



Concepteur Développeur en Informatique

Développer des composants d'interface

Architecture Web – Présentation ASP Net

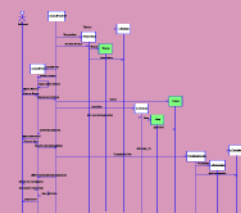
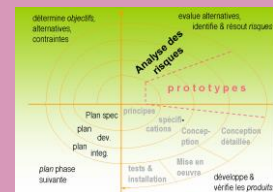
Accueil

Apprentissage

PAE

Evaluation

Impossible d'afficher l'image.



Localisation : U03-E05-S03

SOMMAIRE

1. Introduction	3
1.1. Les différentes architectures et technologies Web	3
1.1.1. Architecture client/serveur initiale	3
1.1.2. Programme s'appuyant sur CGI (Common Gateway Interface)	3
1.1.3. Programme s'appuyant sur une API propriétaire	4
1.1.4. Synoptique du traitement d'une requête.....	6
1.1.5. Une constante http	7
1.2. Architecture d'IIS 7.....	9
1.2.1. Modularité des fonctionnalités	9
1.2.2. Gestion modulaire des ressources	9
1.2.3. Extensibilité	11
1.2.4. Schéma de l'architecture en conclusion.....	11
2. ASP Net.....	12
2.1. L'infrastructure ASP Net.....	12
2.2. Les avantages d'ASP Net	13
2.2.1. Unification des langages	13
2.2.2. Rapidité et sécurité d'exécution :	13
2.2.3. Programmation objet	13
2.2.4. La sécurité.....	13
2.2.5. Homogénéité des développements	14
2.3. Composition d'une page ASP.Net.....	14
2.3.1. Eléments de présentation	14
2.3.2. Elément de traitement	15
2.3.3. Principes de génération de la page aspx	16

1. Introduction

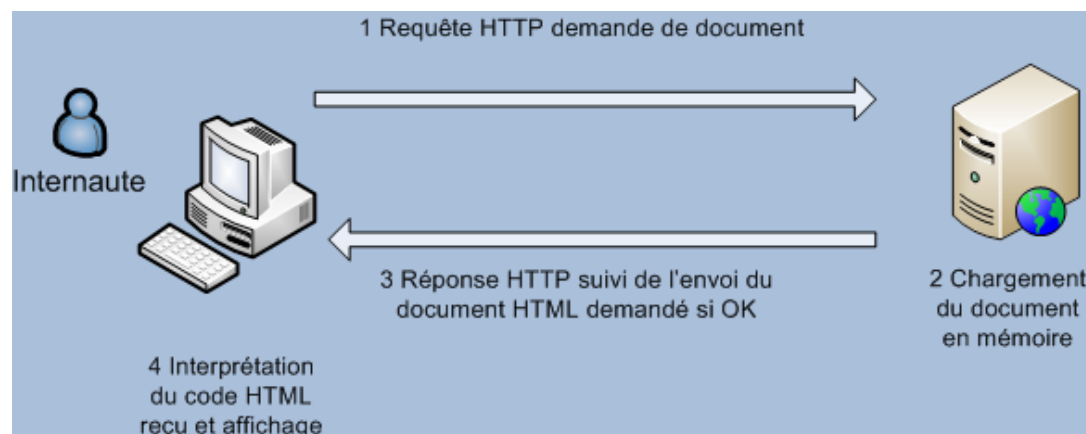
Ce document a pour objectif de vous présenter en bref les architectures et technologies mises en œuvre dans le cadre des développements d'application Web. Il y est fait état de quelques rappels historiques sur les technologies orientées développement Web côté serveur et quelques notions sur l'architecture des services proposés au sein du serveur web IIS 7.

1.1. Les différentes architectures et technologies Web

1.1.1. Architecture client/serveur initiale

Initialement, un serveur Web avait pour unique vocation de renvoyer à un client Web les documents dénommés selon l'adresse URL (Uniform Resource Locator) de la ressource. Un URL détermine le protocole pour accéder au fichier et le nom de celui-ci.

La structure générale d'une URL est : Protocole://Adresse/NomRep/NomDocument. Pour un serveur Web, le protocole retenu est HTTP (HyperText Transfer Protocol) ou sa version sécurisée pour laquelle les échanges seront cryptés à l'aide de SSL (Secure Socket Layout). Le principe de l'architecture initiale reste donc très simple et le protocole HTTP très rudimentaire.



Très rapidement ces fonctionnalités n'ont pas permis de répondre aux nouveaux besoins suscités par le développement des activités commerciales sur le Web.

Le serveur Web ne devait plus se limiter à la transmission de documents HTML mais devait permettre, au travers d'une interface et de scripts applicatifs, de préparer les documents dynamiquement et les envoyer toujours sous le même format et avec le même protocole.

1.1.2. Programme s'appuyant sur CGI (Common Gateway Interface)

CGI fût la première technique disponible. En plus des pages HTML statiques avec extension html, le serveur web héberge des programmes exécutables. Cette Common Gateway Interface permet de développer des applications sur le serveur capables d'interagir avec le client. Des options de configuration particulières indiquent au serveur que ces programmes sont à exécuter lorsque certaines URL sont demandées.

