma classe Java é carregada para a máquina virtual já não pode ser alterada, todas as alterações têm de ser feitas antes do carregamento. Existem ferramentas com esse objectivo, a mais conhecida é talvez o `Javassist`[2]. O `Javassist` consegue meter-se antes do carregamento da classe e fazer alterações ao seu bytecode.

Vamos então ver um exemplo do uso da reflexão do Java. Imaginemos que se quer mostrar os métodos não privados de uma determinada classe:

```
import java.lang.reflect.*;

public void mostrarMetodos(Class c) {
    System.out.println(c + " {");

    //Vamos pedir os métodos que não
são privados da classe
    for (Method m :
c.getDeclaredMethods()) {
        if (!
Modifier.isPrivate(m.getModifiers()))
{
            System.out.println(" " +
m);
        }
    }
    System.out.println("}");
}
```

Neste exemplo estamos efectivamente a usar meta objectos que representam os objectos da linguagem Java. Um desses meta objectos é a `Class`. Para cada tipo, seja primitivo ou referência, há uma instância desta classe para representar esse tipo. Este meta objecto (`java.lang.Class`) possui diversos métodos para a reflexão, uns exemplos são[3]:

`String getName()` - Retorna o nome da entidade.

`boolean isInterface()` - Verifica se o tipo representado é uma interface.

`boolean isPrimitive()` - Verifica se o tipo representado é primitivo.

`int getModifiers()` - Retorna os modificadores para o tipo (codificado em inteiro) representado.

`Class getSuperclass()` - Retorna a classe que representa a superclasse, da classe representada

`Field[] getFields()` - Retorna um array contendo os Fields que representam os fields de acesso publico da classe ou interface.

`Method[] getMethods()` - Retorna um array contendo os Métodos que representam os métodos de acesso público da classe ou interface.

São muito diversas as aplicações possíveis para o uso da reflexão, um exemplo interessante era fazermos um inspector de objectos. Um inspector de objectos é basicamente uma ferramenta de depuração, que apresenta uma descrição visual do estado dos objectos do nosso programa. A ideia era o nosso programa chamar o inspector com um determinado objecto a inspeccionar:

```
MyObject m = new MyObject();

Inspector inspector = new Inspector();
inpector.inspect(m);
```

Então ia ser listada informação sobre o objecto m inspeccionado. Informação como os fields existentes, métodos aplicáveis, etc. Vamos começar então por listar os fields de um objecto:

```
for(Field field :
m.getClass().getDeclaredFields()){
    //Aqui vamos buscar os
modificadores do field (public,
private...)

System.out.print(modToString(field.getM
odifiers())+" ");

    //Aqui vamos buscar o tipo e o
nome do field

System.out.print(field.getType().getCan
onicalName()+" ");
    System.out.print(field.getName()+"
= ");
    //Finalmente vamos buscar o valor
do field
    try{
        field.setAccessible(true);

System.out.println(field.get(object).to
String());
    } catch(IllegalAccessException
iae){
        System.out.println(iae+": Não
tem permissão.");
        throw new RuntimeException();
    } catch(NullPointerException npe){
        throw new RuntimeException();
    }
}
```

O que acontece é que por vezes um objecto herda fields da sua super classe, então também queremos que esses sejam visíveis quando inspeccionamos o objecto:

```
for(Field field :
object.getClass().getSuperclass().getFi
elds()){

System.out.print(modToString(field.getM
odifiers())+" ");

System.out.print(field.getType().getCan
onicalName()+" ");
    System.out.print(field.getName()+"
= ");

    try{
        field.setAccessible(true);

System.out.println(field.get(object).to
String());
    } catch(IllegalAccessException
iae){
        System.out.println(iae+": Não
tem permissão.");
        throw new RuntimeException();
    } catch(NullPointerException npe){
        throw new RuntimeException();
    }
}
```

Este exemplo é bastante parecido com o anterior, a única coisa que fizemos foi aceder á super classe do objecto através do método `getSuperClass()`. Há uma linha em comum nos dois exemplos anteriores que pode merecer alguma curiosidade, é a linha onde aparece `field.setAccessible(true)`. Isto não passa de uma pseudo protecção que o sistema de reflexão do Java tem de maneira a que o programador não posso explicitamente alterar um valor de um field, sem primeiro "dar autorização para tal". Ou seja é o programador que dá autorização a ele próprio para fazer a alteração.

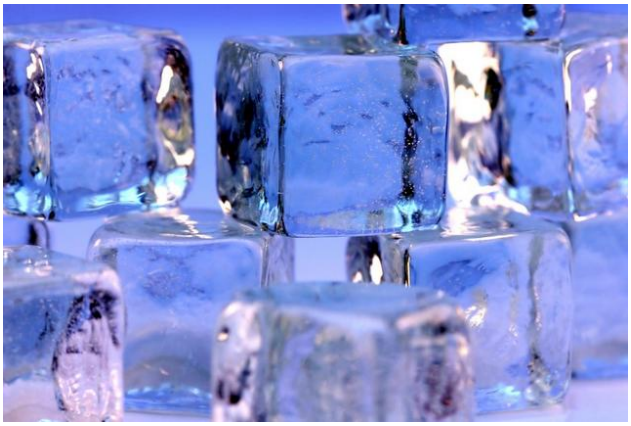
Agora o desejável seria alterar um determinado valor de um `field`, também é possível com a reflexão:

```
//Carregamos a classe com nome "Foo"
Class cls = Class.forName("Foo");

//Vamos buscar o field que tem nome
"bar" e criamos uma instância de Foo
Field f = cls.getField("bar");
Foo foo = new Foo();

System.out.println("bar = " + foo.d);

//Aqui damos um novo valor ao field
f.setInt(foo, 1024);

System.out.println("bar = " + foo.d);
```

Estes exemplos não são nada de mais, comparado com o que se pode fazer com a reflexão do Java. A reflexão do Java tem uma API considerável, que pode ser explorada poupando trabalho ao programador. O poder que se tem sobre esta linguagem não é o mesmo de outras como o Common Lisp, por exemplo, onde existe uma igualdade entre dados e programas, o que permite inspeccionar programas como se fossem dados. Mas o maior problema do sistema de reflexão do Java é definitivamente a parte da intercessão, que é muito limitada.

Conclusões

Com este artigo pretendeu-se que o leitor tenha ficado com a ideia geral do que é a meta programação e a reflexão, e a



ligação que existe entre ambas. A reflexão não está, de maneira nenhuma, directamente ligada ao Java. Muitas outras linguagens têm um sistema de reflexão, tão bom ou melhor que o Java. Exemplos disso são o Common Lisp, Smalltalk, Python, etc. [4]

Referências

[1]
https://cis.med.ucalgary.ca/http/java.sun.com/docs/books/jls/first_edition/html/1.1Update.html

[2]
http://www.csg.is.titech.ac.jp/~ chiba/javassist/

[3]
http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Class.html

[4]
http://en.wikipedia.org/wiki/Reflection_%28computer_science%29

SOBRE O AUTOR

Estudante do Instituto Superior Técnico, está a tirar o Mestrado em Engenharia Informática.

As principais áreas de interesse são os Sistemas Distribuidos e a Engenharia de Software, e como linguagens preferidas tem o C, o C# e o Java.

Luís Antunes

Interacção Python/MySQL

Introdução

Bases de dados são armazéns estruturados de informação, nas quais é guardada uma variedade imensa de dados. Na verdade, são muitas vezes o núcleo duro de aplicações informáticas, sem que, também na maioria das vezes, o utilizador final se aperceba. A importância desta tecnologia hoje em dia é imensa, sendo que foi necessário o desenvolvimento de ferramentas para que programadores possam dela tirar partido. Entre essas ferramentas, incluem-se diversos sistemas de gestão de bases de dados, como o MySQL, o SQLServer, a Oracle, entre outros, que permitem a manutenção das bases de dados de uma maneira relativamente simples. Porém, também para estas ferramentas de gestão foi desenvolvido software para facilitar ainda mais a vida a quem desenvolve. Neste artigo pretende-se explicar e exemplificar o uso de uma API que permite a manutenção de bases de dados usando MySQL a partir de scripts de Python. Porém, não é limitado apenas a MySQL. Outros adaptadores para outros sistemas de gestão de bases de dados funcionam de maneira muito semelhante, por vezes até bastando apenas mudar uma linha de código.

Instalação

Antes de mais nada há que fazer o download do software necessário. Este artigo e o código foram criados e testados numa distribuição de GNU/Linux, neste caso Ubuntu, mas deve funcionar da mesma forma noutros sistemas operativos. Primeiro, é necessário ter o Python instalado (www.python.org). Depois, é preciso ter o MySQL instalado (www.mysql.com). Por último, a nossa peça de ligação entre os dois: http://sourceforge.net/projects/mysql-python. Na minha máquina, o processo não demorou mais de 2 minutos graças ao gestor de pacotes do Ubuntu. Antes de se dirigirem ao site, caso usem linux, verifiquem se no vosso gestor de pacotes já não está uma versão disponível. Para verificarem se está tudo a funcionar, cheguem a uma qualquer consola Python e tentem importar o módulo (cuidado com as maiúsculas!):

```
import MySQLdb
```

Ligação ao servidor

O primeiro passo na ligação à base de dados é a ligação ao servidor MySQL. Por muito complicado que isto possa parecer, faz-se numa linha de código:

```
ligação = MySQLdb.connect('localhost',
'utilizador', 'password')
```

E já está. Caso isto não funcione, convém rever quer a localização do servidor, quer o nome e password do utilizador da base de dados. Claro que podemos substituir cada elemento por uma variável previamente definida pelo utilizador (ou pelo próprio script). Também podemos optar por ligar (ou tentar ligar) directamente a uma base de dados (BD). O código que se segue tira partido desta última abordagem:

```
nome_utilizador = 'utilizador'
palavra_chave = 'password'
servidor = 'localhost'
base_de_dados = 'teste'
MySQLdb.connect(servidor,
nome_utilizador, palavra_chave,
base_de_dados)
```

Se tudo correr bem a consola age silenciosamente, ligando-se ao servidor e à BD em questão. Mas caso nos tenhamos enganado no nome da BD, ou, um caso provável, se esta não existir, muito provavelmente aparece algo como:

```
Traceback (most recent call last):
[ ..... ]
_mysql_exceptions.OperationalError:
(1049, "Unknown database 'teste'")
```

