.Net et les pages dynamiques ASP.Net

Michel RIVEILL
riveill@unice.fr - http://www.essi.fr/~riveill

Laboratoire I3S
Ecole d'Ingénieur en Sciences Informatiques (ESSI)

Plan

- IIS
- Applications Web
- Configuration
- Trace
- Session
- Cache
- Gestion des erreurs
- Déploiement
- Disponibilité
- Sécurité
- Coté serveur
- Coté client
- Mise en œuvre des contrôles
- Développer des formulaires

IIS

- Internet Information Server
  - Supporte ASP.NET
  - S'exécute dans le processus `inetinfo.exe`
    - comme FTP, NNTP, SMTP
    - Avec lesquels il partage des ressources
  - Localisation par défaut
    - `c:\inetpub\wwwroot`
- Répertoire virtuel
  - Fourni un niveau d'indirection entre l'URL et la localisation des fichiers
  - Par exemple :
    - `URL = http://myServer/myApplication/foo.asp`
    - Fichier = `d:\myFolder\myAppFolder\foo.asp`

Plan

- IIS
- Applications Web
- Configuration
- Trace
- Session
- Cache
- Gestion des erreurs
- Déploiement
- Disponibilité
- Sécurité
- Coté serveur
- Coté client
- Mise en œuvre des contrôles
- Développer des formulaires

Applications Web

- Quesaco ?
  - Toutes les ressources (fichiers, pages, handlers, modules, code exécutable, etc.) associées à un répertoire et ses descendants
    - Fichier de configuration
    - Données partagées (variables de l'application)
    - `global.asax`
  - Etendre les sessions utilisateurs
    - Une session est constituée d'un ensemble de pages web accédée par un même utilisateur, dans la même unité de temps
    - Utilisation de données partagées (variables de session)

Applications Web
`global.asax`

- Localisé à la racine de l'application
- Peut définir et initialiser des variables de session et applicatives
  - Objet précis créé par une classe, COM ProgID ou COM ClassID
  - Attention : l'utilisation d'objets partagés peut être une cause d'inter blocage ou d'erreur (gestion de la concurrence)

```
<object id="items" runat="server“
  scope=“application”
  class="System.Collections.ArrayList" />
```

Applications Web
`global.asax`

- Peut contenir du code utilisateur qui capture les événements de la session ou de l'application (comme ASP)
  - `Application_OnStart`, `Application_OnEnd`
  - `Session_OnStart`, `Session_OnEnd`

```
void Application_OnStart() {
  Application["startTime"]=DateTime.Now.ToString();
}
void Session_OnStart() {
  Session["startTime"]=DateTime.Now.ToString();
}
```

Applications Web
`global.asax`

- Peut utiliser du code présent dans les pages
- Peut contenir des directives applicatives
  - `<%@ Application Description="This is my app…" %>`
  - `<%@ Application Inherits="MyBaseClass" %>`
  - `<%@ Application Src="Global.cs" %>`
    - Utilisé par Visual Studio.NET
  - `<%@ Import Namespace="System.Collections" %>`
  - `<%@ Assembly Name="MyAssembly.dll" %>`

Plan

- IIS
- Applications Web
- Configuration
- Trace
- Session
- Cache
- Gestion des erreurs
- Déploiement
- Disponibilité
- Sécurité
- Coté serveur
- Coté client
- Mise en œuvre des contrôles
- Développer des formulaires

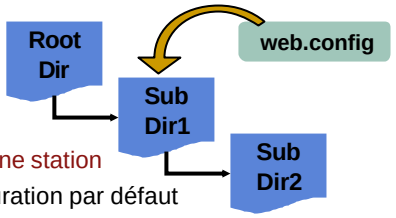
Configuration

- But
  - Fournir une configuration extensible pour les administrateurs et les développeurs
  - Structuration hiérarchique de l'application
- Solution
  - Sauvegarder les données de configuration dans des fichiers XML
    - Format lisible aussi bien par un programmeur que par une machine
- Spécifié dans les différentes sections de configuration, e.g.
  - `Security`, `SessionState`, `Compilation`, `CustomErrors`, `ProcessModel`, `HTTPHandlers`, `Globalization`, `AppSettings`, `WebServices`, `WebControls`, etc.
- Les information de configuration sont sauvegardées dans `web.config`
  - C'est un fichier (pas une DLL, Registry ou une métabase)
- `<!-- web.config peut avoir des commentaires -->`
  - Comme tous fichiers XML

Configuration

Configuration Hiérarchy

- Les fichiers de configuration sont sauvegardé dans les répertoires applicatifs
  - Le système détecte automatiquement les changement de configuration
- Architecture hiérarchique
  - S'applique au répertoire courant et à ses descendant



- Fichier de configuration 'racine' d'une station
  - Fourni un ensemble de configuration par défaut
  - Est hérité par toutes les applications web de la station

```
C:\WINNT\Microsoft.NET\Framework\v1.0.2914\
              config\machine.config
```