Sont alors développées des applications CGI et des scripts CGI.

L'application CGI est un programme compilé écrit le plus souvent en langage C, ou un programme précompilé en Java. Son rôle est de préparer le contenu des pages HTML à envoyer.

Les scripts CGI ont le même rôle mais sont codés dans des langages non compilés, traditionnellement en PERL.

Dans tous les cas, l'interface CGI lance toujours un exécutable (programme lui-même, ou interpréteur de script) qui crée sur le serveur un processus par requête (processus distincts du processus du serveur Web).

Cette technique a donc pour inconvénient de limiter le nombre de transactions simultanées et d'être extrêmement gourmande en ressources systèmes.

1.1.3. Programme s'appuyant sur une API propriétaire

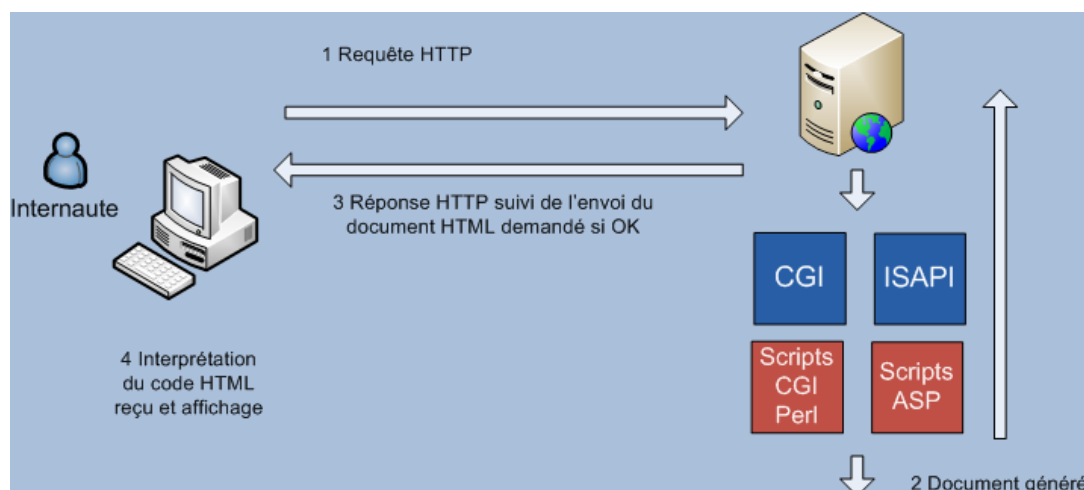
Les API propriétaires (c'est-à-dire des interfaces natives des serveurs Web) sont apparues en 1996 dans le but d'améliorer les performances par rapport à CGI.

Un programme écrit avec l'API est exécuté dans le contexte de la tâche du serveur HTTP et non comme un processus indépendant (la DLL de l'API n'est chargée qu'une fois).

Microsoft a développé son interface, ISAPI, acronyme d'Internet Server Application Programming Interface. Cette interface possède les inconvénients de ces avantages : s'exécutant dans le processus du serveur, elle peut compromettre la stabilité de celui-ci.

Ces API existent sous deux formes, les filtres et les extensions, toujours livrées sous formes de DLL multithreads. Elles sont résidentes en mémoire après l'appel et partageables entre plusieurs clients simultanément. Les pages ASP et ASP Net sont traitées par des filtres ISAPI.

Le schéma suivant figure cette architecture :



Les pages ASP Net aussi jusqu'à la version de IIS 7.

Pour le serveur Web de Microsoft (IIS), un programme ISAPI (appelé filtre ISAPI) s'appuie sur une DLL multi-threads, restant en mémoire après l'appel, et qui est réentrante (partageable entre plusieurs clients simultanément). Les pages ASP et ASP.Net sont traitées par des filtres ISAPI.

Les scripts serveur s'exécutent sur le serveur Web avant envoi de la page au client contrairement aux scripts clients qui s'exécutent sur le client, une fois la page chargée.