É-nos devolvida uma excepção, que pode ser aproveitada. De facto, uma vez perguntei numa mailing-list se não havia maneira de ligar a um servidor e verificar se dada base de dados ou tabela existia. A resposta foi consensual: mesmo que exista, lidar com estas excepções torna o processo mais simples. A biblioteca MySQLdb traz consigo um método que permite então lidar com as excepções: MySQLdb.Error. Chamando este método através de um except, pode-se aceder a 2 argumentos: o número do erro e a mensagem do erro. No nosso caso, vemos que o erro tem o número 1049. Vamos então melhorar o nosso código para incluir o tratamento deste erro:

```
try:
    connection =
MySQLdb.connect(servidor ,
nome_utilizador, palavra_chave,
base_de_dados)
except MySQLdb.Error, e:
    if e.args[0] == '1049':
            if (raw_input('Base de
dados inexistente. Deseja criar?')) ==
's':
                    connection =
MySQLdb.connect(servidor ,
nome_utilizador, palavra_chave)
                    cursor =
connection.cursor()
                    command = 'CREATE
DATABASE %s' %base_de_dados

cursor.execute(command)
                    cursor.close()
```

Este excerto de código inclui exemplos de métodos ainda não explicados, mas cada coisa a seu tempo. De facto, como se pode observar, a excepção gerada tem 2 argumentos, sendo o primeiro, tal como já havia dito, o número do erro (`e.args[0]`). Podemos depois decidir como lidar com o problema. Neste caso, optei por perguntar ao utilizador se queria criar a base de dados teste, uma vez que esta ainda não existia. Caso a resposta fosse afirmativa, o programa encarregava-se de ligar ao servidor, criar um cursor, construir a query de SQL que cria a base de dados, e executar o comando.

Antes de avançarmos para a definição de cursor, e dos seus métodos, deixo antes um link para uma listagem dos erros, e respectivos códigos numéricos:

http://dev.mysql.com/doc/refman/5.0/en/error-messages-server.html

Cursores

Um cursor não passa de um apontador para a base de dados. De facto é o que nos permite interagir com ela. Irei abordar 2 dos seus propósitos: execução de comandos e recuperação de resultados.

Como vimos no exemplo acima, caso não tivéssemos uma base de dados à qual ligar, éramos confrontados com um erro. Porém, conseguimos contornar esse erro, tirando partido, entretanto, da criação de uma base de dados a partir do nosso script. Como o conseguimos?

```
cursor.execute('CREATE DATABASE %s'
%base_de_dados)
```

Este pequeno pedaço de código ilustra um método associado ao cursor que nos é indispensável: `execute()`. É este método que permite a interacção "directa", por assim dizer, com a base de dados. É através dele que passamos os comandos de SQL e que fazemos perguntas, que criamos tabelas e campos, que preenchemos bases de dados, ou que as actualizamos, ou mesmo destruimos. Como vimos, a sintaxe é relativamente simples: o código SQL é passado como argumento à função `execute()`, sob forma de uma string. Para além do método `execute()`, supondo que queremos efectuar vários comandos similares de uma só vez, por exemplo, uma múltipla inserção, podemos recorrer a um método "irmão", mais "guloso": `executemany()`. A sua sintaxe é semelhante à do `execute()`.

Construamos a lista seguinte, de modo a ser inserida na base de dados a que já estamos ligados:

```
listaValores = [  ('Vermelho',
'Benfica'),  ('Azul', 'Campeão'),
('Verde', 'Sporting')]
```

O código do método `executemany()` será:

```
cursor.executemany("INSERT INTO
tabela1 (campo1, campo2) VALUES (%s,
%s)", listaValores)
```

A vantagem de usar este método está na performance (bem como na organização do código). De facto, só temos que aceder uma vez à base de dados, em vez de múltiplos acessos com vários `execute()`.

O outro método relevante na interacção com as bases de dados é o método `fetchone()`. Até aqui, tínhamos apenas interagido com a base de dados numa relação unidireccional (utilizador –> base de dados). Contudo, o método `fetchone()`, bem como os associados `fetchmany()` e `fetchall()`, permitem-nos tornar essa interacção bidireccional, tornando possível ao utilizador recuperar informação inserida na base de dados. A sintaxe do comando é relativamente simples. Após um `execute()` com uma string de SQL que envolva um SELECT e que nos devolva n resultados, podemos aceder aos resultados

usando uma das três funções - `fetchone()`, `fetchmany()`, `fetchall()` - consoante queiramos um, vários, ou todos os resultados. Para definirmos o número de resultados a ver no `fetchmany()` usamos outro método do cursor: `arraysize`. Este, aceita um valor inteiro que, por defeito, é 1. Segue-se um exemplo:

```
resultado = cursor.fetchone()
```

Este método devolve uma tupla de resultados, contendo uma linha por cada resultado, e caso o SQL não devolva resultados, o método devolve `None`. Simples e eficaz. Para aceder ao conteúdo da linha podemos ou imprimir a variável linha ou, caso queiramos uma coluna em específico, aceder tal como a uma lista: `resultado[0]`, `resultado[2]`, etc.

Outro método útil é o `rowcount()`. Este devolve o número de linhas afectadas/produzidas pelo último `execute()` (ou `executemany()`). Exemplificando, um SELECT que devolva um resultado de uma linha, tem um `rowcount()` igual a 1, enquanto um INSERT que afecte, digamos, 3 linhas, tal como o caso dos clubes lá em cima, dá um `rowcount()` igual a 3.

Por fim, um último método que deve ser sempre chamado no final de todos os acessos à base de dados, é o `commit`. Chama-se tendo como classe-mãe a `connection` e sem argumentos. Depois deste `commit`, fecha-se o acesso à base de dados, pelo método close, também derivado da classe `connection`. Pondo estas ideias na prática:

```
connection.commit()
connection.close()
```

Conclusão

Como vimos, interagir com bases de dados em MySQL através de scripts de Python torna-se, graças à MySQLlib, como dizem os ingleses, "a walk in the park", ou, em bom português, "como tirar o doce a uma criança". Uma vantagem interessante desta biblioteca, é que o seu código é exactamente igual ao usado por outras bibliotecas que acedem a outros provedores de bases de dados, como a SQLite ou o PostgreSQL. Deixo, em jeito de despedida, links para bibliotecas em Python para fazer ligação a outras bases de dados, bem como o desafio de lhes aplicarem este código e, ou me dizerem que funciona, ou me enviarem hate-mails porque não funciona!

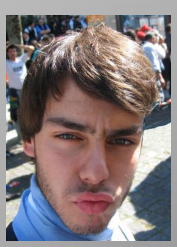
PostgreSQL:

- http://pypgsql.sourceforge.net/
- http://www.initd.org/
- http://www.pygresql.org/

SQLite

- sqlite3 (incluída com o Python 2.5)

SOBRE O AUTOR



Estranhamente, João Rodrigues não vem das hostes dos informáticos tendo tido o primeiro contacto com a programação no 3º ano do curso. Um descontraído finalista da Licenciatura em Bioquímica em Coimbra, com particular interesse pelas áreas da Bioinformática e da Proteómica, entrou para a Revista PROGRAMAR e com o objectivo de mostrar que há mais na informática para além de bits e bytes.

João Rodrigues

O Namespace MY em VB.Net 2005

O VB.NET 2005 introduziu uma característica muito interessante em relação às outras versões, criando o namespace My. O namespace My contém um conjunto de objectos que simplifica o acesso a diversas áreas do .NET Framework, como o sistema de ficheiros ou dados da aplicação. Este namespace não está disponível noutras linguagens .NET. Para além de reduzir a quantidade de código (na implementação de diversas funcionalidades era necessário recorrer a APIs) o namespace My expõe alguns objectos que são gerados dinamicamente à medida que se acrescentam características aos projectos. Por exemplo, cada classe form que se acrescenta a um projecto fica disponível através de uma propriedade distinta do objecto My.Forms.

Objectos do Namespace My

• My.Application - expõe informação acerca da aplicação actual, tal como o seu percurso no sistema de ficheiros, a sua versão, e modo de autenticação do utilizador;

• My.Computer - encerra um conjunto de subclasses destinadas a lidar com as características mais importantes do computador, tais como o sistema de ficheiros, subsistemas de áudio e vídeo, memória, teclado, rato, rede, portas série, impressoras, etc;

• My.Forms - apresenta uma propriedade por cada classe form definida no projecto e possibilita referenciar a instância desse form sem se ter de criar explicitamente um objecto da mesma;

• My.Resources - possui um objecto derivado por cada recurso(mais à frente explicarei melhor o que é um recurso) definido no projecto;

• My.Settings - expõe uma propriedade por cada valor de configuração(setting) definida nas configurações do projecto;

• My.User - devolve informação sobre o utilizador actualmente registado, possibilitando a implementação de mecanismos de autenticação;

• My.WebServices - mantém uma propriedade por cada serviço Web que o projecto referencie, possibilitando o acesso aos mesmo sem haver a necessidade de criar um objecto proxy de cada vez que seja necessário invocar um método no serviço Web(este não será tão abordado).

