Related technologies

Dennis Olsson

Tuesday 21 August

ASP – Language description

Active Server Pages was first distributed with IIS 3.0 in December 1996. The language (VBscript) is very similar to (Visual) Basic making it easy to read and understand, but also "isolates" the language from the main stream of languages such as C and all it's followers.

The language uses objects, like Java, but makes it much harder to write own objects and such to include. Variables are weakly typed.

Finally, each page is parsed once for every visited.

Basic syntax

All loops and other statements are terminated with an End-part, such as:

```
If name <> "foo" Then
  Response.Write("Your name is not foo!")
End If
```

As can be seen in the above example, the line is ended with a line break. Also, the code is case-insensitive, allowing the programmer to have his/hers own coding style.

Furthermore, comparison is made using "=" for "equals", "<>" for "not equals" and the keyword `not` as a prefix instead of `!` for negation.

Platforms

Since ASP was developed by Microsoft, the best support only exists on Windows platforms. However, there is a module called **Apache::ASP** for the web server Apache, allowing ASP to run on almost any platform.

Other implementations of ASP-parsing web servers has popped up, and died, throughout the years.

Re-using code

ASP has no easy way to let the user create it's own classes. However, it has good support to link in external **DLL**-files which can be compared in some sense with JAR files.

Therefore ASP may access data supplied by classes and such made in any other language that can create DLL files, for instance Visual Basic, which makes it easier to create DLL files that can communicate with other parts in the system.

This was widely used to create a backbone of DLL files that the ASP pages should communicate with, a structure hard to maintain and update.

Pros/Cons in short

The best support exists only on Windows platforms. The coding style differs a lot from most other languages. It's hard to structure the code for use in larger projects.

The pages **can** run on any platform with third party tools. Inclusion of external libraries is easy if you're an administrator that can register them.

ASP.NET – Language description

The language is a modern successor of ASP. It has error handling and better class creation support.

ASP.NET has a lot of interesting features such as memory leak-handling by reloading parts of the execution tree.

Opposed to classic ASP, each page is compiled on the first visit and then cached for upcoming visits until the file itself changes, just like JSP.

Since ASP.NET is a part of .NET it has access to its complete class library, giving a huge API to work with.

Platforms

ASP.NET is created by Microsoft and therefore only has support for Windows. However, the Mono-project aims to fully support .NET on the Linux, Solaris, Mac OS X, Windows, and Unix platforms. Mono also has a module for Apache enabling it to run ASP.NET-applications almost completely. Some issues still remain.

Pros/Cons

Faster than ASP Classic. More languages supported (ASP pages can be written in several languages).

The downside is the single platform-support by the creating company. Even if the mono project will fully support ASP.NET it's not the original. The users risk to be locked in with products from a single company.

PHP – Language description

PHP was created in -94 by a programmer to make his homepage more dynamic and to be able to collect information such as the number of visitors.

The recent version, PHP5, has a syntax similar to Java. It also has support for defining classes inside the source code for the page.

All variables are dynamically typed, meaning it can contain an integer, a string or an object. There is no way to restrict it.

Execution of code

The pages are, just as ASP, parsed each time they are visited. This can be accelerated using other software, but not natively.

There is also software that can optimize the code, making the execution faster. PHP source code can also be compiled, and then just simply executed when the page is loaded, but support for this is also from a third party.

Platforms

PHP may be runned on almost any platform. It can be runned with Apache, IIS, or almost any other web server.

Pros/Cons

PHP is widely used, and has a bad rumor created by many security issues, which of most are solved. It has a large, well documented, API, and the ability to define classes in the web-page code. Furthermore it's cross platform compatible.

One downside is it's API. Instead of having hundreds of classes performing their task on their type (for instance String.toLower), PHP has today 5177 loose functions. Also, they have no real naming convention.

When sites grow, the structure needed to maintain order within the code does not exist in PHP. Therefore sites does not scale well concerning load-balancing and code structure.

Development environments

There are numerous tools to help with development of dynamic web pages. Some support the creating of GUI, and others only support the code writing process by tab completion.

No matter what editor used, it should at least provide syntax highlighting.

JSP

There are several large environments for developing JSP-based sites.

For example, Eclipse supports, with the addition of plugins, the creation of both HTML and JSP. The Netbeans IDE can even integrate Dreamweaver MX and get support for graphical web page creation.

ASP.NET

Since ASP.NET is a part of .NET, it uses the same editor as other parts of .NET. By creating web pages in the GUI-builder, a programmer used to making regular programs can easily create an advanced, interactive, website in his/hers regular programming language.

This also makes it easy to migrate a .NET program to a web environment.

ASP/PHP

Both ASP and PHP lack a dedicated environment, but is on the other hand supported by, for instance, Dreamweaver for syntax highlighting.

Debugging is harder, and even though there are for example compilers for PHP, there are no general way to debug scripts, except by debug printing.

IDE cons

The problem with using any IDE is that with each piece of control given up, more control is lost.

Some problems include cross-browser support, accessibility issues and traffic amount problems.

Please note that different problems applies to what tools the programmer uses. An IDE can be used for syntax highlighting only, for complete code generation, or anywhere in between.

IDE cons

Cross-browser support is mainly graphical, since for instance Firefox and IE interprets different parts of the CSS standard in different ways. This will not be a problem if a descent IDE is used.

Accessibility issues is about the flow in the page as well as the design. The problem here is also AJAX, leaving blind people dead in the water, since visual changes aren't seen.

Traffic amount problems may happen when style sheets etc. is included inside the same document as the content. If this doubles the size of the file, it doubles the traffic load since regular style sheets would be cached by the client.

IDE cautions

Finally some things to take into notice: think ahead.