Configuration

`web.config` Sample

```
<configuration>
  <configsections>
    <add names="httpmodules"
       type
       ="System.Web.Config.HttpModulesConfigHandler"/>
    <add names="sessionState"
       type="..."/>
  </configsections>

  <httpModules>
    <!--- http module subelements go here -->
  </httpModules>

  <sessionState>
    <!--- sessionstate subelements go here -->
  </sessionState>
</configuration>
```

Configuration

Définition par l'utilisateurs

- Créer le fichier `web.config` dans le répertoire adéquat

```
<configuration>
  <appSettings>
    <add key="CxnString"
      value="localhost;uid=sa;pwd=;Database=foo"/>
  </appSettings>
</configuration>
```

- Récupérer les informations à l'exécution

```
string cxnStr = ConfigurationSettings
                  .AppSettings["CxnString"];
```

Configuration

Traitants de configuration spécifiques

- Étendre l'ensemble des traitants de configuration pour obtenir celle requise par une application particulière
- Implémenter l'interface:

  `System.Web.Configuration.IConfigurationSectionHandler`
- Ajouter le nouveau traitant à l'un des fichiers de configuration
  - `web.config`
  - `machine.config`

Agenda

- IIS
- Applications Web
- Configuration
- Trace
- Session
- Cache
- Gestion des erreurs
- Déploiement
- Disponibilité
- Sécurité
- Coté serveur
- Coté client
- Mise en œuvre des contrôles
- Développer des formulaires

Trace

- ASP.NET supporte le mode trace
  - La méthode ‘naturelle’ pour suivre la mise au point
  - Plus de trace en dur : `Response.Write()`
  - Les instructions de trace peuvent être activée ou inhibée (plus besoin de modifier le code)
- Solution élégante pour collecter tous les détail d’une requête
  - Arbre de contrôle du serveur
  - Variables du serveur, en-tête, cookies
  - Chaînes de paramètres : formulaire / requête
  - Le mécanisme de traces donne de très nombreuses information sur la page demandée
- Peut être activé par page ou au niveau applicatif

Trace

Méthodes et propriétés

- Méthodes
  - `Trace.Write` : écrit la catégorie et le texte de la trace
  - `Trace.Warn` : écrit la catégorie et le texte de la trace en rouge
- Propriétés
  - `Trace.IsEnabled`: vrai si les traces sont activées pour l'application ou pour cette page
  - `Trace.Mode`: `SortByTime`, `SortByCategory`
- Implémenté dans la classe `System.Web.TraceContext`

Trace

Trace au niveau page

- Pour activer les traces dans une page
  1. Ajouter une directive de trace en début de page
     - `<%@ Page Trace="True" %>`
  2. Ajouter les traces dans la corps de la page
     - `Trace.Write("MyApp", "Button Clicked");`
     - `Trace.Write("MyApp", "Value: " + value);`
  3. Accéder à la page depuis un navigateur

Trace

Trace au niveau applicatif

- Pour activer les traces sur de multiples pages
  1. Créer le fichier `web.config` à la racine de l'application

```
<configuration>
  <trace enabled="true" requestlimit="10"/>
</configuration>
```

  2. Lire une ou plusieurs pages de l'application
  3. Accéder aux traces à partir de l'URL

```
http://localhost/MyApp/Trace.axd
```

Trace

Démonstration sur les traces

- Démonstration : Trace1.aspx
  - Visualiser les traces obtenues

Trace

Tracer depuis un composant

- Pour ajouter des traces à un composant
  - Importer le namespace web
    `using System.Web;`
  - Activer les traces dans le constructeur de l'objet (optionel):
    `HttpContext.Current.Trace.IsEnabled = true;`
  - Écrire une trace
    `HttpContext.Current.Trace.Write("category","msg");`

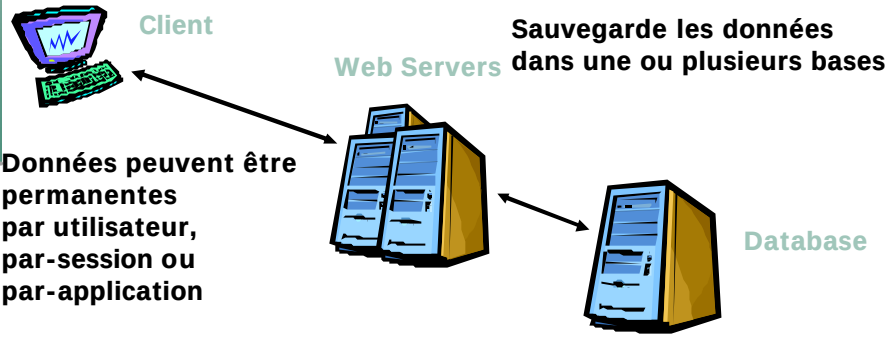
Agenda

- IIS
- Applications Web
- Configuration
- Trace
- Session
- Cache
- Gestion des erreurs
- Déploiement
- Disponibilité
- Sécurité
- Coté serveur
- Coté client
- Mise en œuvre des contrôles
- Développer des formulaires