My.Application

Aqui é possível obter aspectos relacionados com a aplicação em si, tal como a versão, o título, quem a produziu, a empresa, entre outros... Vamos criar um botão que nos mostra uma messagebox com alguma desta informação.

```
Dim sbInfo As New
System.Text.StringBuilder
With My.Application.Info
    sbInfo.Append("Título: " & .Title
& vbNewLine)
    sbInfo.Append("Versão: " &
.Version.ToString & vbNewLine)
    sbInfo.Append("Descrição: " &
.Description & vbNewLine)
    sbInfo.Append("Empresa: " &
.CompanyName & vbNewLine)
    sbInfo.Append("Direitos: " &
.Copyright)
End With
MessageBox.Show(sbInfo.ToString(),
"Informação do Sistema",
MessageBoxButtons.OK,
MessageBoxIcon.Information)
```

Podemos também chamar a aplicação com parâmetros.

```
Dim fInfo As New
IO.FileInfo(Application.ExecutablePath)

System.Diagnostics.Process.Start(fInfo.
Name.ToString(), "m nome:anolsi")
'Aqui é chamada uma nova instância da
aplicação
```

No event load do form principal obtemos os parâmetros:

```
For Each s As String In
My.Application.CommandLineArgs
     Select Case s
          Case "m" :
MessageBox.Show("Bem-Vindo")
'Argumento que mostra a mensagem de
boas vindas
```

```
            Case "nome:" & ChrW(0) To
"nome:" & ChrW(65535) :
MessageBox.Show(s.Substring(5))
'Argumento que mostra o nome do
utilizador
        End Select
Next
```

My.Computer

Através do My.Computer vamos ler um ficheiro para uma textbox, alterá-lo e depois gravá-lo. Antes de mais, no código, vamos acrescentar duas linhas no início do código antes de "Public Class ..."(zona de declarações):

```
'Aqui iremos declarar constantes.
Public Const FICHEORIGINAL As String =
"c:\exemplo.txt"
Public Const FICHENOVO As String =
"c:\copia.txt"

Private Sub btnAbrir_Click(ByVal
sender As System.Object, ByVal e As
System.EventArgs) Handles
btnAbrir.Click
    If
My.Computer.FileSystem.FileExists(FICHE
ORIGINAL) Then 'Verifica se o ficheiro
existe
       Me.txtFiche.Text =
My.Computer.FileSystem.ReadAllText(FICH
EORIGINAL) 'Lê o texto todo para a
TextBox
    End If
End Sub
Private Sub btnGravar_Click(ByVal
sender As System.Object, ByVal e As
System.EventArgs) Handles
btnGravar.Click

My.Computer.FileSystem.WriteAllText(FIC
HEORIGINAL, Me.txtFiche.Text, False)
'Escreve o texto da TextBox para o
ficheiro sem adicionar (sobrepõe).
End Sub
Private Sub btnCopiar_Click(ByVal
sender As System.Object, ByVal e As
System.EventArgs) Handles
btnCopiar.Click

My.Computer.FileSystem.CopyFile(FICHEOR
IGINAL, FICHENOVO) 'Copia o ficheiro
FICHEORIGINAL para FICHENOVO
End Sub
```

Este constitui ainda uma excelente porta de comunicação entre o programa e os periféricos do computador, como o rato (My.Computer.Mouse), o teclado (My.Computer.Keyboard), o monitor (My.Computer.Screen), as Portas (My.Computer.Ports). Ainda mais importante é o My.Computer.Network, que nos permite fazer Ping, DownloadFile, UploadFile... Funções que em versões anteriores nos obrigavam a ter mais linhas de código.

Aqui está uma maneira simples de verificarmos o acesso ao P@P:

```
If My.Computer.Network.IsAvailable()
Then 'Verifica se está ligado a alguma
rede
     Try 'O Try...Catch apanha o erro
no caso de estar ligado a uma rede mas
não à internet
          If
My.Computer.Network.Ping("portugal-a-
programar.org") Then 'Tenta fazer um
ping para "portugal-a-programar.org"

MessageBox.Show("Acessível!", "Teste
de Ligação ao P@P",
MessageBoxButtons.OK,
MessageBoxIcon.Information)
          Else

MessageBox.Show("Inacessível!", "Teste
de Ligação ao P@P",
MessageBoxButtons.OK,
MessageBoxIcon.Information)
          End If
     Catch ex as Excepcion
          MessageBox.Show("Não foi
possível realizar o pedido de Ping.",
"Erro!", MessageBoxButtons.OK,
MessageBoxIcon.Error)
     End Try
Else
     MessageBox.Show("Não está ligado
a nenhuma rede.", "Erro!",
MessageBoxButtons.OK,
MessageBoxIcon.Error)
End If
```

Como podem verificar é bastante simples de usar. É claro que as nossas constantes poderiam ser variáveis, obtidas até a partir de uma dialogbox onde iríamos obter o ficheiro para abrir ou para gravar, ou mesmo para copiar. No entanto isso fica fora das competências do namespace My.

My.Forms

O My.forms permite, pura e simplesmente, acesso aos formulários criados, sendo também possível recorrendo simplesmente ao nome do formulário. Exemplo:

```
frmPrincipal.text = "Aqui fica o título!"
My.Forms.frmPrincipal.text = "Aqui fica o título!"
```

A única vantagem consiste na ajuda do intelliSense que é disponibilizada. Ao escrevermos "My.Forms.", o intelliSense apresenta todos os formulários criados até à altura. Outro exemplo é verificar se um determinado formulário está ou não a ser exibido e caso não esteja, mostra-o:

```
If My.Forms.frmSec Is Nothing Then
My.Forms.frmSec.Show()
```

My.Resources

Os recursos estão muito ligados às settings, as quais vamos tratar a seguir, mas constantes porque não é possível ser alterado a nível do código em run-time. Vamos criar um exemplo onde iremos colocar os valores das constantes criadas a propósito do My.Computer.

1. Abrir o My Project no Solution Explorer e ir para o separador Resources;
2. Iremos dar-lhes o mesmo nome FICHEORIGINAL e FICHENOVO com os mesmos valores.

```
Private Sub btnAbrir_Click(ByVal
sender As System.Object, ByVal e As
System.EventArgs) Handles
btnAbrir.Click
    If
My.Computer.FileSystem.FileExists(My.Re
sources.FICHEORIGINAL) Then
Me.txtFiche.Text =
My.Computer.FileSystem.ReadAllText(My.R
esources.FICHEORIGINAL)
End Sub
Private Sub btnGravar_Click(ByVal
sender As System.Object, ByVal e As
System.EventArgs) Handles
btnGravar.Click

My.Computer.FileSystem.WriteAllText(My.
Resources.FICHEORIGINAL,
Me.txtFiche.Text, False)
End Sub

Private Sub btnCopiar_Click(ByVal
sender As System.Object, ByVal e As
System.EventArgs) Handles
btnCopiar.Click

My.Computer.FileSystem.CopyFile(My.Reso
urces.FICHEORIGINAL,
My.Resources.FICHENOVO)
End Sub
```

My.Settings

Tal como referi anteriormente o que distingue as settings dos resources é o facto de as settings se comportarem como variáveis globais, enquanto os resources como constantes globais. Ainda de referir que as settings podem ser de qualquer tipo como as variáveis mas que não são "esquecidas" ao terminar o programa. Estas possuem uma extrema relevância, uma vez não necessitamos de guardar as variáveis em ficheiros ou no registry do Windows, nem que sejamos confrontados com a necessidade de proteger a informação, ou garantir que ela não é alterada. A própria linguagem encarrega-se disso. Para criarmos uma nova setting, vamos ao My Project e, como não poderia deixar de ser, ao separador Settings. Vamos criar uma setting cuja função é guardar o último tamanho do formulário. Essa setting será do tipo System.Drawing.Size. Convém ter um valor por defeito. Depois inserirmos o seguinte código:

```
Private Sub frmPrincipal_Load(ByVal
sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
    Me.Size = My.Settings.Tamanho
End Sub
Private Sub
frmPrincipal_ResizeEnd(ByVal sender As
Object, ByVal e As System.EventArgs)
Handles Me.ResizeEnd
    My.Settings.Tamanho = Me.Size
    My.Settings.Save()
End Sub
```

Para voltarmos a coloca o valor por defeito das settings, devemos incluir esta instrução quando o quisermos fazer:

```
My.Settings.Reset()
```

My.User

O My.User serve essencialmente para saber se o utilizador está autenticado, qual o nome do utilizador e se este é administrador, limitado, convidado, etc. De salientar que se o My.User for usado em aplicações Windows este refere-se ao utilizador que está a executar a aplicação, enquanto em aplicações web este refere-se a quem acede à aplicação e não onde ela está a correr.

```
If My.User.IsAuthenticated Then
'Verifica se o utilizador está
autenticado
      'O valor do My.User.Name será
algo do tipo
"Nome_do_Computador\Utilizador" e por
isso vamos separá-los pela \
      Dim parte() As String =
My.User.Name.Split("\"c)
      Dim tipo As String = ""
      If
My.User.IsInRole(ApplicationServices.Bu
iltInRole.Administrator) Then
'Verifica se é administrador através
do IsInRole
            tipo = "É Administrador"
      Else
            tipo = "Não é
administrador"
      End If

MessageBox.Show(String.Format("Nome do
Computador: {0} " + vbNewLine +
"Utilizador: {1} " + vbNewLine +
"{2}", parte(0), parte(1), tipo),
"Utilizadores", MessageBoxButtons.OK,
MessageBoxIcon.Information)
Else 'Caso o utilizador não esteja
autenticado (caso do convidado) a
aplicação termina
      MessageBox.Show("Não está
autenticado.", "Erro!",
MessageBoxButtons.OK,
MessageBoxIcon.Error)
      Me.Close()
End If
```

Conclusão

Como podemos constatar, a existência deste namespace veio trazer muitas vantajens ao programador, servindo até como um ponto de partida para as funcionalidades mais simples e, mesmo assim, aqui apenas foram apresentadas algumas funcionalidades do namespace My, porque o artigo não possui extensão suficiente para serem abordadas todas as suas funcionalidades. Mesmo assim o leitor pode, a partir deste momento, explorar sozinho todas as outras funcionalidades do namespace.

SOBRE O AUTOR

Frequentando actualmente o 2ºano do Curso Profissional de Informática de Gestão, António Silva adora programar em VB e C. Estou também a aprender JavaScript, PHP e ainda XHTML.

António Silva

Engenharia de Software

Introdução

A engenharia de Software(ES) pode ser considerada uma das áreas mais abstractas, mais importantes, porém, é das áreas menos exploradas da Informática. Esta subsecção da Informática aborda de forma precisa e consistente os passos para a criação de uma boa aplicação de Software, desde a sua génese até ao lançamento do produto. Neste pequeno artigo, será abordado de forma simples e básica o Engenharia de Software, e os conceitos adjacentes.

Estudo de fiabilidade

Metaforicamente falando é possível fazer uma analogia entre a criação de uma aplicação de software e a construção de um edifício. Ambos são faseados. O plano de fiabilidade é uma fase crítica para concretização de um projecto, dado ser neste ponto que a equipa de desenvolvimento verifica se existem condições para a sua concretização. Isto acontece pois é necessário ter certezas absolutas que o software terá a menor quantidade de falhas possível. Para facilmente absorvermos este conceito, basta imaginarmos um avião, que hoje em dia é pilotado automaticamente por uma aplicação de software extremamente complexa. Um pequeno "bug" por si só basta para que possam existir erros no sistema de voo e causar a morte a dezenas/centenas de pessoas e prejuízos de milhares de euros.

Documentação

Uma boa peça de software está devidamente documentada. Deste modo, facilita-se não só a apresentação dos requisitos, como também é possível fazer uma boa manutenção/evolução. Assim, existem três fases que dão origem a três documentos fulcrais: Requirement Analysis, Requirement Defination e Requirement Specification.