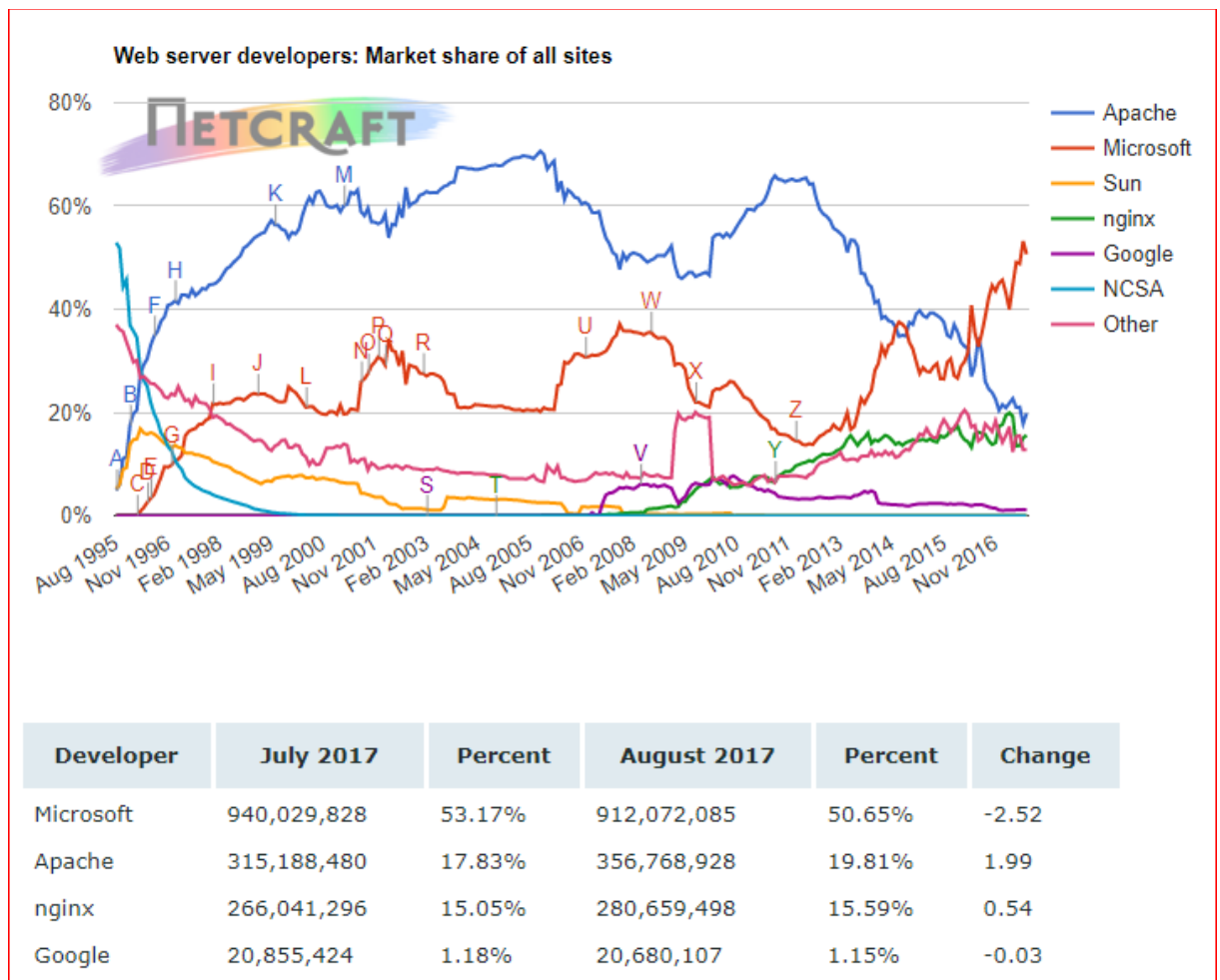
Les scripts serveur peuvent générer tout type de sortie, non seulement des valeurs pour les variables ou les expressions, mais aussi du texte et des balises html ainsi que des scripts client.

Une même page de script serveur peut générer différentes pages HML en fonctions des valeurs récupérées.

Des scripts client et serveur peuvent cohabiter sur une même page Web.

1.1.4. Le marché des serveurs Web

Part de marché des serveurs web (source NetCraft)



Nginx, solution open source, connaît un fort taux de croissance.

Google pour sa solution GWS (Google Web Server) qui repose sur une solution Apache modifiée

Mais aussi des chiffres bien différents, tels ceux de W3Techs

Web Servers

Most popular web servers

	usage	change since 1 August 2017
1. Apache	49.0%	-0.2%
2. Nginx	34.8%	+0.3%
3. Microsoft-IIS	10.9%	-0.2%
4. LiteSpeed	2.9%	+0.1%
5. Google Servers	1.1%	

percentages of sites

1.1.5. Synoptique du traitement d'une requête ciblant un programme

Quelle que soit la technologie retenue, une transaction sur le Web peut se résumer aux étapes suivantes.

- ❶ Le client charge la page
Le navigateur interprète le code html. Les scripts (Javascript) inclus dans la page sont interprétés par le moteur de script du navigateur.
Des plugins peuvent être chargés et installés pour exécuter des applets Java, des composants tels que Silverlight (RIA Microsoft) ou des composants Flash. Depuis l'événement de HTML 5, les technologies nécessitant des plugins sont peu à peu abandonnées.
- ❷ Le client envoie une requête au serveur Web (en HTTP) en désignant le programme serveur par son URL.
- ❸ Le moteur HTTP lance le programme ou passe le relai au serveur d'application.
L'architecture peut en effet comporter un serveur http en charge de la réception des requêtes et la transmission des réponses et un serveur d'application qui va préparer les documents.
- ❹ Le programme s'exécute sur le serveur
Traitement des informations saisies par l'utilisateur dans le formulaire de la page Web.
Interrogation d'une base de données et transmission des résultats à une page html.
Redirection d'une demande utilisateur vers une page particulière en fonction d'une condition comme la vérification d'un mot de passe par exemple.
Si l'application exige certains traitements intermédiaires (authentification, suivi du cheminement,...), le serveur d'application (entre le serveur Web et le serveur de données) prend en charge la gestion du contexte utilisateur.
- ❺ Le programme serveur transforme les résultats en page html qui est renvoyée au moteur HTTP.
- ❻ Le moteur HTTP renvoie la page html au client, avec les fichiers associés : multimédias, feuilles de styles, scripts, composants clients (applets, ActiveX), ...