Gestion de session

Le problème

- HTTP…
  - Quand et ou sauvegarder les données ?
  - Comment transmettre les données d'une page dans une autre ?
  - Comment contourner le protocole HTPP qui est sans état ?
- Three-Tier Architecture



Gestion de session

Client

- Gestion de session côté client, un scénario :
  - Le client requête une page initiale
  - Le serveur génère une réponse HTTP/HTML qu'il renvoie au client
    - Cette réponse inclue un description de la session (état)
  - L'utilisateur regarde la réponse et effectue une sélection, provoquant une autre requête auprès du serveur
    - Cette seconde requête contient la description de la session envoyée dans la réponse à la première requête
  - Le serveur (le même ou un autre) reçoit la requête et la traite

Gestion de session

Client

- URL dans un hyperlien (<a>)
  - Contient la requête
  - Visible aux utilisateurs
    - Bonne ou mauvaise chose
- Élément caché dans un formulaire
  - comme `__VIEWSTATE`
- Cookies
  - Limité à 4K
  - Peut être supprimé ou interdit par les utilisateurs

Gestion de session

Serveur Web (Middle-Tier)

- Variables d'application
  - Partagées par toutes les sessions, les utilisateurs
- Variables session
  - Nécessite de passer l'identifiant de session au client
  - Géré par la base de données ou par ASP.NET State Service
- Cache
  - Simalaire aux variables d'application
  - Peut être mis à jour périodiquement

Gestion de session

Base de données

- Niveau applicatif
  - Fait partie intégrante de la conception de la base de données
- Niveau session
  - Gestion de l'état de la session construite sur mesure dans la base de données
  - ASP.NET supporte la gestion de session au niveau de la base de données

Gestion de session

Dans ASP.NET

- ASP.NET supporte
  - Supporte différent mode de gestion de session
    - Par utilisateur
    - Par application
- Les données liées à la gestion des sessions peuvent être sauvegardées dans le serveur Web (middle-tier)

Gestion de session

Variables d'application

- Les variables de l'application sont sauvegardées dans une instance de `HttpApplicationState`
- Accédé depuis la propriété `Page.Application`
- L'objet `Application` peut être verrouillé
  - Gestion des accès concurrent
  - Nécessaire uniquement en cas de modification
- A utiliser avec prudence
  - Préférer les accès en lecture
  - Initialiser l'ensemble dans `global.asa`
  - Éviter de sérialiser vos pages

Gestion des sessions

- Session
  - Contexte communiqué par un utilisateur à un serveur par l'intermédiaire de multiple requête HTTP
  - Nécessaire pour construire une application ASP.NET
- HTTP est un protocole sans état, sans session
- ASP.NET introduit le concept de "session"
  - Identificateur de session : chaîne de 120 bit ASCII
  - Évènement de session : `Session_OnStart`, `Session_OnEnd`
  - Variables de session : données partagées par plusieurs requêtes
- ASP.NET améliore les sessions ASP

Gestion des sessions

Identificateur de session

- Par défaut, les SessionId sont sauvegardé dans un cookie (coté client)
- Il est aussi possible de sauvegarder le SessionId dans une URL
  - N'existe pas dans ASP
- Aucune modification de l'application n'est nécessaire
  - Tous les liens relatifs continuent de fonctionner

```
<configuration>
  <sessionState cookieless="true"/>
</configuration>
```

Gestion de session

Variables de session

- ASP sauvegarde les informations liées à la session dans le processus IIS
  - Si le serveur 'crash'… on perd la session
  - Une session est propre à un serveur Web
- ASP.NET sauvegarde les informations de session :
  - Dans le processus IIS
  - Dans un autre processus : ASP State NT service
  - Dans une base de données

```
<sessionstate inproc="false"
  server="AnotherServer" port="42424" />
```

```
<sessionstate inproc="false"
  server="AnotherServer" port="42424"
  usesqlserver="true" />
```

Gestion de session

Variables de session

- Les objets 'vivants' ne sont sauvegardés les objet session
  - ASP.NET sérialise les objets entre les requêtes
- ASP.NET fournit la possibilité
  - De récupérer l'état de l'application après un 'crash'
  - De tolérer la panne, le redémarrage sur serveur IIS
  - D'exécuter une application sur plusieurs stations (ferme de stations – Web Farm)
  - D'exécuter une application sur plusieurs processeurs (Web Garden)

Gestion de session

Variables d'application et de session

- Demonstration : ApplicationAndSession.aspx

Gestion de session

Transfert du contrôle entre les pages

- Lien sur une page
- Retour d'un post
- `Response.Redirect`
  - Redirection d'une requête HTTP
  - Demander au navigateur d'aller à une autre URL
- `Server.Transfer`
  - Idem redirection mais sur un seul serveur
- `Server.Execute`
  - Exécuter une page depuis une autre et lui rendre le contrôle
  - Les pages sont exécutées sur le même serveur