• Requirement Analysis

É um documento simples, resultante de uma reunião entre a equipa de desenvolvimento, ou o gestor da equipa, e o cliente. Funciona como uma espécie de acta, onde são apresentados de forma corrente (linguagem normal), os pedidos do cliente e aquilo que será possível fazer. É possível também encontrar diagramas e maquetas, que facilitem a compreensão por parte do seu leitor.

• Requirement Definition

Já se trata de um documento mais cuidado, onde os dados já foram filtrados e estruturados por grau de importância/risco. Continua a ser escrito em linguagem corrente, e a contar com alguns diagramas/maquetas.

• Requirement Specification

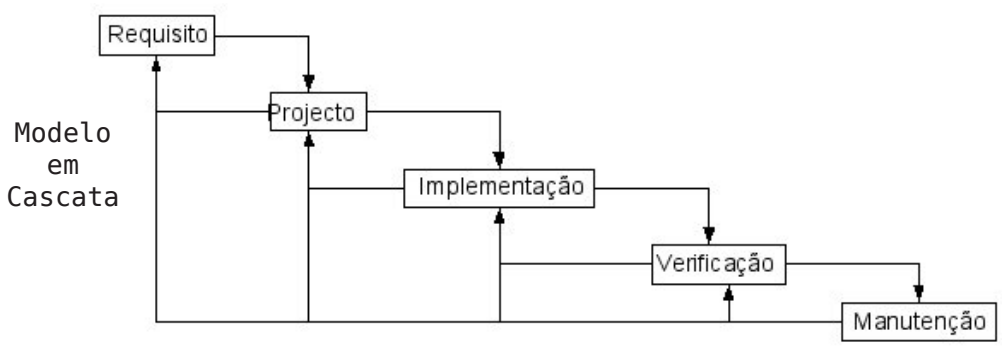
Nesta fase, estudam-se os resultados da discussão com o cliente e cria-se um documento mais "abstracto" e "profissional", nele já poderá existir algum algoritmo/bloco de código que esclareça situações dúbias. Ocasionalmente, devido ao seu nível de precisão, este documento pode ser utilizado com contrato entre ambas as partes.

Modelos de desenvolvimento de aplicações

Existem vários modelos de desenvolvimento. Aqui apenas serão abordados de forma superficial dois deles, que influenciaram de forma muito significativa, Modelo em cascata e Modelo em espiral.

• Modelo em Cascata

É o modelo mais simples, e também mais desactualizado. No entanto, para se falar neste modelo, é necessário fazer uma pequena alusão à época em que foi usado. Foi proposto por volta dos anos 70, por Royce, e teve um grande impacto pela sua simplicidade/facilidade de utilização. É necessário recordar que nessa época os compiladores (que existiam) levavam bastante tempo a fazer o seu trabalho, por isso

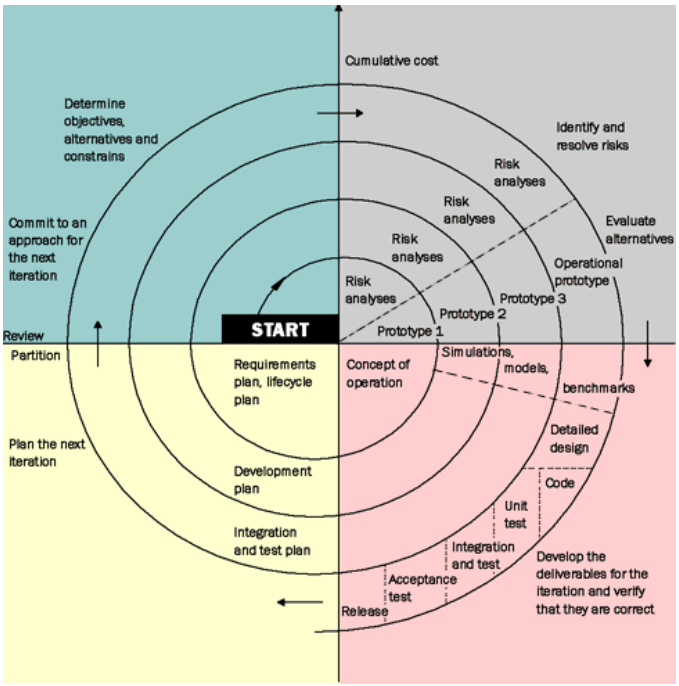


uma aplicação não podia ser baseada em tentativa erro, mas sim em fundamentos teóricos. Este modelo veio consolidar esses factos.

No modelo em Cascata (WaterFall), para se poder avançar uma fase, é necessário que a fase anterior esteja completamente terminada, como se pode ver pela imagem. Este modelo é sequencial, e qualquer fase pode ser revista em qualquer momento do desenvolvimento.

• Modelo em Espiral

O modelo em espiral é um modelo ainda hoje muito utilizado. Foi apresentado no final da década de 80 (1988) por Boehm. Trata-se de um modelo iterativo, em que existe em cada momento de do ciclo, pelo menos uma análise de riscos e a criação de um protótipo (como se pode verificar pela imagem abaixo). Este modelo ainda é muito actual, e apesar de aparentar ser bastante complexo, permite a criação de aplicações muito fiáveis.



Validação/Verificação

Validação e verificação são dois conceitos, que apesar de terem escrita diferente, têm significados muito idênticos, e por isso são muitas vezes confundidos. Em 1979, Boehm, "criou" uma definição bastante simplista sobre estes dois conceitos, que permite a sua fácil distinção:
- Validation - *"Are we building the right product?"*
- Verification - *"Are we building the product right?"*

Sendo assim, é possível afirmar que a validação de uma aplicação de software consiste na certeza de que o produto corresponde às especificações do cliente e aos requisitos impostos pela equipa de desenvolvimento, enquanto a verificação consiste na confirmação que o software contém o menor número de bugs possível, que é estável e robusto.

Evolução/Manutenção

Sendo parte integrante do processo de desenvolvimento de uma qualquer aplicação de software, a evolução/manutenção, estão dependentes de uma documentação precisa, rigorosa, e bem estruturada. Isto porque muitas vezes quem oferece suporte a uma aplicação ou quem a fará evoluir, devido às necessidades dos utilizadores, não é a equipa que desenvolveu o software. Sendo assim, caso não haja bons documentos, será necessária muita engenharia reversa, o que fará os custos do software aumentar abruptamente.

Custos

Para além de englobar métodos de trabalho, documentação, a engenharia de software engloba ainda um área de análise de custos.
Esta análise de custos não se refere apenas a analisar o custo financeiro que poderá ter o desenvolvimento de uma determinada aplicação, mas também os custos humanos/psicológicos. Neste ponto entra a agilidade de um bom gestor/coordenador da equipa de desenvolvimento, que terá por obrigação manter a equipa de desenvolvimento motivada para que não existam "derrapagens", e deste modo tudo seja concretizado dentro dos tempos previstos.

SOBRE O AUTOR

Francisco Ribeiro, estudante do segundo ano da Licenciatura em Engenharia Informática, na Universidade do Minho. Tem interesse especial por haskell, C, e Linux. Como autodidacta, estuda AS2 e Python.

Francisco Ribeiro

Proposta Sintáctica: Linguagem Projecto de Programação

Introdução

Este artigo propõe e discute um formato padrão para a elaboração da documentação escrita da linha de raciocínio lógico do código de programação de computadores em língua portuguesa, por meio do uso de uma Linguagem de Projecto de Programação (LPP), de maneira que possa tal código textual ser traduzido para qualquer linguagem formal estruturada de programação de computadores.

O tema aqui exposto não é assunto inédito, pois já foi discutido em 1975 no artigo "PDL - A Tool for Software Design", escrito pelos investigadores Stephen H. Caine e E. Kent Gordon. Neste artigo os autores apresentam uma forma de escrita que objectiva representar, de forma clara, as acções a serem executadas em um computador sem levar em consideração o uso específico de uma linguagem formal de programação de computadores. Não se trata de apresentar uma tradução da proposta PDL, mas sim formatar e propor critérios que permitam o uso de uma nomenclatura em língua portuguesa mais equilibrada e homogénea em relação à forma que vem sendo utilizada. Deseja-se assim sugerir algumas regras de escrita que possam evitar a multiplicidade de dialectos encontrados.

O factor que motivou a elaboração desta discussão decorre do facto de vários profissionais da área de desenvolvimento de programas de computadores principalmente no Brasil ao fazerem (quando fazem) uso de alguma técnica semelhante a técnica PDL não o fazem segundo uma regra ou forma única, cada qual, quando a usa, fá-lo de uma forma particular e diferente, ocasionando dificuldades no uso da técnica.

LINGUAGEM DE PROJECTO DE PROGRAMAÇÃO – LPP

Uma das questões aqui discutidas é o facto de se definir um formato textual padrão e homogéneo para a utilização dos comandos de pseudocódigo português a ser utilizado na documentação do código de um programa de computador. Além da preocupação em relação às palavras de comandos a serem definidas, existe a questão relacionada com o nome da técnica em português, pois vários são os nomes sugeridos, como: portugol (GUIMARÃES & LAGES, 1994), linguagem de descrição (VELOSO, et. al., 1996), linguagem universal de programação (TERADA, 1991), linguagem de leitura simples (SZWARCFITER & MARKENZON, 1994), e outras tantas formas encontradas e definidas no sentido de tentar representar e indicar a mesma ideia.

O facto de cada autor ou profissional definir uma forma particular de escrever a estrutura funcional de um código computacional e também o seu nome de identificação não caracteriza nenhuma espécie de erro, mas estabelece uma série de dialectos que se tornam confusos e geram interpretações e discussões desnecessárias sobre que forma é ou não válida.

A tabela 1 mostra as palavras-chave de pseudocódigo usadas pelos autores pesquisados e suas correlações com as linguagens formais de programação de computadores PASCAL e STRUCTURED BASIC e também a proposta de comandos da Linguagem de Projecto de Programação.

Para melhor interpretação da tabela considere o nome de cada coluna segundo a legenda:

BAS – Código em Structured BASIC;
GUL – Guimarães e Lages;
LPP – Linguagem de Projecto de Programação;
PAS – Código Pascal;
SZM – Szwarcfiter & Markenzon;
TER – Terada;
VEL – Veloso, et. al.