Dans le contexte de la formation, serveur Web et serveur d'application seront confondus.

1.1.6. Une constante http

Les limites du Web sont celles aussi du protocole http.

Dans une application Web vous utilisez un protocole de communication entre le serveur et le client dit sans état.

La durée de connexion au serveur est celle nécessaire pour obtenir la ressource demandée mais ce canal de communication n'est pas maintenu au-delà du temps nécessaire à l'échange.

Une fois que le serveur a terminé la transmission de la réponse sous la forme d'un flux HTML le lien de communication est brisé.

Il s'agit d'un protocole très simple. 3 méthodes sont particulièrement utilisées :

HTTP méthode	Description ou quand l'utiliser
Head	Cette méthode permet d'obtenir des informations d'entête sur le navigateur utilisé, la page d'où l'on vient, le système d'exploitation utilisé.
Get	Permet d'obtenir une ressource Permet d'envoyer des données (en faible volume) dans l'adresse url
Post	Permet d'envoyer des données dans le corps de la requête. C'est la méthode privilégiée pour poster les données.

A la suite d'une requête, vous obtenez une réponse qui précise dans l'entête de la réponse l'état du traitement de la demande (status). Ces codes de statut sont codifiés selon les règles suivantes.

Codes de statuts (groupe)	Description
1XX	Requête reçue, processus en cours.
2XX	Succès : l'action a bien été reçue, comprise et acceptée.
4XX	Erreur client : La requête comporte une erreur de syntaxe ou le serveur n'arrive pas à satisfaire la requête.
5XX	Erreur serveur : Le serveur n'arrive pas à satisfaire une requête qui semble être valide.

Quelques exemples. Vous trouverez sur internet ou sous IIS l'ensemble des codes d'état.

Codes Statut	Raison
100	Continue
200	Ok
201	Créé
400	Mauvaise requête
401	Non autorisé
403	Interdit
404	Non trouvé
408	Délai de requête dépassé
501	Non implémenté

Nous verrons au cours des développements futurs l'ensemble des propriétés disponibles au niveau des entêtes http.

Le tableau ci-dessous vous présente quelques-unes de ces propriétés :

Propriété	Description
Statut	Première ligne Version http et le statut de la réponse Exemple http 1.1 200 OK
Date	Date heure de la génération du message
Server	Modèle de serveur http répondant à la requête
Content-Length	Taille en octets du contenu qui suit
Content-Type	Le type Mime de la ressource
Expires	Date heure à laquelle la ressource doit être considérée comme obsolète (utilisée pour la mise en cache sur le navigateur)
Accept	Liste des types Mime acceptés par le client. Le caractère * indique que tous les types sont acceptés.
Accept-charset	Précise les encodages de caractères acceptés

1.2. Architecture d'IIS 7

IIS7 la version d'Internet Information Services proposé dans Windows 7 et Windows Server 2008 se base sur une architecture totalement nouvelle en rupture avec les anciennes versions d'IIS.

L'architecture d'IIS7 repose sur deux principes essentiels qui font son intérêt :

- La modularité
- L'extensibilité

1.2.1. Modularité des fonctionnalités

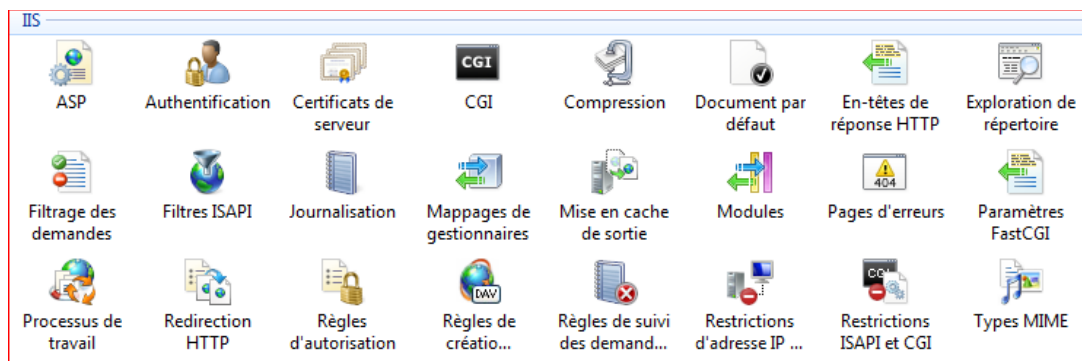
La modularité va permettre de configurer ces serveurs WEB en fonction des services qui doivent être proposés. Auparavant les services étaient proposés sous la forme d'un bloc monolithique, une seule application. Cette contrainte imposait de monter tous les services et exigeaient des ressources importantes pour des fonctionnalités parfois basiques.

Aujourd'hui, sur cette nouvelle version, les services sont proposés sous forme de modules activables ou non.