Plan

- IIS
- Applications Web
- Configuration
- Trace
- Session
- Cache
- Gestion des erreurs
- Déploiement
- Disponibilité
- Sécurité
- Coté serveur
- Coté client
- Mise en œuvre des contrôles
- Développer des formulaires

Cache

- De nombreux sites Web génèrent régulièrement les mêmes pages
  - Par exemple
    - Un catalogue de produit peut-être mis à jour chaque nuit
    - Et accédé en lecture des milliers de fois chaque jour
- Cache du coté serveur augmente notablement les performances et diminue le temps de réponse (facteur d'échelle)
- ASP.NET fournit le support pour
  - Cacher les pages à émettre
  - Cacher les données

Cache

de page out

- La page Web est complètement cachée (HTML)
- Il faut indiquer la durée de vie de la page dans le cache (en secondes)

```
<%@ OutputCache Duration="60“ VaryByParam="none" %>
```

- Il est possible de cacher plusieurs version d'un même page
  - Paramètres GET/POST ; utilisation de `VaryByParam`
  - En-tête HTTP ; utilisation de `VaryByHeader`
    - E.g. `Accept-Language`
  - En fonction du type de navigateur ou du client ; utilisation de `VaryByCustom`

Cache

Partiel d'une page out

- Il est possible de cacher
  - Uniquement une partie de page en y plaçant un contrôle utilisateur
  - De multiples versions d'un contrôle utilisateur en utilisant la propriété `VaryByControl`

Cache

dans le navigateur

- Ne pas confondre cache coté serveur et les mécanismes de cache coté client
  - Proxy
  - Cache du navigateur
- utiliser `Response.Cache` pour spécifier la politique de cache HTTP
  - Contient un objet `HttpCachePolicy`

Cache
de données

- Cache de données est similaire aux variables d'applications
- Peut cacher des objets, des bouts de document HTML, etc.
- Utilisation :
  - Recherche d'une donnée
  - Si `null` alors création de la donnée et insertion dans le cache

```
DataView Source = (DataView)Cache["MyData"];
if (Source == null) {
  Source = new DataView(ds.Tables["Authors"]);
  Cache["MyData"] = Source;       // Save in cache
}
```

- L'objet caché est sauvegardé dans la page c'est une instance de `System.Web.Caching.Cache`

Cache
de données

- Le cache peut être nettoyé pour pouvoir faire de la place aux nouveaux entrants
- Il est possible d'exprimer une date d'expiration
  - Temps absolu (e.g. minuit), relatif (dans 1 heure)
- Les données cachées peuvent dépendre d'un fichier ou d'une autre donnée du cache

```
Cache.Insert("MyData", Source, null,
   // Expire in 1 hour
   DateTime.Now.AddHours(1), TimeSpan.Zero);
Cache.Insert("MyData", Source,
   // Dependent on file
   new CacheDependency(Server.MapPath("authors.xml")));
```

Cache
de données

- Il est fréquent que quelques pages soit présente à de multiples exemplaires dans le cache
  - Généralement lorsque deux clients en demande l'accès simultanément
- Sans effet significatif la plupart du temps
  - Peut être gênant si le coût de la mise en copie est prohibitif, ou s'il y a d'autres effets secondaires
- Si cela arrive, il y a deux solutions
  - Contrôler la population du cache avec `Application_OnStart`
  - Synchroniser les accès au cache

Cache
de données

```
private static String cacheSynchronize = "myKey";

DataView Source = (DataView)Cache["MyDataSet"];
if (Source == null) {
  lock (cacheSynchronize) {
    Source = (DataView)Cache["MyDataSet"];
    if (Source == null) {
       // Have to test again
      // Open database ...
      Source = new DataView(ds.Tables["Authors"]);
      // Save in cache
      Cache["MyDataSet"] = Source;
    }
  }
}
```

Cache
de données

- ASP.NET page state maintenance is great, but `__VIEWSTATE` can get quite large
- Why store constant data in `__VIEWSTATE`?
  - E.g. dropdowns listing countries, states, days of the week, months, product categories, SKUs, etc.
- Instead, set `EnableViewState=false`, cache that data on the server, and populate the control from the cache in `Page_Load`
- Can cache data or even HTML
  - Use `Control.Render()` to obtain a control's HTML

Plan

- IIS
- Applications Web
- Configuration
- Trace
- Session
- Cache
- Gestion des erreurs
- Déploiement
- Disponibilité
- Sécurité
- Coté serveur
- Coté client
- Mise en œuvre des contrôles
- Développer des formulaires

Gestion des erreurs

- .NET CLR fourni une architecture unique pour la gestion des exceptions
  - Erreur à l'exécution sont propagées sous la forme d'exception
  - VB a été étentu et supporte `try/catch/finally`
- ASP.NET permet la définition de traitant d'exception
  - Redirection automatique des utilisateurs vers la page quand des erreurs non récupérées surviennent
  - Défini des messages d'erreurs clairs en lien avec les applications