A indicação das linguagens formais de programação PASCAL e STRUCTURED BASIC são consideradas no sentido de ajudar a demonstrar o motivo em propor a definição de uma regra de codificação mais genérica para a forma LPP em relação ao formato utilizado pelos demais autores. Note que a LPP engloba um conjunto maior de instruções em relação ao conjunto de instruções das linguagens PASCAL e STRUCTURED BASIC e também do conjunto proposto no trabalho particular de cada um dos autores apontados.

Os campos da tabela que se encontram grafados em branco em qualquer uma das formas apresentadas mostram a ausência de comandos referentes às acções indicadas pelos autores de cada obra consultada. Quanto aos campos relacionados com as linguagens formais de programação PASCAL e STRUCTURED BASIC que estão em branco, estes indicam a ausência da acção na linguagem em si.

Além da proposta de padronização do conjunto de palavras reservadas, há a necessidade de definir um padrão para a representação genérica de uso dos operadores aritméticos, operadores lógicos e operadores relacionais. Assim sendo,

segue-se na tabela 2 a apresentação e as comparações entre as várias definições dos operadores aritméticos, lógicos e relacionais a serem utilizados pela LPP.

Definida a proposta de estrutura de codificação a ser utilizada pela LPP cabe mostrar na tabela 3 a relação dos comandos propostos e sua classificação sintáctica dentro da esfera da língua portuguesa, de acordo com a representação da acção a ser executada por cada um dos comandos.

Uma linguagem de programação de computadores de alto nível (mesmo que seja uma linguagem de documentação como é a proposta da LPP) é formada por um conjunto de preposições, conjunções, substantivos, adjectivos, advérbios, verbos e interjeições e devido a estas características recebe o nome de linguagem de programação, que é o conjunto de instruções e regras de composição e encadeamento, pelo qual se expressam acções que são executadas por um computador. Assim sendo, passa-se a ter um instrumento eficiente e eficaz de comunicação.

Conclusão

Este artigo não pode ser conclusivo, uma vez que abre a proposta de discussão em apresentar uma forma de documentação que seja oficializada para uma representação mais formal e homoénea do formato escrito da linha de raciocínio computacional. Sugere-se assim uma forma mais homogénea de escrita que seja útil para os profissionais de desenvolvimento de software, professores, alunos e demais interessados.

GUL	VEL	TER	SZM	PAS	BAS	LPP
ATÉ	ATÉ	ATÉ		TO	TO	ATÉ
ATÉ	ATÉ	FIM-REPITA		UNTIL	LOOP	ATÉ_QUE
CARACTER	CAR			STRING	STRING	CARACTERE
				CASE	SELECT	CASO
VETOR/MATRIZ	VET			ARRAY	DIM	CONJUNTO
DE	DE	←		:=	=	DE
ENQUANTO	ENQUANTO	ENQTO	ENQUANTO	WHILE	WHILE	ENQUANTO
ENTÃO	ENTÃO	ENTÃO	ENTÃO	THEN	THEN	ENTÃO
IMPRIMA()	ESCREVA()	PARE-COM-SAÍDA()		WRITE()	PRINT	ESCREVA
FAÇA	FAÇA	FAÇA	FAÇA	DO		FAÇA
FIM	FIM	PARE		END		FIM
				END	END SELECT	FIM_CASO
FIM_ENQUANTO		FIM-ENQTO			WEND	FIM_ENQUANTO
FIM_PARA		FIM-PARA	PARE		NEXT	FIM_PARA
FIM_REGISTRO				END	END TYPE	FIM_REGISTRO
FIM_SE		FIM-SE		;	END IF	FIM_SE
INÍCIO	INÍCIO			BEGIN		INÍCIO
INTEIRO	INT			INTEGER	AS INTEGER	INTEIRO
LEIA()	LEIA()			READ()	INPUT	LEIA
LÓGICO	LOG			BOOLEAN		LÓGICO
PARA	PARA	PARA	PARA	FOR	FOR	PARA
PASSO	INCR				STEP	PASSO
PROCEDIMENTO	PROC		PROCEDIMENTO	PROCEDURE	SUB	PROCEDIMENTO
			ALGORITMO	PROGRAM		PROGRAMA
REAL	REAL			REAL	AS SINGLE	REAL
REGISTRO	REG			RECORD		REGISTRO
REPITA	REPITA	REPITA		REPEAT	DO	REPITA
SE	SE	SE	SE	IF	IF	SE
				OF	CASE	SEJA
SENÃO	SENÃO	SENÃO	SENÃO	ELSE	ELSE	SENÃO
TIPO	TIPO			TYPE	TYPE	TIPO
	VAR			VAR	DIM	VAR

GUL	VEL	TER	SZM	PAS	BAS	LPP
+	+	+		+	+	+
–	–	–		–	–	–
*	*	*		*	*	*
/	/	/		/	/	/
DIV	DIV			DIV		DIV
←	←	←	:=	:=	=	←
**					^	↑
E	&			AND	AND	.E.
OU	V			OR	OR	.OU.
NÃO	¬			NOT	NOT	.NÃO.
>	>	>	>	>	>	>
<	<	<	<	<	<	<
≥	≥	≥	≥	>=	>=	>=
≤	≤	≤	≤	<=	<=	<=
=	=	=	=	=	=	=
≠	≠	≠	≠	<>	<>	<>

CONJUNTO	Adjectivo
DE	Preposição
ENQUANTO	Conjunção
ENTÃO	Advérbio
ESCREVA	Verbo (Imperativo Afirmativo)
FAÇA	Verbo (Imperativo Afirmativo)
FIM	Substantivo Masculino
FIM_CASO	Substantivo Masculino com Substantivo Masculino
FIM_ENQUANTO	Substantivo Masculino com Conjunção
FIM_PARA	Substantivo Masculino com Preposição
FIM_REGISTRO	Substantivo Masculino com Substantivo Masculino
FIM_SE	Substantivo Masculino com Conjunção
FUNÇÃO	Substantivo Feminino
INICIO	Substantivo Masculino
INTEIRO	Adjectivo
LEIA	Verbo (Imperativo Afirmativo)
LÓGICO	Adjectivo
PARA	Preposição
PASSO	Substantivo Masculino
PROCEDIMENTO	Substantivo Masculino
PROGRAMA	Substantivo Masculino
REAL	Substantivo Masculino
REGISTRO	Substantivo Masculino
REPITA	Verbo (Imperativo Afirmativo)
SE	Conjunção
SEJA	Interjeição
SENÃO	Conjunção
TIPO	Substantivo Masculino
VAR (variável)	Substantivo Feminino

Bibliografia

CAINE, S. H.; GORDON, E. K. PDL: A Tool for Software Design. In PROCEEDINGS OF THE I975 NATIONAL COMPUTING CONFERENCE, 1975, Anaheim, CA. Montvale, NJ: AFIPS Press, 1975, 271-276.

GUIMARÃES, A. de M.; LAGES, A. de C. Algoritmos e Estruturas de Dados. 18. ed. Rio de Janeiro: Livros Técnicos e Científicos, 1994, 216 p.

SZWARCFITER, J. L.; MARKENZON, L. Estruturas de Dados e seus Algoritmos. Rio de janeiro: Livros Técnicos e Científicos, 1994, 320 p.

TERADA, R. Desenvolvimento de Algoritmos e Estruturas de Dados. São Paulo: Makron Books, 1991, 255 p.

VELOSO, P. et. al. Estruturas de Dados. Rio de Janeiro: Campus, 1996, 228 p.

SOBRE O AUTOR

Natural da Cidade de São Paulo, Augusto Manzano tem 23 anos de experiência em ensino e desenvolvimento de programação de software. É professor da rede federal de ensino no Brasil, no Centro Federal de Educação Tecnológica de São Paulo. É também autor, possuindo na sua carreira mais de cinquenta obras publicadas. .

Augusto Manzano

Network Scanning

Quando chega a hora de verificar se uma rede é ou não segura, a melhor opção é talvez fazer um scan à rede e verificar se existem falhas ou outras situações que possam comprometer a integridade de uma rede. Basicamente para cada ataque a uma determinada rede é necessário levar em conta dois parâmetros e são eles o endereço de IP e o número da porta do host que vamos "atacar" ou testar. Por exemplo, se tivermos um exploit para o IIS será necessário o endereço de IP do host que esteja a correr o IIS e talvez necessitemos do número da porta no caso de o IIS não estar a usar a porta standard.

Ora temos aqui um obstáculo. É necessário termos estes dois parâmetros para que possamos efectuar qualquer tipo de tentativa quer para atacar quer para testar as defesas de uma rede, mas como é que podemos então saber qual o endereço IP de um host assim como as portas abertas e os serviços que estão a correr nele?

Essa é a função de um network scanner. Na verdade este tipo de ferramentas são muito úteis não só para verificar estes parâmetros, como também para sabermos a topologia de uma rede ou até mesmo verificar as regras de um firewall.

Como é que os scanners funcionam?

Apesar de existir uma grande variedade de scanners, praticamente todos eles seguem os mesmos princípios. Que princípios? Como é que aplicações, como por exemplo o FTP, funcionam? Estas aplicações usam frames para enviar informação. No entanto uma aplicação não começa logo a enviar dados. É necessário primeiro proceder a uma série de averiguações que vão definir por exemplo qual a sequência das frames, qual a quantidade de informação enviada antes de receberem um acknowledgement e muitos outros parâmetros.

Os scanners usam estas "conversas" para fazer o scan, eles enviam frames para o host e depois esperam a resposta. Tão simples como isto. Se for dada uma resposta é bem provável que determinada aplicação esteja a correr. Mas, existe sempre um mas, não se deixem enganar porque existem programas como o PortSentry cujo objectivo é confundir os scans e que alguns administradores instalam para confundir e boicotar as tentativas de scan.

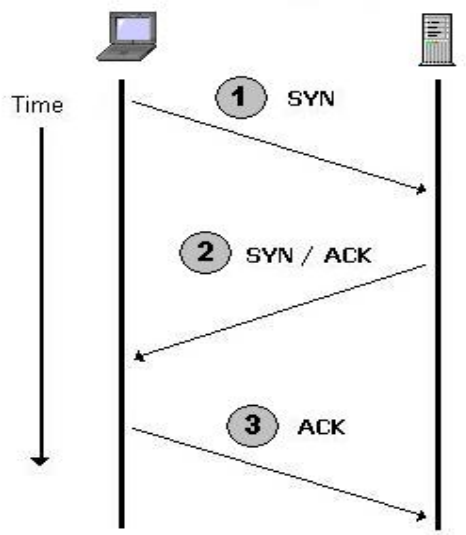
Mas como é que um Network Scanner determina qual a aplicação que está a correr?