Ainsi si vous ne souhaitez pas utiliser le module de mise en cache ou le module de log proposés par Microsoft, vous êtes tout à fait libre de supprimer l'utilisation de ces modules pour les sites web que vous souhaitez.

Il vous est donc possible de personnaliser complètement le comportement du serveur web.

L'image suivante montre les différents services d'IIS fournis sous la forme de modules applicatifs.



Il est possible de les arrêter s'il n'est pas nécessaire de fournir la fonctionnalité couverte.


1.2.2. Gestion modulaire des ressources

Il est aussi possible d'isoler le processus d'exécution de votre application Web dans un processus distinct de celui dans lequel s'exécute votre serveur Web en définissant un pool d'application particulier.

Les pools d'application définissent les paramètres de configuration et les frontières des applications.





Un pool d'application est composé d'une ou plusieurs applications qui partageraient un contexte commun d'exécution.

Vous pouvez allouer des ressources spécifiques à chaque pool comme l'utilisation max de CPU (unités de traitement du processeur), un contexte commun d'exécution comme la version d'ASP Net, gérer le recyclage des ressources.

 **Pools d'applications**

Cette page permet de consulter et de gérer la liste des pools d'applications sur le serveur. Les pools d'application sont comportent une ou plusieurs applications et permettent d'isoler les différentes applications.

Filtrer : Atteindre Regrouper par : Aucun regroupement

Nom	État	Version du ...	Mode pipeline ...	Identité	Applications
 ASP.NET v4.0	Démarré	v4.0	Intégré	ApplicationPoolId...	0
 ASP.NET v4.0 Classic	Démarré	v4.0	Classique	ApplicationPoolId...	0
 Classic .NET AppPool	Démarré	v2.0	Classique	ApplicationPoolId...	0
 DefaultAppPool	Démarré	v2.0	Intégré	ApplicationPoolId...	2

Les applications sont par défaut démarrées dans DefaultAppPool.

Vous pouvez créer des pools particuliers et restreindre les ressources mobilisables au sein de ce pool comme le figure l'image ci-dessous. Toutefois, en tant que développeur, nous n'aurons pas à définir ces propriétés. Laissons cette tâche aux administrateurs du système.

Paramètres avancés

Nombre maximum de pannes 5

Paramètres de l'exécutable de ferr

Type de réponse "Service indispo HttpLevel

☒ **Recyclage**

Désactiver le recyclage avec chevi False

Désactiver le recyclage pour les m False

☒ Générer une entrée de journal poi

☒ Heures spécifiques **Tableau de TimeSpan[]**

Intervalle de temps régulier (minu 1740

Limite de la mémoire privée (Ko) 0

Limite de la mémoire virtuelle (Kc 0

Nombre limite de demandes 0

☒ **UC**

Affinité du processeur activée False

Intervalle limite (minutes) 5

Limite 0

Limiter l'action NoAction

Masque d'affinité du processeur 4294967295

Délai d'inactivité (minutes)

[idleTimeout] Durée (en minutes) pendant laquelle un processus de travail reste inactif avant son arrêt. Un processus de travail est inactif s'il ne traite pas de demandes et qu'il ne reçoit aucune nouvelle demande.

OK Annuler

1.2.3. Extensibilité

IIS7 est aussi susceptible de voir ses fonctionnalités étendues par le développement de modules propres. Il est ainsi totalement envisageable de proposer de nouvelles fonctionnalités et de personnaliser le comportement du serveur.

Auparavant cette possibilité existait mais il était nécessaire de l'implémenter sous forme de filtres ISAPI qui devaient être programmés dans le langage natif le C++.

Aujourd'hui les modules peuvent être développés dans n'importe quel langage disponible dans l'architecture Dot Net (C#, VB, ...).

Ainsi, si nous ne souhaitons pas implémenter l'authentification des utilisateurs telle que proposée par IIS, nous pouvons développer notre propre module afin qu'il réponde de manière plus pertinente à nos souhaits.

Nous ne travaillerons pas sur la conception et le développement de modules pour IIS car cela constitue des approches réservées à des spécialistes de l'implémentation d'applications Web. Nous nous contenterons d'utiliser les modules fournis.

1.2.4. Schéma de l'architecture en conclusion

Le schéma suivant présente le traitement complet d'une requête dans le contexte http de celle-ci.



❶ La première phase de traitement d'une demande reçue par le serveur est la création du contexte HTTP à partir de la classe HttpContext.

Le contexte http permet d'accéder à la demande (requête), la réponse, les en-têtes, et les cookies présents dans le flux http.

❷ Ensuite les différents modules peuvent être instanciés et leur méthode d'initialisation Init invoquée. Le traitement doit commencer par l'authentification

(un des modules en fonction des choix retenus) et la vérification des autorisations.

Ensuite, les modules invoqués sont ceux qui sont configurés.

❸ Ensuite c'est dans la phase ExecuteHandler que seront effectués les traitements proprement dits de la requête. L'API adaptée en fonction de la nature de la requête est alors sollicitée.

❹ La réponse est transmise avec éventuellement appel à des modules de compression (zip) et d'historisation (log).