Gestion des erreurs

Pages spécifiques

- Un utilisateur peut spécifier des pages d'erreur dans `web.config`

```
<configuration>
  <customerrors mode="remoteonly"
                defaultredirect="error.htm">
    <error statuscode="404"
           redirect="adminmessage.htm"/>
    <error statuscode="403"
           redirect="noaccessallowed.htm"/>
  </customerrors>
</configuration>
```

Gestion des erreurs

Error Events

- Surcharger la `Page.HandleError` pour récupérer toutes les erreurs non traitées dans cette page
- Un événement global applicatif est levé si un exception levée n'est pas traitée
  - Accès à la `Request` courante
  - Accès à l'objet `Exception`
  - Regarder l'évènement `HttpApplication.Error`
- Gestion des erreurs par défaut
  - Mise en log des erreurs (utilisation de la classe `EventLog`)
  - Envoir d'un email aux administrateurs (utilisation de la classe `SmtpMail`)

Gestion des erreurs

Générer des logs

```
<%@ Import Namespace="System.Diagnostics" %>
<%@ Assembly name="System.Diagnostics" %>
<script language="C#" runat=server>
public void Application_Error(object Sender, EventArgs E) {
  string LogName = "MyCustomAppLog";
  string Message = "Url " + Request.Path + " Error: «
             + this.Error.ToString()
  // Create event log if it doesn't exist
  if (! EventLog.SourceExists(LogName)) {
    EventLog.CreateEventSource(LogName, LogName);
  }
  // Fire off to event log
  EventLog Log = new EventLog();
  Log.Source = LogName;
  Log.WriteEntry(Message, EventLogEntryType.Error);
}
</script>
```

Gestion des erreurs

Envoyer un mail par SMTP

```
<%@ Import Namespace="System.Web.Util" %>
<%@ Assembly name="System.Diagnostics" %>
<script language="C#" runat=server>
public void Application_Error(object Sender,
EventArgs E) {
  MailMessage MyMessage = new MailMessage();
  MyMessage.To = "scottgu@microsoft.com";
  MyMessage.From = "MyAppServer";
  MyMessage.Subject = "Unhandled Error!!!";
  MyMessage.BodyFormat = MailFormat.Html;
  MyMessage.Body = "<html><body><h1>"
      + Request.Path
      + "</h1>" + Me.Error.ToString()
      + "</body></html>";
  SmtpMail.Send(MyMessage);
}
</script>
```

Agenda

- IIS
- Applications Web
- Configuration
- Trace
- Session
- Cache
- Gestion des erreurs
- Déploiement
- Disponibilité
- Sécurité
- Coté serveur
- Coté client
- Mise en œuvre des contrôles
- Développer des formulaires

Déploiement

- Par copie des fichiers (XCOPY)
  - Placer dans le répertoire `.\bin`
- Plus de DLL
  - Pas de déploiement de DLL, ni de mise à jour de registres
    - Sauf si vous utilisez COM ou d'autres DLLs
  - Pas de DLLs verrouillée
    - Les DLLs sont "partagées par copies" dans des répertoires cachés
    - Les fichiers .aspx sont automatiquement compilés
      - Faux pour codebehind
- Mise à jour du code à chaud (.aspx et assemblages)
  - Pendant le fonctionnement du serveur
  - Il n'est pas nécessaire stopper/redémarrer
- Les applications sont isolées
  - Chaque application à sa propre version des composants
- Désinstaller = `delete /s *.*`

Agenda

- IIS
- Applications Web
- Configuration
- Trace
- Session
- Cache
- Gestion des erreurs
- Déploiement
- Disponibilité
- Sécurité
- Coté serveur
- Coté client
- Mise en œuvre des contrôles
- Développer des formulaires

Disponibilité

- ASP.NET handles failures in user and system code
- Detects and recovers from problems
  - Access violations, memory leaks, deadlocks
- Supports pre-emptive cycling of apps
  - Time and request-based settings

Disponibilité

Modèle de recouvrement

- ASP.NET s'exécute dans un autre processus
  - `aspnet_ewp.exe`
  - 'Protège' les IIS des pannes d'ASP
- Il est aussi possible de configurer les processus ASP.NET pour 'survivre'
  - Aux fuites de mémoire

```
<processmodel memorylimit="75" />
```

  - Interblocage

```
<processmodel requestqueuelimit="500" />
```

Availability
Preemptive Recycling

- ASP.NET optionally supports pre-emptive cycling of worker processes
  - Eliminates need for admins to “kick the server” once a week
- Can be configured two ways:
  - Time based (reset every n minutes)

```
<processmodel timeout=“60” />
```

  - Request based (reset every n requests)

```
<processmodel requestlimit=“10000” />
```

Agenda

- IIS
- Applications Web
- Configuration
- Trace
- Session
- Cache
- Gestion des erreurs
- Déploiement
- Disponibilité
- Sécurité
- Coté serveur
- Coté client
- Mise en œuvre des contrôles
- Développer des formulaires