Para respondermos a esta pergunta temos que analisar como o protocolo TCP/IP funciona. Grande parte das aplicações usa dois protocolos da Camada 4 do modelo OSI: o protocolo TCP (Transmission Control Protocol) e o UDP (User Datagram Protocol). Ambos os protocolos têm que comunicar com as camadas acima e com diversas aplicações e sessões, assim para não se confundir os dados usam-se portas que permitem múltiplas conexões existir apesar de ser usado apenas um IP. Tanto o UDP como o TCP podem usar ambos 65536 portas. Muitos protocolos e programas têm portas definidas. Os network scanners (NS) determinam quais as aplicações a correr testando as portas mais usadas e verificar quais aceitam conexões. Além de verificar quais as portas abertas, certos NSs ainda conseguem ir mais longe: conseguem verificar a versão da aplicação que está a correr. Exemplo de um NS que consegue fazer isso é o famoso Nmap. Vamos então ver como funciona o scanning em TCP e depois em UDP, porque existem certos parâmetros que devem ser conhecidos. Não nos interessa saber usar um programa se não soubermos como ele funciona.

Scanning usando o TCP

O objectivo deste scan é como é óbvio determinar quais são as portas TCP que tem aplicações a correr. Usando o protocolo TCP é possível descobrir quais as portas abertas sem ser necessário para isso completar a conexão. Como?

O protocolo TCP é um protocolo que presta um serviço fiável, no sentido em que aquilo que o TCP recebe é aquilo que envia, contrariamente ao que acontece com o UDP, usando para isso o flow control e uma comunicação connection-oriented.



Temos que compreender que o protocolo TCP, contrariamente ao UDP, é um protocolo que prima pela fidelidade e integridade da conexão. E isso é visto na maneira como o TCP inicia uma conexão, usando o three-way handshake. Como podemos ver pela figura, o host primeiro envia um "acordo de conexão" com um pacote SYN (de sincronização). Depois o server enviam um pacote de ACK (acknowledgment) e estabelece as regras. A sincronização dos pacotes também é definida nesta altura o que requer o tal pacote com o SYN. Por fim o ultimo segmento é também um ACK do host que notifica o server de que as regras de conexão foram aceites e a conexão está estabelecida e que os dados vão começar a ser transferidos.

Assim para verificar se uma aplicação está á escuta numa determinada porta, o NS pode enviar um pacote com TCP SYN e esperar pelo resultado. Se a resposta for um SYN/ACK podemos dizer que a porta está aberta, se for RST podemos dizer que a porta está fechada, se não houver uma resposta podemos dizer que ou uma firewall está a bloquear as conexões a essa porta, ou simplesmente não existe um serviço a escutar naquela porta do host com aquele IP address.

Existem diversos tipos de scans com TCP, isto porque existem 6 tipos diferentes de flags nas frames enviadas:

- URG (URGent pointer);
- ACK (ACKnowledgement);
- PSH (PuSH);
- RST (ReSeT).
- SYN (SYNchronisation);
- FIN (FINished).

A flag URG é usada para identificar informação que (logicamente) é urgente. A vantagem do uso desta flag é que esses segmentos passam á frente de todos os outros e processados imediatamente.

A flag ACK é usada para avisar que a recepção de um pacote foi bem sucedida. Mas já estão a ver como seria se em cada pacote enviado fosse recebido um ACK, por isso é possível definir a quantidade de pacotes a receber antes de se enviar um ACK. A flag PSH que, similarmente à URG, garante que a informação seja trata com prioridade mas acompanha o processo dos dados do princípio ao fim do envio. Na verdade, é um pouco mais complexo do que isto, mas apenas estamos a fazer um pequeno resumo.

A flag RST é usada quando um segmento chega mas por engano ou quando o serviço que é requisitado não está disponível. Esta é uma das flags usadas para verificar se determinada aplicação está a correr.

A flag SYN talvez a mais conhecida de todas e que é usada no 3-way Handshake, como vimos acima.

A flag FIN é usada para terminar as conexões e caminhos virtuais criadas pela SYN e é acompanhada pela flag ACK para confirmar em ambos os lados que a conexão vai ser terminada.

Como podemos ver neste pequeno resumo, estas flags são usadas pelo TCP/IP para indicar o estado de uma comunicação feita. Por omissão, um scan TCP apenas usa a flag SYN porque é esta a que produz resultados mais fiáveis e também é a que menos dá nas vistas, porque é encarada como tráfego normal, igual a qualquer outro.

No entanto, podem ser usadas diversas combinações de flags que podem retornar resultados interessantes. Com o nmap, para usarmos uma combinação de flags basta fazer, por exemplo, isto:

```
C:\>nmap –scanflags SYNFIN nmap.org
```

Neste caso dizemos ao Nmap que faça um scan ao nmap.org mas usando para isso apenas as flags SYN e FIN.

Tipos de Scan TCP com Nmap

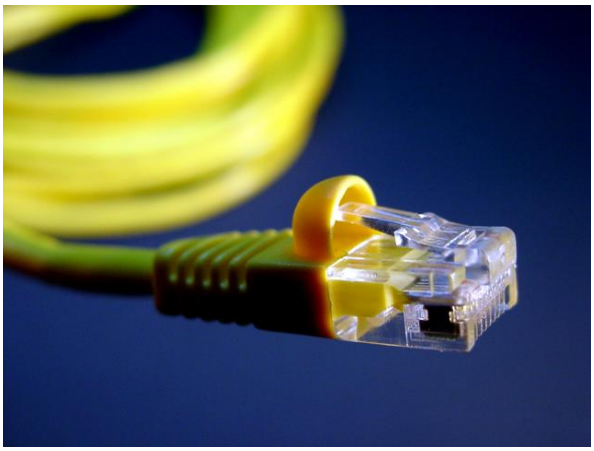
O Nmap permite a combinação arbitrária de flags mas algumas certamente apresentaram resultados mais úteis que outros. Por isso o Nmap prevê atalhos para os mais conhecidos. De seguida mostro alguns:

`SYN scan –sS`
É o scan feito por defeito pelo nmap.

`Connect scan –sT`
É similar ao feito com o –sS, mas neste caso a conexão é feita totalmente e depois desligada. É inferior ao –sS porque envolve mais pacotes e tem maior probabilidade de dar nas vistas.

`Windows scan –sW`
Este scan trabalha da seguinte forma, primeiro faz um scan com o ACK e depois verifica o tamanho da janela TCP enviada pelo host destinatário. Alguns sistemas operativos usam diversos tamanhos da janela dependendo se a porta esta aberta ou não.

`ACK scan –sA`
Este scan é particularmente útil para descobrir as regras de firewall de certos firewall's. Um host que receba este pacote deverá retornar um RST independentemente se a porta está aberta ou não. Se o RST é enviado o Nmap assume então que a porta não está a ser filtrada por um firewall ao passo que se não houver qualquer resposta da parte do alvo o Nmap deduz então que existe algures um firewall. Mas, claro que isto depende muito do tipo de firewall que esteja a proteger. Apesar disso sempre dá para ficarmos com uma ideia do que se passa.



UDP Scanning

Agora é que as coisas ficam um pouco mais complicadas. Fazer um scan usando o UDP é um pouco mais complicado devido à forma como o protocolo UDP funciona. Contrariamente ao TCP, que tem o cuidado de verificar o estado da ligação e definir até mesmo regras para a transferência de dados, o UDP não usa o handshakes e o primeiro pacote de dados enviado é logo enviado para a aplicação, não existe qualquer tipo de preocupação por parte do UDP em verificar se os dados foram ou não entregues. Mas mesmo assim este protocolo pode ser usado para fazer um scan, como vamos ver de seguida.

Tipos de scan UDP com o Nmap

Existem dois tipos de scan que podemos fazer usando o UDP: o scan com pacotes vazios e o scan com informação do protocolo. O primeiro scan, o dos pacotes vazios, envolve enviar pacotes UDP sem nenhum dado para uma porta e esperar o resultado. É um scan muito incerto, mas existe um scanner de nome Unicornscan que produz resultados mais fidedignos em virtude da maneira como trabalha. Para fazê-lo com o Nmap basta:

```
C:\>nmap –sU nmap.org
```

O scan usando dados de protocolo são mais sofisticados que envolve enviar dados de aplicações válidas em pacotes UDP para portas para ver se alguma aplicação responde. Usando esta técnica apenas são consideradas portas abertas se responderem a esses dados enviados com um nonerror. Mas este tipo não é muito aconselhável porque demora muito tempo, o que aumenta a possibilidade de ser detectado. Mas mesmo assim se querem fazer este tipo de scan podem fazê-lo com o nmap bastando para isso:

```
C:\>nmap –sU –sV nmap.org
```

Mas atenção que este scan demora muito tempo, por isso, se possível, limitem as portas a serem testadas.

Mapeamento de Rede

Agora que sabemos como cada protocolo se comporta ao ser feito um scan, podemos passar para outra parte que é o mapeamento da rede, descobrindo quais são os endereços de IP que tem um PC associado a ele.

Quando nos deparamos com um rede desconhecida, uma das primeiras coisas a ser feita é determinar qual o endereço IP que tem um PC associado e isto é importante porque grande parte das redes usa o chamado NAT (Native Address Translation) em que apenas uma pequena percentagem dos IPs são usados. Mas ao fazermos um scan host rapidamente podemos identificar quais os IPs com PCs associados a eles. Usando o Nmap podemos usar uma opção para fazer este scan host:

```
C:\>nmap –n –sP 12.125.10.1-15
```

A primeira opção que usamos é a –n que diz ao Nmap para não fazer lookups no endereço IP em que está a ser feito o scan isto porque um reverse DNS lookups demora mais tempo e por isso dá mais nas vistas. Na próxima opção a -sP o Nmap envia um ping assim como também um pacote SYN para a porta 80 para determinar se naquele endereço IP algum PC está a escutar. Acontece também que se estivermos na mesma subrede que o IP a que estamos a fazer o scan o ARP é usado para sabermos quais são os endereços IP em uso.

Como vimos, a opção –sP faz com que o Nmap envie um ping para o endereço IP para verificar se este tem ou não um PC associado, mas é bem provável que exista uma firewall pelo caminho e o que acontece é que estes pings são então bloqueados. Até mesmo a firewall que vem com o Windows XP faz isso.