Even if the pages are used on an intranet, there may be handicapped employees in the future, the company may expand past the high-speed internet connections, and the company may have several systems on the workstations in the future.

Don't rely on the ease of changing things. Remember the year 2000.

IDE pros

Enough bashing.

A descent IDE will speed things up tremendously, especially if features such as GUI building etc. is used.

The previous issues may not even be issues when the pages are used in for instance for prototyping. Also, the code can be adjusted by hand, but make sure that the IDE does not erase the changes when the GUI is modified later on.

The code can also be generated at first, and then adjusted by professionals to confirm with standards withing the company.

IDE pros

The pros with for instance .NET is that very few people use anything else, including employees, so a person familiar with the IDE will be familiar with the environment at a new job.

The same program used for writing the code can also debug it, assist with code generation, warn for future issues, provide an overview of the source tree, assist with revision control.

IDE – Finally

Have a clue what happens beneath the surface of the tools you use in the IDE that affect the result. It will help you in fixing problems later on.

Remember: Each tool has it's purpose.

Database management system

A DBMS is a system allowing a user to save some data in a structured form.

The most common type is the Relational Database using the standard called SQL, which is a text-based language for communicating with a database.

Pronunciation

SQL should be pronounced each letter at a time, and preferably not ”sequel“ as some do, since **SEQUEL** (acronym for **Structured English Query Language**) which is another database query language which was used as a base for SQL.

Calling SQL SEQUEL would be like calling C++ C. Not entirely wrong, but not correct either.

Differences between DBMS

There are numerous aspects to have in mind when choosing a DBMS. However, performance does not have to be the main issue. Another server is cheaper than a lot of time spent on configuration, support, etc.

A major concern may instead be the feature-list.

Be aware of neat features that is unique to one product, since it's a one-way ticket to be locked in with one vendor.

Plan for changes

No matter which DBMS chosen, there will be issues when migrating to another.

PostgreSQL is said to be most complying with the SQL standard, and it's free. This could however open up for usage of keywords **only** supported by PostgreSQL.

Two important things a large system must support is that the DBMS can be runned on a separate server, and that it's possible to load balance between several servers in some way.

Always assume the DBMS will be changed and that there will be an infinite number of users.

Some equivalents and glossary

A basic set of differences and equivalents between **Oracle**, **MS SQL Server**, **MySQL** and **PostgreSQL** will be discussed. These are the four systems referred to as ”all DBMS“ below.

The expression ”all platforms“ translate to **Windows**, **Mac OS X**, **Linux**, **BSD** and **UNIX**.

Some equivalents and glossary

All DBMS supports **transactions**, which is a way of giving the server a large job, and letting it do everything at once. Either all or nothing succeeds. This can speed things up hundreds of times.

Unicode is also supported, and is the standard character set for Java. It allows for over a million separate characters in the language. The most generous estimate of possible need is estimated to about quarter-million characters.

Some equivalents and glossary

When selecting a subset of data from the database, the data must sometimes be handled and worked on in a new table before the result is returned. These tables are called **Temporary tables**, and are supported by all systems. Such a table is removed when the session is terminated.

There is also support for trigged events upon certain changed, joining results from several tables, some kind of own data types and functions.

Differences

The differences between these systems are not tremendous, and this list is far from complete.

Differences and problems in the area around ”PostgreSQL handle one single type of date/time-string wrong“ is not handled, but some larger issues are described below.

Oracle

The Oracle DBMS is suitable for a tremendous amount of data. It is expensive, but fast and works well with **large** data sets. It runs on all platforms except BSD.

It has support for ”Materialized view“, which is a way to have the same field in different tables, making changes to one of them global.

MS SQL Server

Microsoft software mix well with Windows environments, only. MSSQL also has support for "Materialized views", but have some limitations in load balancing. Data can't be spread using partitioning between several hosts "randomly" (hash-based).

It might be convenient having the same company as vendor for the whole system, since the products probably work better together than with third party programs. The downside of this is the dependence upon one single firm.

Also, MSSQL is widely used for semi-large databases and therefore consultants and expertise is widely available. Microsoft also provides a lot of help online.

MySQL

MySQL is free, as in Open Source. It however costs for commercial companies to use. On the other hand, it runs on "all" platforms, and is known to be easy to install.

Since it costs, the company can give support for the product, making the end user gaining access to expertise.

This DBMS does not support **Data domain**, a way to make custom data type out of the basic ones.

Also, there is no support for dividing the tables into sections. The standard specifies that tables are placed as:

database.**scheme**.**table**

MySQL lacks the support for schemes.

PostgreSQL

PostgreSQL is Open Source and completely free to use. It runs on all major platforms, but is often harder to install than MySQL.

The problem with PostgreSQL is when several users accesses the database. The owner of something is the creator. So if I'm allowed to write to a scheme, and create a table no-one else can access, the owner of the scheme can't remove it.

Also, PostgreSQL is known to be slow, especially with large data sets.

DBMS-Summary

As in most cases in real life, you get what you pay for.

PostgreSQL is stable and free, but is slow when it comes to large sets of data, and good support might be hard to find.

MySQL can be runned on all platforms, and support is provided by the vendor.

MSSQL is easy to administrate, and is literally made for the Windows environment. Support can be provided by the vendor and a lot of external companies.

Oracle suits tremendous amount of data, and is optimized to work fast.

Yin-Yang

The good part of complying with the standard results in a negative impact on performance, and vice versa. All DBMS tend to lock the product in to that particular database, and migrating may be hard, no matter what vendor is chosen. This applies mostly to the optimized keywords not supported by the SQL standard, but the SQL-specified keywords may differ as well.

If you want to stay free, use a set of commands available in several DBMS, or create accessors that may be replaced when the DBMS changes.