Sécurité

- Pourquoi sécuriser
  - Protéger les accès à certaines parties d'un site
  - Protéger les accès en écriture à certaines données
- Configuration de la sécurité
  - `Tag <security>` dans le fichier `web.config`
- Authentification, Autorisation, Usurpation d'identité
- Accès sécurisé au code
  - Est-ce que le code est bien celui prévu ?
  - Protéger le serveur contre du code malveillant

Sécurité
Authentification

- Qui êtes vous ?
- Le serveur doit authentifier les client
- Les clients doivent authentifier le serveur
  - Utilisation de Kerberos
- Besoin d'un annuaire pour sauvegarder les comptes clients
  - NT: Security Accounts Manager
    - OK pour une utilisation sur un Intranet (40,000 comptes)
  - Windows 2000: Active Directory
    - OK pour une utilisation sur un Intranet ou sur Internet

Sécurité
Authentification

Authentification IIS
- Compte 'Anonyme'
  - Le même W2K/NT compte pour tous les visiteurs
- Authentification de base
  - Support standard
  - Password envoyé en texte clair
- Authentification intégrée Windows
  - NTLM
  - Kerberos (Windows 2000)
- Certificats Client
  - Associés à un compte W2K/NT

Authentification ASP.NET
- Fournis le module d'accès à 'Passport'
  - Utilisation du profil API
- Personnalisable par des formulaires
  - Facile à utiliser, utilisation de cookie
  - Active un écran de login
  - Supporte les mécanismes de certifications avec base de données, exchange, etc.

Sécurité
Autorisation

- Étape 1 (authentification)
  - Qui êtes vous ?
- Etape 2 (autorisation)
  - Qu'avez-vous le droit de faire
- W2K/NT DACLs (Discretionary Access-Control List)
  - Permissions et refus en lecture/écriture/exécution/etc. accordés à un utilisateur ou à un groupe d'utilisateurs
- IIS fournit aussi un contrôle
  - Read, write, run script, run executable, directory browsing, script access for virtual directories, directories and files

Sécurité
ASP.NET Autorisation

- Définition des autorisations par utilisateur ou par rôle
- Les rôles définissent des groupes logiques
  - Exemple: “Utilisateur”, “Administrateur”, “Invité”, etc.
  - Permet de séparer les aspects développement et administration
- Les développeurs peuvent inclure dans le code les étapes ou les droits doivent être vérifiés
  - `if (User.IsInRole(“Admin”) { }`
- IIS authentifie l’utilisateur
  - Le ticket est passé à l’application ASP.NET
  - ASP.NET qui fonctionne avec les droits de cet utilisateur
  - Les accès sont permis selon les règles de NTFS

Agenda

- IIS
- Applications Web
- Configuration
- Trace
- Session
- Cache
- Gestion des erreurs
- Déploiement
- Disponibilité
- Sécurité
- Coté serveur HTTP
- Coté client
- Mise en œuvre des contrôles
- Développer des formulaires

HTTP Runtime

- Modèle d'exécution de bas niveau
  - Remplacement logique d'ISAPI
- Construit pour résister aux pannes
  - Le code 'managed' s'exécute dans un processus 'unmanage'
  - Chaque requête est traitée par un fil d'exécution indépendant
  - Multi-threaded
- Fournis des possibilités pour remplacer/étendre le cœur du serveur
  - Élimine la « boite noire magique » d'ASP/IIS
  - Supporte d'autres abstraction de programmation
- `HttpModules` permet d'intercepter les requêtes
  - Similaire au filtre ISAPI
  - Utilisé pour l'authentification, la gestion de l'état d'une session, le cache en sortie
- `HttpHandlers` permet de construire des requêtes de bas niveau
  - Implémenté par des objets `IHttpHandler`
  - Activé pour traiter les requêtes
  - Pages et services construits à l'aide de IHttpHandler
- Ajout de modules et de traitant dans `web.config`

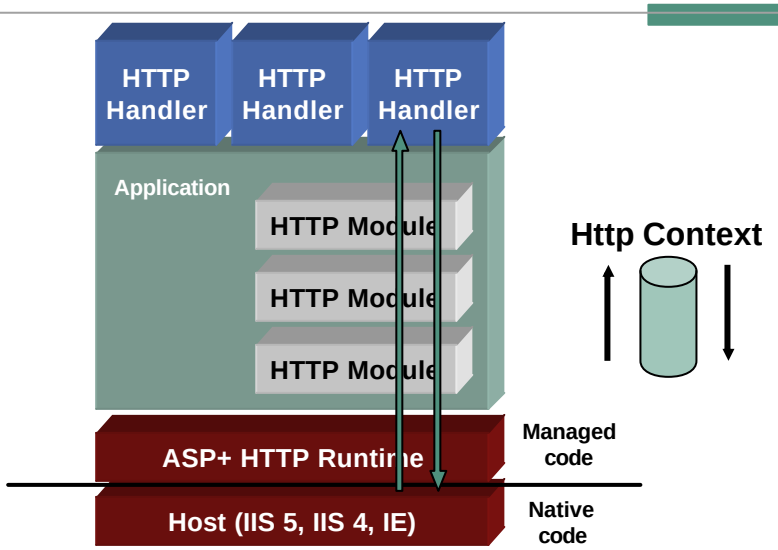
HTTP Runtime

- ASP.NET Web Forms et les services Web Services sont construit en utilisant le runtime ASP.NET HTTP
  - Pas d'astuces cachées
  - Il est possible de construire la même chose