É claro que existe maneira de dar a volta a esta situação, usando a opção –Po que diz ao Nmap para se conectar a cada porta apesar de parecer não existir um PC. Como se pode imaginar, é um processo bastante moroso. Mas sendo o Nmap uma ferramenta bastante completa, podemos usar outras opções para chegar quase ao mesmo resultado. Ao usarmos as opções –Psportlis (para os pacotes SYN) e PAportlist (para os pacotes ACK) bastará definir as portas que queremos que sejam usadas. Estas são algumas das opções que podemos usar, mas existem outras tais como a –PE, -PP e –PM em que se utiliza o ICMP para fazer scans.

É claro que é necessário fazer scans às portas certas e não perder tempo com outras. Quais é que são as mais comuns? E quais é que são as mais propícias a dar bons resultados?

Bem, depende do sistema operativo que está a ser usado pelo endereço de IP que queremos fazer um scan. Esses sistemas operativos requerem que determinadas portas estejam abertas de maneira a que o sistema possa, por exemplo, partilhar recursos. As portas mais usadas em Windows são TCP - 135, 139, 445, 1025-1030, 3389 e em UDP - 137, 138, 445 e 1025-1030. Em sistemas Unix são TCP - 21, 22, 23, 25, 80, 111 e em UDP - 53, 67-69, 111, 161 e 514.

Estas não são as únicas portas que podemos usar, existem muitas outras usadas tanto em sistemas Windows como em sistemas Unix. Por omissão, o Nmap verifica perto de 1700 portas TCP. Mas claro que o Nmap não é limitado só as estas 1700 portas. É possível usar as portas que bem entendermos bastando para isso usar a seguinte opção –p:

```
C:\>nmap –p 135-139,80,3389 12.125.10.1
```

É possível também usar uma outra opção do Nmap, a –F, que indica ao Nmap para fazer um scan rápido as todas as portas especificadas no ficheiro nmap-services que contém perto de 1200 portas. Outra possibilidade bastante interessante é podermos combinar no mesmo scan, um scan a portas TCP e outro a portas UDP. No caso de queremos fazer um scan às portas TCP 1025 a 1030 e nas portas UDP da 67 a 69 e da 111 a 120:

```
C:\>nmap –pT:1025-1030,U:67-69,111-120
12.125.10.1
```

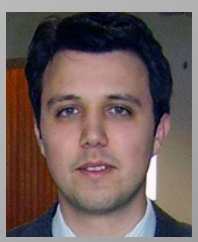
Reparem que usamos o T: para identificar as portas TCP e o U: para as portas UDP.

Para terminar, é bom salientar que com o Nmap podemos usar 3 maneiras para identificarmos os hosts pretendidos. Pode-se usar a opção de um único host, a opção em que se define uma escala de IP's e por fim uma não tão conhecida, a CIDR (Classless Inter-Domain Routing) em que basicamente se indica o endereço de IP e a subrede pretendida da seguinte maneira: 10.0.1.2/24.

Conclusão

Chegámos ao fim deste artigo onde foi demonstrado que usando um scanner como o Nmap, pode-se obter muita informação de uma rede. Com paciência e experiência é possível definirmos a cartografia da rede, os sistemas a correr em cada host e até mesmo saber quais são as regras de firewall e se existem falhas. Também vimos como os protocolos TCP e UDP são usados ao ser feito um scan.

SOBRE O AUTOR



Residente em Lisboa, Ciro Cardoso é um grande amante do vasto mundo que é a informática em especial a segurança. Está actualmente a trabalhar na próxima etapa do percurso Cisco com a certificação CCNP. Gosta bastante da área de redes, configuração de protocolos assim como a implementação.

Ciro Cardoso

Revista Linux

A Verdadeira Revista Portuguesa de Linux

Edição Bimestral em formato PDF

Download Gratuito

www.revista-linux.com

Python - Curso Completo

Este livro apresenta-se como uma escolha para quem quer conhecer todos os aspectos do Python, tanto no desenvolvimento web como no desenvolvimento para Desktop.

Apesar de ter sido lançado há alguns anos, o livro alcança, actualmente, a maior parte dos objectivos a que se propôs. Houve evoluções na linguagem que não estão reflectidas neste livro, como a existência de tipos booleanos.

Os primeiros 4 capítulos deste livro estão claramente direccionados para as pessoas que nunca tiveram contacto com linguagens procedimentais ou orientadas a objectos, pois faz da melhor forma possível a abordagem ao mundo da programação para um iniciante. No capítulo 5 nota-se que o livro já está algo ultrapassado, uma vez que no Python 2.1 ainda strings não eram consideradas objectos. A abordagem das bibliotecas gráficas de Python está bem actualizada e tem uma referência bastante completa da biblioteca Tkinter, no entanto peca por falar de outras bibliotecas multi-plataformas e não da wxPython, baseada no wxWidgets para C++. Em relação ao acesso a bases de dados, houve um claro desenvolvimento durante este tempo neste campo, e não há qualquer referência a uma das melhores e mais usadas bibliotecas do momento para acesso ao MySQL, MySQLdb. O capítulo que fala sobre manipulação de XML apenas peca pela passagem demasiado breve em relação à SAX (Simples API for XML). Um dos capítulos mais interessantes do livro é o que trata da programação em redes, onde se explica o uso de sockets e threads de uma forma extremamente simples, mas ao mesmo tempo de forma bastante completa. Infelizmente nem tudo são rosas, e o livro peca por não abordar com mais pormenor como se usar a smtplib. Um capítulo que prova que este livro se destina a todos é o "Extender o Python", onde se explica como criar extensões usando C, e o capítulo sobre ZOPE, onde se apresenta de forma completa esta framework.

Como conclusão pode-se dizer que este livro é destinado a quem quer dar os seus primeiros passos na programação ou na linguagem Python, não sendo considerado a melhor compra para quem quer aprofundar os seus conhecimentos de Python, onde os livros das editoras internacionais de renome como a O'Reilly reinam.

Nome: Python – Curso completo

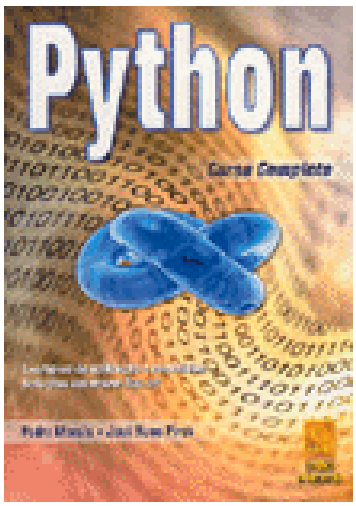
Colecção – Curso Completo

Editora – FCA

Autores – Pedro Morais / José Nuno Pires

ISBN 972-722-270-6

Abril de 2002



SOBRE O AUTOR

David Ferreira é um jovem apaixonado pelo mundo da informática. Deu os seus primeiros passos na programação aos 12 anos através da linguagem PHP, tendo hoje conhecimentos de XHTML, JavaScript, Python, redes, e outros. É um entusiasmado pelo mundo do hardware e software open-source, que o levou a explorar os sistemas GNU/Linux, com a distribuição Kurumin e depois Ubuntu.

David Ferreira

Techdays 2008

O maior evento tecnológico nacional esteve de volta ao Centro de Congressos de Lisboa, para mais uma edição de conhecimento, inovação e entretenimento, dedicada a profissionais de IT, programadores e académicos desta área. Este evento veio na continuidade dos anos anteriores e dos vários eventos realizados pela Microsoft nesta área. A cerimónia de abertura contou com a presença de Carlos Zorrinho, o coordenador do Plano Tecnológico, e teve o Alto Patrocínio do Presidente da República.

A edição 2008 do Techdays ficou marcada pelo enorme interesse e participação dos profissionais de IT e programadores nacionais ligados às tecnologias Microsoft. No final do período de registo antecipado, 70% dos lugares estavam já vendidos e a 1 semana do evento, atingiu-se um impressionante número de 2500 inscrições (2000 profissionais e 500 estudantes). A capacidade máxima foi atingida, tendo sido dadas mais de 150 sessões técnicas por 120 oradores, claramente a maior edição deste evento.

Durante os três dias foram demonstradas novas formas de utilizar, rentabilizar, gerir sistemas de informação e o que se pode fazer com toda a riqueza de abordagem e ferramentas de desenvolvimento aplicacional. Durante o Techdays, a Microsoft aproveitou também para apresentar a sua nova gama de servidores – Windows Server 2008, Microsoft SQL Server 2008 e Visual Studio 2008. Entre os vários objectivos apontados para o evento, a Microsoft identificou a criação junto dos seus programadores da nova figura do "Devigner" (resultante do cruzamento entre o designer e o developer) e do "Admilaxado" (um administrador de sistemas mais relaxado) para o qual a empresa acredita que os seus novos produtos vão contribuir decisivamente. O Techdays associou-se este ano à causa do ambiente lançando o slogan "For a world 2.0", possibilitando a todos os participantes a entrega de material informático obsoleto para posterior reciclagem, bem como efectuando a plantação de árvores junto com a Quercus para contrabalançar as emissões de CO_2 resultantes da energia consumida durante o evento.

As sessões deste ano focaram-se particularmente na Web 2.0, na mobilidade, no meio ambiente e na forma como as novas tecnologias podem ser utilizadas ao serviço desta temática e no desenvolvimento económico e social. Os vários Hands on Labs foram muito concorridos, tendo havido muitas pessoas que não tiveram oportunidade de realizar os vários laboratórios por se encontrarem cheios.

Devido à impossibilidade de assistir a todas as sessões, pois são realizadas várias em simultâneo, em seguida deixo uma breve descrição de algumas sessões a que assisti.

INT04 - Qual é o Contexto desta Conversação? Activando Conversações Longas em Serviços de Workflow. Serviços "Duráveis"

Orador: José António Silva, Microsoft

Tendo em consideração a natureza dos workflows onde uma instância de workflow pode estar activa durante um período de tempo longo ("long running instance"), é necessário desenvolver os serviços WCF de forma a suportar este tipo de cliente. Esta sessão abordou o tema de serviços "duráveis". Estes são um novo tipo de serviço da .NET Framework 3.5 que permite simplificar a persistência de estado numa "conversação" entre um serviço WCF e um cliente (ex: um workflow). O modelo de persistência do estado de um serviço WCF é em tudo idêntico ao de WF, sendo possível guardar o estado em BD, files system, etc . De forma a tornar um serviço WCF "durável" basta usar o prefixo "Durable" nos atributos da classe do serviço.Mais informação sobre "Durable Services" em http://weblogs.asp.net/gsusx/archive/2007/06/14/orcas-durable-services.aspx (post antigo mas com uma boa explicação) e em http://www.microsoft.com/uk/msdn/nuggets/nugget/270/Durable-Services-with-WCF-V35.aspx (screencast).