2. ASP Net

2.1. L'infrastructure ASP Net

ASP.NET est la technique que nous utiliserons pour le développement et l'exécution d'applications sur un serveur Web et en natif, sur le serveur Web IIS.

ASP.NET fait lui-même partie du .NET Framework.

En travaillant avec ASP Net vous avez donc accès à l'ensemble de l'infrastructure de Dot Net (sauf bien entendu aux frameworks spécialisés destinés à d'autres architectures comme Windows Forms).

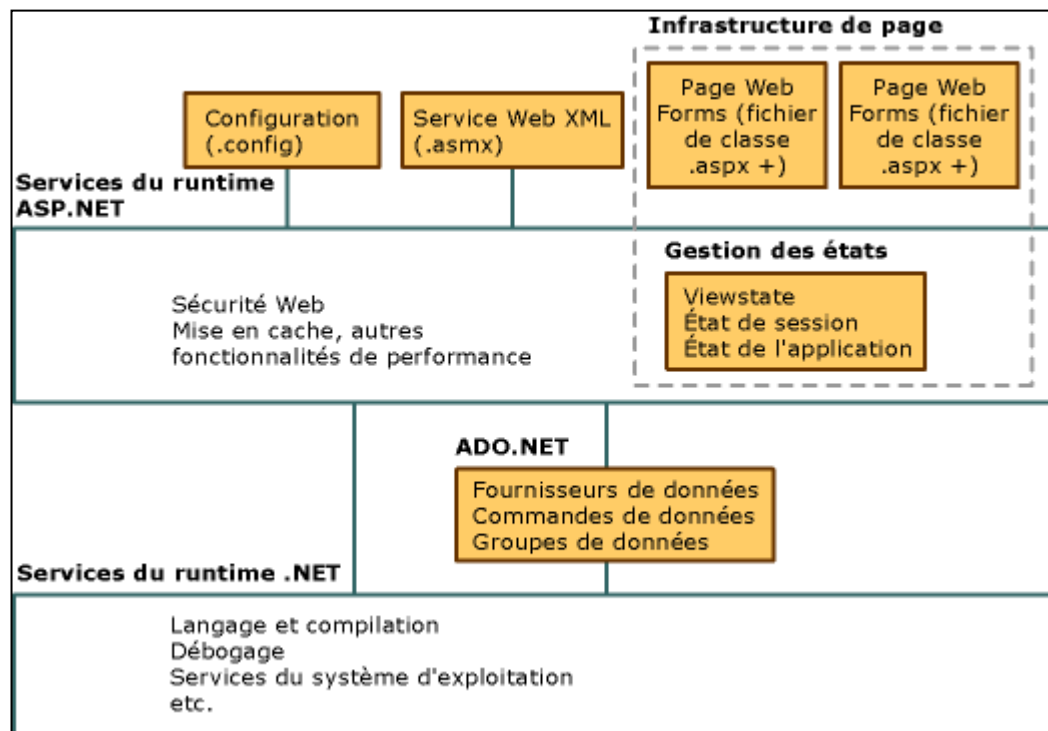
Vous pouvez, par exemple, créer des applications Web ASP.NET à l'aide de n'importe quel langage de programmation .NET (Visual Basic, C#, les extensions managées de C++ et tout langage respectant la spécification de Dot Net).

Vous bénéficiez des fonctionnalités de débogage .NET.

Vous accédez aux données via ADO.NET ou autres frameworks comme Linq To Entities qui sera vu ultérieurement.

De la même façon, vous pouvez accéder aux services de système d'exploitation à l'aide des classes .NET Framework, etc.

Vous voyez dans le schéma ci-dessous les principaux objets fournis par les services du Runtime ASP Net.



2.2. Les avantages d'ASP Net

2.2.1. Unification des langages

Vous pouvez pour coder vos pages utiliser le langage de votre choix.

Comme pour tout autre composant Dot Net, le code source de chaque langage est compilé par le CLR en langage intermédiaire (MSIL, Microsoft Intermediate Language) grâce à la spécification commune (CLS, Common Language Specification) , puis lié avec des méta données pour créer un exécutable (ou une dll) appelé assembly.

Le CLR joue ensuite le rôle de machine virtuelle qui compile en code natif (code machine optimisé pour le processeur) et exécute le fichier binaire MSIL.

2.2.2. Rapidité et sécurité d'exécution :

Au moment de l'exécution de la page, il y a traduction de ce langage intermédiaire par le compilateur JIT (Just In Time) du CLR en code natif.

Exemple: une page est compilée lorsqu'elle doit s'afficher, mais les méthodes de traitement des événements de ses contrôles ne sont compilées qu'au moment où ces événements surviennent.

Ces compilations n'ont lieu qu'une seule fois et restent en mémoire pour les prochaines invocations de la page.

Cependant elles peuvent être supprimées en cas de modification du code de la page ou en cas de besoin de ressources mémoire.

En cours d'exécution de la page, le CLR surveille (débordements mémoire, ...) et recycle (garbage collector) l'espace mémoire.

Vous bénéficiez donc dans le développement Web de l'ensemble des fonctionnalités disponibles au niveau du framework Dot Net.