HTTP Runtime

- HTTP module pipeline
  - Managed classes
  - Chaque module implémente une interface spécifique
    - Par exemple : gestion des états, sécurité
  - Chaque requête est traitée par le même pipeline
- Traitant de requête (handler)
  - Managed classes
  - Plusieurs traitant de requête pour une seule application
    - Mais une seule par URL

HTTP Runtime



HTTP Runtime
HTTPContext

- `HttpContext` est un objet qui encapsule tout les information liées à une requête Http
  - `System.Web.HttpContext`
- `HttpHandler` et `HttpModules` accède aux objets `HttpContext`
- `HttpHandlers` et `HttpModules` peuvent ajouter des objets à `HttpContext`

HTTP Runtime
System.Web.HTTPContext

```
public class HttpContext {
  public HttpRequest          Request     { get; }
  public HttpResponse         Response    { get; }
  public HttpServerUtility    Server      { get; }
  public HttpApplication      Application { get; }
  public HttpSession          Session     { get; }
  public IPrincipal           User        { get; }
  public Cache                Cache       { get; }
  public IDictionary          Items       { get; }

  public IHttpHandler         Handler     { get; }
  public Exception            Error       { get; }
  public DateTime             TimeStamp   { get; }

  public Object               GetConfig(String name);
}
```

HTTP Runtime
HTTPRequest/HTTPResponse

- Les objets HttpRequest/HttpResponse contiennent les requêtes/réponses

HTTP Runtime
HttpHandlers

- HttpHandlers permet de traiter des URLs ou des groupes d'URL avec la même extension au sein de la même application
  - Analogue aux extensions ISAPI
- Quelques exemple d'HTTPHandler
  - ASP.NET Page Handler
  - ASP.NET Service Handler
  - Server-Side XSL Transformer
  - Image Generator Service

HTTP Runtime
`HttpHandlers`

- Construit comme des classes qui implémentent l'interface `System.Web.IHttpHandler`

```
public interface IHttpHandler {
  public void ProcessRequest(HttpContext context);
  public bool IsReusable();
}
```

- Méthode `ProcessRequest()` Method
  - Méthode qui exécute chaque requête
- Méthode `IsReusable()`
  - Indique si le pooling est activé

HTTP Runtime
`HttpHandler` Registration

1. Compile and deploy .NET Library DLL within "bin" dir under application vroot
2. Register `HttpHandler` in `web.config`

```
<configuration>
  <httphandlers>
    <add verb="*" path="foo.bar" type="assembly#class"/>
  </httphandlers>
</configuration>
```

3. Ensure that `HttpHandler` file extension is registered within IIS to xspisapi.dll
   - Hint: Copy/Paste ".aspx" registration entry

HTTP Runtime
HttpModules

- HttpModules enable developers to intercept, participate or modify each individual request into an application
- HttpModule Examples:
  - Output Cache Module
  - Session State Module
  - Personalization State Module
  - Custom Security Module

HTTP Runtime
HttpModules

- Built as classes that implement the System.Web.IHttpModule interface

```
public interface IHttpModule {
  public String ModuleName { get; }
  public void Init(HttpApplication application);
  public void Dispose();
}
```

HTTP Runtime

`HttpModules`

- `HttpModules` can use Init() method to sync any `HttpApplication`
  - Global Application Events
  - Per Request Application Events
- Global Application Events
  - `Application_Start`
  - `Application_End`
  - `Application_Error`

HTTP Runtime

Per Request Application Events

- Per Request Events (in order):
  - `Application_BeginRequest`
  - `Application_AuthenticateRequest`
  - `Application_AuthorizeRequest`
  - `Application_ResolveRequestCache`
  - `Application_AquireRequestState`
  - `Application_PreRequestHandlerExecute`
  - `Application_PostRequestHandlerExecute`
  - `Application_ReleaseRequestState`
  - `Application_UpdateRequestCache`
  - `Application_EndRequest`
- Per Request Transmission Events:
  - `Application_PreSendRequestHeaders`
  - `Application_PreSendRequestContent`

HTTP Runtime

HTTP Module Registration

1. Compile and deploy .Net Library DLL within “bin” dir under app vroot
2. Register `HttpModule` in `web.config`

```
<configuration>
  <httpmodules>
    <add type=“assembly#classname”/>
  </httpmodules>
</configuration>
```

Agenda

- IIS
- Applications Web
- Configuration
- Trace
- Session
- Cache
- Gestion des erreurs
- Déploiement
- Disponibilité
- Sécurité
- Coté serveur
- Coté client
- Mise en œuvre des contrôles
- Développer des formulaires