Algumas tools interessantes para WCF referidas durante a sessão:

- Configuration Editor Tool – aplicação que permite que permite a edição das configurações de serviços WCF com uma interface gráfica. Mais informação em http://msdn2.microsoft.com/en-us/library/ms732009.aspx.
- WCF Test Client – aplicação que permite efectuar testes "offline" sobre serviços WCF. Mais informação em http://msdn2.microsoft.com/en-us/library/bb552364.aspx.
- Service Trace Viewer Tool – aplicação que permite analisar logs de mensagens geradas pelo WCF. Mais informação em http://msdn2.microsoft.com/en-us/library/ms732023.aspx.

ARC01 - Software + Services: The Convergence of SaaS, SOA and Web 2.0

Orador: Beat Schwegler, Microsoft

Esta sessão retratou um tema muito em voga: o Software + Services. A sessão não teve qualquer demo, tendo tido uma componente bastante teórica, tendo sido iniciada com a

referência a três conceitos importantes: SaaS (http://msdn2.microsoft.com/en-us/architecture/aa699384.aspx), SOA (http://msdn2.microsoft.com/en-us/architecture/aa948857.aspx) e Web 2.0 (http://twopointouch.com/2006/08/17/10-definitions-of-web-20-and-their-shortcomings/).

Foram referidos alguns exemplos de modelos de negócio usados com S+S: Subscription/License Model, Advertisement Base Model (ex: Google) e exemplos de aplicações S+S: Eve Online, o Amazon S3 e a British Library. Por fim, foram ainda referidos alguns exemplos concretos de implementação do S+S pela Microsoft:

- Finished Services – Windows Live, Office Online
- Attached Services – XBOX Live
- Building Blocks – BizTalk Services

INT06: Viagem ao Centro da Nuvem – O Internet Service Bus (ISB) e os BizTalk Services

Orador: João Pedro Martins a.k.a "Jota", Create IT

A sessão começou de uma forma muito interessante com o Jota a "provocar" a audiência com algumas ideias sobre a forma como será o mundo das aplicações no futuro como a transição de um mundo com "data centers" nas próprias empresas para um em que o "hosting" é feito por grandes empresas com super "data centers" dedicados a fazer o "hosting" de milhares de aplicações. Foi uma forma interessante de cativar desde início a audiência. Neste sentido, foram dadas algumas estatísticas interessantes como a previsão do aumento de número de servidores de hosting da Microsoft de 200000 actuais para 800000 em 2011, indo de encontro à adopção do conceito de Software como um serviço (S+S) com as aplicações a ser alojadas em "hosting" externo e serem expostas como serviços. Os BizTalk Services, são basicamente a visão da Microsoft da forma como as aplicações irão comunicação entre si no futuro, facilitando o desenvolvimento de aplicações orientadas a serviços (SOA). A ideia fundamental dos BizTalk Services é a de permitir a comunicação segura entre as aplicações das organizações através de firewalls.

DEV06 - ADO.NET Entity Framework e LINQ To Entities

Orador: Luís Falcão (ISEL)

Nesta sessão foi abordada a ADO.NET Entity Framework (http://msdn2.microsoft.com/en-us/library/aa697427(VS.80).aspx), uma nova framework cujo objectivo é aumentar o nível de abstracção no que diz respeito à programação da camada de acesso a dados. Um dos problemas mais comuns no desenho das classes DAL é o esforço associado ao mapeamento entre as classes DAL e a BD. A Entity Framework facilita esta tarefa ao disponibilizar um diagrama integrado no Visual Studio 2008. Usando o Solution Explorer é possível importar a estrutura de uma base de dados para um diagrama de classes que é a transformação do modelo de dados num modelo de classes mapeado directamente com a estrutura da BD. Depois é possível definir novas relações e heranças entre as classes (não é possível definir herança ao nível da BD), mapear os dados de uma classe para que estes sejam divididos entre duas ou mais tabelas entre outras funcionalidades.

Na nova API vem incluído um novo .NET provider para Entity Framework (Entity Client) que é o correspondente ao SqlClient (para SQL Server) para actuar sobre as entidades criadas com a Entity Framework. Ainda se encontra em versão beta e nesta fase só permite gerar o Entity Model a partir da BD, não permitindo ainda criar primeiro o modelo antes e gerar a BD a partir deste.

Conclusão

Em suma, foi, sem dúvida nenhuma, um óptimo evento no qual se viu que a Microsoft procura recuperar da imagem negativa e também dar a conhecer o quanto poderoso é neste momento o .NET. Para mais informações sobre este evento visitem o site oficial http://www.techdays.pt/

SOBRE O AUTOR

Residente no Porto, João Brandão tem 9 anos de experiência em desenvolvimento de software e em montagem e elaboração de redes de computadores. Ocupa desde Dezembro de 2005 o cargo de Senior Engineer no Software Development Center da Qimonda AG. Tem como principais actividades e responsabilidades o desenvolvimento software e gestão de projectos numa grande variedade de aplicações na área web e não web.

João Brandão

Festival Nacional de Robótica 2008

Realizou-se entre os passados dias 2 a 6 de Abril a 8ª edição do Festival Nacional da Robótica na Universidade de Aveiro (provas sénior) e no Pavilhão Gimnodesportivo da Escola Básica João Afonso (provas júnior).

O Festival Nacional da Robótica teve ínicio em 2001 e tem como objectivo a promoção da Ciência e da Tecnologia junto dos jovens e do público em geral, através de competições de robôs. O Festival ocorre todos os anos em cidades distintas, incluindo ainda um Encontro Científico onde investigadores nacionais e estrangeiros da área da Robótica se reúnem para apresentar os mais recentes resultados da sua actividade. Este evento tem tido desde o seu início um enorme crescimento, quer em número de equipas e participantes, quer em termos de público.

Em relação à edição deste ano houve uma novidade em relação aos anos anteriores: a introdução das provas Micro-Rato e Ciber-Rato no calendário das provas, uma vez que este estas são provas realizadas todos os anos pela Universidade de Aveiro, sendo por isso este ano incluídas.

As provas Júnior são as que atraem mais participantes, tendo atraído 13 equipas para o Futebol Robótico 2x2, 103 para a Busca e Salvamento e 25 para a Dança, num total de 141 equipas. Já do lado das equipas Sénior houve 12 equipas inscritas para a prova de Condução Autónoma, 4 para o Futebol Robótico (INFAIMON Cup), 12 equipas para o Micro-Rato e 7 para o Ciber-Rato, totalizando 35 equipas.

A prova de Futebol Robótico Sénior é sempre a que atrai mais público, e este ano não fugiu à regra. O Pavilhão Aristides Hall (da Universidade de Aveiro), encheu para ver a final e alguns jogos. O facto de serem robôs grandes e jogar (e não os pequenos do Futebol Robótico 2x2) chama muito o público. Para além das provas acima mencionadas, estiveram também no Pavilhão Aristides Hall exposições do ISEP (Instituto Superior de Engenharia do Porto) com algumas demonstrações, destacando-se a apresentação do projecto FALCOS (Flight Autonomous Light Cooperative Observation System) com os seus dois "mini-aviões" autónomos, o ROAZ e o ROAZ II (qualquer coisa como dois barcos autónomos) e o seu "mini-submarino", operado também remotamente.
Já do outro lado da Avenida da Universidade, no Pavilhão Gimnodesportivo da Escola Básica João Afonso, o Futebol Robótico 2x2 foi o que chamou mais atenção do público, vibrando o público com cada golo marcado por cada um dos pequenos robôs.

Pela negativa há a destacar o facto das candidaturas ao programa Ciência Viva estarem encerradas e por isso, este ano, foi imposto um limite às equipas Júnior: apenas seriam permitidas 3 equipas por escola (para cada competições). Isto teve mais impacto nas equipas de Busca e Salvamento mas, de um modo geral, podemos dizer que a edição deste ano do Festival Nacional da Robótica foi um sucesso.

Haverá mais para o ano, só não se sabe ainda é a cidade.

SOBRE O AUTOR

Sendo participante na Revista PROGRAMAR desde a 4ª Edição, Joel Ramos é, actualmente, o coordenador-adjunto desta. Os seus primeiros passos na programação foram aos 13 anos com o PHP sendo ainda a sua linguagem predilecta. É também um aficcionado pela área da robótica, sobretudo desde que em 2007 participou no Festival Nacional da Robótica tendo repetido a participação em 2008.

Joel Ramos

</Xssed>

O Xssed é um portal onde podem ser encontradas informações relacionadas com falhas de cross-site scripting. Em particular, mostra como descobrir este tipo de falhas, como as corrigir, e como programar de forma a que estas não aconteçam. Tem também o maior arquivo da Internet ao nível deste tipo de vulnerabilidades, onde mostra quais os sites vulneráveis, por quem foram descobertas (reportadas) essas vulnerabilidades, e se já foram corrigidas, para além de detalhes sobre o erro em si.

www.xssed.com

ExtJS

A Ext JS é uma biblioteca de JavaScript cross-browser, ou seja, funciona correctamente nos browsers mais usados. Tem como objectivo auxiliar programadores de software web-based na criação das suas aplicações, para que possam proporcionar aos utilizadores uma navegação interactiva, simples, e descomplicada.

www.extjs.com

Google Doctype

O Google Doctype pretende ser uma enciclopédia assente numa wiki escrita por web developers para web developers. Abrange temas como a segurança (actualmente com especial ênfase a falhas de XSS), manipulação do DOM em JavaScript, dicas sobre CSS, entre outros. E o melhor disto tudo é que todos podemos contribuir.

http://code.google.com/doctype/

Queres participar na Revista PROGRAMAR? Queres integrar este projecto, escrever artigos e ajudar a tornar esta revista num marco da programação nacional?

Vai a

www.revista-programar.info

para mais informações em como participar,
ou então contacta-nos por

revistaprogramar
@portugal-a-programar.org

Precisamos do apoio de todos para tornar este projecto ainda maior...

contamos com a tua ajuda!

Equipa PROGRAMAR

Um projecto Portugal-a-Programar.org