2.2.3. Programmation objet

La programmation en ASP.Net repose sur un ensemble de classes.

Comme pour toute autre application Dot Net, il vous suffit de lier la librairie que vous souhaitez incorporer et d'importer l'espace de noms adéquat pour utiliser une classe dans une page Asp.Net.

2.2.4. La sécurité

La sécurité est renforcée par le biais de services d'authentification (Authentication Provider) qui assurent les contrôles grâce à des listes ACL de Windows, des bases de données d'utilisateurs ou des fichiers XML.

Il existe donc trois modes d'authentification (Window, Passport, Cookie).

La sécurité est également renforcée par le biais du contrôle d'accès aux ressources (contrôle au niveau fichier si authentification Windows, contrôle au niveau URL par un fichier de configuration – web.config).

Enfin ces mécanismes d'authentification et de contrôle d'accès peuvent être renvoyés au serveur IIS par un mode dit « Impersonation ».

Le suivi de session sous ASP.NET est mieux sécurisé que par le passé et ne nécessite plus forcément l'utilisation d'un cookie dans le navigateur du client (voir la propriété cookieless du fichier de configuration web.config). Dans ce dernier cas un ID unique est alors embarqué dans l'URL.

Les données de session d'un utilisateur peuvent être conservées sur le serveur Web mais aussi dans un processus séparé ou dans un serveur SGBD. Ces mécanismes sont plus lents du fait des communications interprocessus nécessaires mais peuvent faciliter la reprise des données en cas de défaillance de l'application ou du service Web.

2.2.5. Homogénéité des développements

La séparation est claire entre les différentes couches de l'application. Vous pouvez toutefois, si vous programmez sans vous fixer certaines règles, violer ce principe et faire alors, comme par le passé, du code spaghetti.

2.3. Composition d'une page ASP.Net

Une page Asp.Net est constituée de plusieurs éléments dont je vous propose un aperçu rapide dans cette introduction à la technologie Asp.Net : Les éléments de présentation et les éléments de traitement.

2.3.1. Eléments de présentation

La couche de présentation est fournie sous la forme d'une page HTML comme vous pouvez le constater ci-dessous. Certaines balises préfixées par « asp: » vous sont aujourd'hui inconnues mais plus pour longtemps.... Ici le document default.aspx et le document default.aspx.designer qui contient les contrôles serveur générés en mode conception.

```
<%@ Page Title="Page d'accueil" Language="C#" MasterPageFile="~/Site.master"
AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="SupportCours._Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <h2>
        Bienvenue dans ASP.NET!
    </h2>
    <p>
        Pour en savoir plus sur ASP.NET, consultez
        <a href="http://www.asp.net" title="Site Web ASP.NET">www.asp.net</a>.
    </p>
    <p>
        Vous pouvez également trouver de la
        <a href="http://go.microsoft.com/fwlink/?LinkID=152368"
        title="Documentation ASP.NET sur MSDN">documentation sur ASP.NET sur MSDN</a>.</p>
    <p>
        <asp:Button ID="btnEntree" runat="server" onclick="btnEntree_Click"
            Text="Entrez sur le site" Width="168px" />
    </p>
</asp:Content>
```

```
namespace SupportCours {

    public partial class _Default {

        /// <summary>
        /// Contrôle btnEntree.
        /// </summary>
        /// <remarks>
        /// Champ généré automatiquement.
        /// Pour modifier, déplacez la déclaration de champ du fichier
        /// de concepteur dans le fichier code-behind.
        /// </remarks>
        protected global::System.Web.UI.WebControls.Button btnEntree;

    }

}
```

2.3.2. Élément de traitement

Et un fichier qui contient le code, ici en C#, destiné à contrôler les flux, réaliser les différentes opérations de l'application, qu'elles s'appliquent aux objets métiers ou aux objets de persistance. Ici le document default.aspx.cs

```

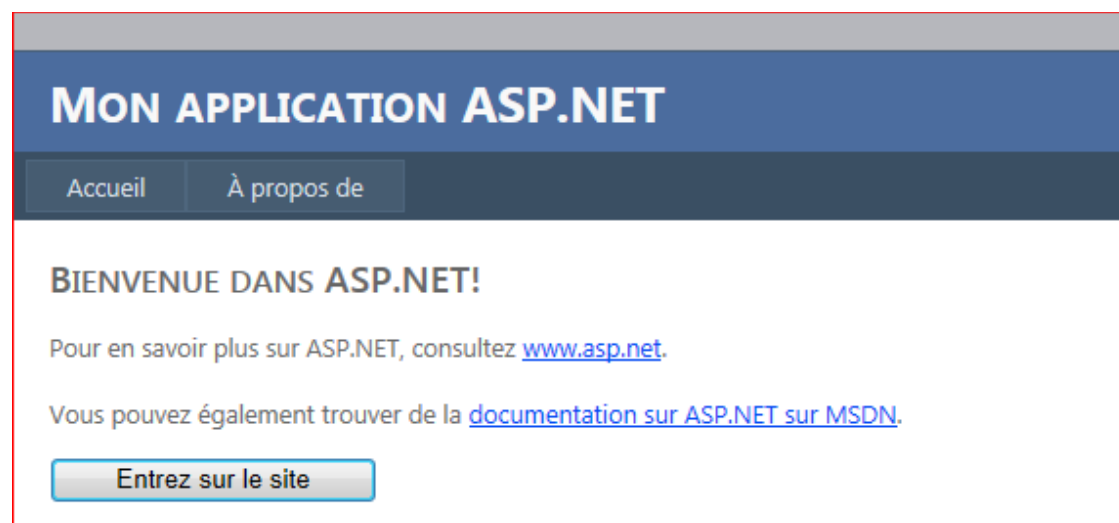
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Web.UI;
6 using System.Web.UI.WebControls;
7
8 namespace SupportCours
9 {
10     public partial class _Default : System.Web.UI.Page
11     {
12         protected void Page_Load(object sender, EventArgs e)
13         {
14
15         }
16
17         protected void btnEntree_Click(object sender, EventArgs e)
18         {
19             this.btnEntree.ForeColor = System.Drawing.Color.Red;
20         }
21     }
22 }
23