Targeting Uplevel Clients

- But
  - Pages render more richly to uplevel clients, but work well in downlevel clients too
- Page developers can identify target client

```
<%@ Page ClientTarget="Uplevel" %>
<%@ Page ClientTarget="Downlevel" %>
<%@ Page ClientTarget="Auto" %>
```

Targeting Uplevel Clients
`Request.Browser`

- Provides more granular control of what is rendered to a given client
- Exposes specific client capabilities

```
if (Request.Browser.Frames == True) {
  // Use frames
}
```

Agenda

- IIS
- Applications Web
- Configuration
- Trace
- Session
- Cache
- Gestion des erreurs
- Déploiement
- Disponibilité
- Sécurité
- Coté serveur HTTP
- Coté client
- Mise en œuvre des contrôles
- Développer des formulaires

Creating Controls

- ASP.NET provides two ways to create your own server-side controls
  - User Controls: Essentially a mini .aspx file
    - Formerly called a “Pagelet”
  - Custom Controls: You derive a class from `System.Web.UI.Control`

Creating Controls
User Controls

- Provide a simple way for page developers to author controls
  - Stored in a .ascx file
- Not just a server-side include
- Enables full encapsulation
  - Supports nested controls
  - Separate code namespace
  - Separate code language
- Can partition work across multiple developers
- Great way to reuse work across multiple pages and applications

Creating Controls
Registering User Controls

- Registers user control for use on a page

```
<%@ Register TagPrefix="Acme“ TagName="Message"
    Src="pagelet1.aspc" %>

...

<Acme:Message runat="server"/>
```

Creating Controls

Exposing a User Control Object Model

- User controls can expose object model to pages
- Properties, fields, events, methods
  - Just make them `public`

```
<script language="C#" runat="server">
  public string Color = "blue";
</script>

<font color=<%=Color%>>
This is a simple message pagelet
</font>
```

Creating Controls

Programmatic Use of User Controls

- `Page.LoadControl(string source)`
  - Dynamically instantiates a user control
- Get and customize an instance:

```
foo = Page.LoadControl("foo.ascx");
foo.color = "red";
```

- Insert into the control hierarchy:

```
myPanel.Controls.Add(foo);
```

Creating Controls
Custom Controls

- A class that you create
- Derived from `System.Web.UI.Control`

```
using System;
using System.Web;
using System.Web.UI;

public class MyControl : Control {
  protected override void Render(HTMLTextWriter w) {
    w.Write("<h1>Control output</h1>");
  }
}
```

Creating Controls
Custom Controls

- Must implement `Render()` method
- Can expose properties, methods and events
- Should maintain state
- Should handle postback data
  - Can generate client-side script to do postback
- Should handle child controls
  - Render them
  - Handle events
- Can expose and implement templates
- Can handle data binding

Creating Controls

Custom Controls vs. User Controls

User Controls	Custom Controls
Good for application-specific UI	Good for reuse, encapsulate common UI
Easy to build	Can be more complex to build
Less flexibility, performance, designer support	Total flexibility, better performance, and designer support
No template support	Can support templates

Agenda

- IIS
- Applications Web
- Configuration
- Trace
- Session
- Cache
- Gestion des erreurs
- Déploiement
- Disponibilité
- Sécurité
- Coté serveur
- Coté client
- Mise en œuvre des contrôles
- **Développer des formulaires**

Developing Web Forms
Using Notepad

- Incredibly simple: just create .aspx file as a text file, edit, and hit the page
- That's how the demos in this module were done
- Use tracing as a debugging aid

Developing Web Forms
Using Visual Studio.NET

- Create a new project
  - Choose "Web Application"
  - Specify URL of application

Developing Web Forms
IBuySpy

- http://www.ibuyspy.com

Developing Web Forms
Debugging

- Basic Debugger ships with the SDK
  - Multi-language
  - Single stack trace
  - Breaks, watches, etc.
- Visual Studio® Debugger
  - Adds remote debugging support
  - Supports managed/unmanaged debugging

Developing Web Forms
Debugging

1. Create `web.config` file in application root

```
<configuration>
  <compilation debugmode="true"/>
</configuration>
```

2. Attach using debugger
3. Set breakpoints
4. Hit page or service

Pour aller plus loin…

- Quick Start Tutorial
  `http://localhost/quickstart/ASPPlus/default.htm`
- ASP.NET Overview
  `http://msdn.microsoft.com/msdnmag/issues/0900/ASPPlus/ASPPlus.asp`
- Caching
  `http://msdn.microsoft.com/voices/asp04262001.asp`
- Session state
  `http://msdn.microsoft.com/library/default.asp?URL=/library/welcome/dsmsdn/asp12282000.htm`
- Validation
  `http://msdn.microsoft.com/library/techart/aspplusvalid.htm`
- Tracing
  `http://msdn.microsoft.com/library/default.asp?URL=/library/welcome/dsmsdn/asp01252001.htm`