```

Vous remarquerez que cette page hérite d'un type **Page** de l'espace de nom System.Web.UI. Elle est déclarée comme partielle.

Nous retrouvons des gestionnaires d'événement associés aux événements Load de la page et Click du bouton.

C'est en fait la compilation de 3 documents default.aspx, default.aspx.designer, default.aspx.cs qui donnera la page dans sa version finale au format HTML.



Nous pouvons consulter le source généré par curiosité et constatons que le contrôle serveur asp :button a donné naissance à un contrôle HTML de type input submit avec une génération automatique d'une propriété name et d'un id.

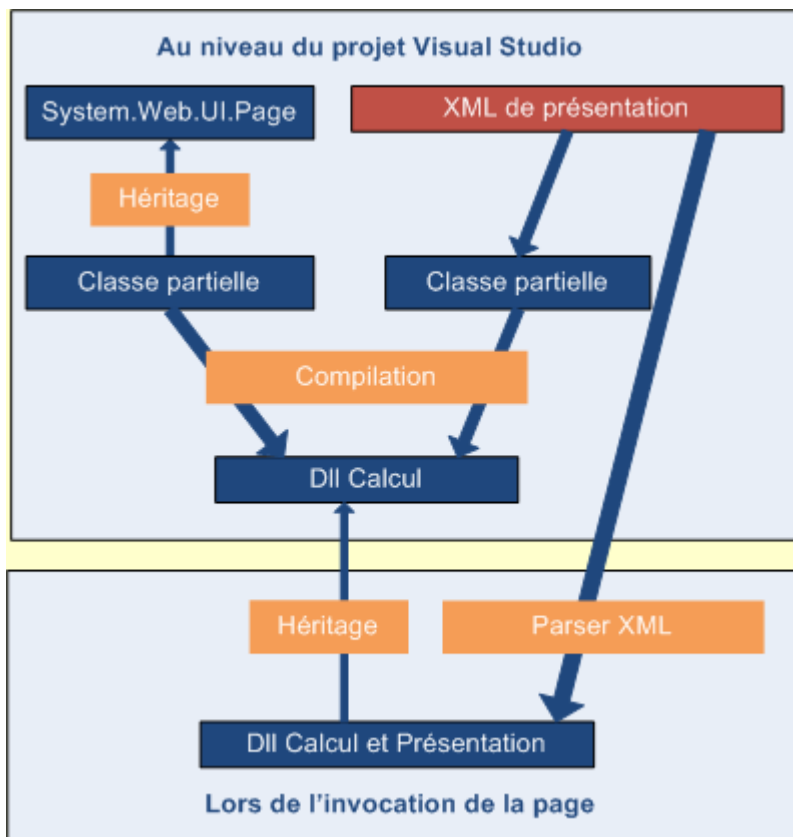
```
<div class="main">

  <h2>
    Bienvenue dans ASP.NET!
  </h2>
  <p>
    Pour en savoir plus sur ASP.NET, consultez
    <a href="http://www.asp.net" title="Site Web ASP.NET">www.asp.net</a>.
  </p>
  <p>
    Vous pouvez également trouver de la
    <a href="http://go.microsoft.com/fwlink/?LinkID=152368"
    title="Documentation ASP.NET sur MSDN">documentation sur ASP.NET sur MSDN</a>.</p>
  <p>
    <input type="submit" name="ctl00$MainContent$btnEntree" value="Entrez sur le site"
    id="MainContent_btnEntree" style="width:168px;" />
  </p>

</div>
```

2.3.3. Principes de génération de la page aspx

Voici de manière synthétique comment se déroule le processus de génération d'une page asp.net. Lors de la première demande d'une page, il y a compilation des différents éléments pour former un assembly qui sera chargé par IIS.



1. Demande par le client de la ressource via son URL
2. Lors de la première demande, la classe dérivée de Page prend en compte la présentation définie par le fichier .aspx. Recours à un PARSEUR XML.
3. Lors de cette première demande, la classe est compilée en assembly
4. L'assembly est chargé par IIS et invoqué pour générer la page HTML
5. IIS transmet la page au client