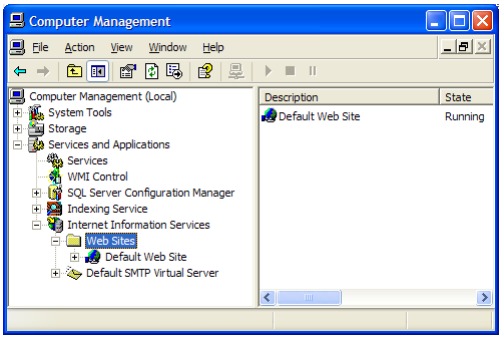
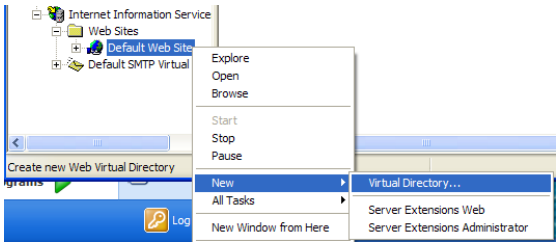
Alojamento (*hosting*) de objectos remotos no *IIS*

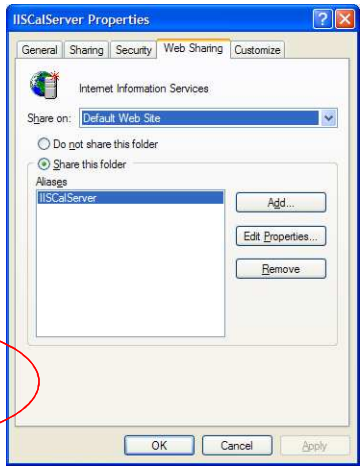
Em vez alojar os objectos remotos num processo Windows, é possível alojar objectos remotos no servidor *Web IIS - Internet Information Services*



Criar uma directoria virtual (Windows XP com IIS 6.0)



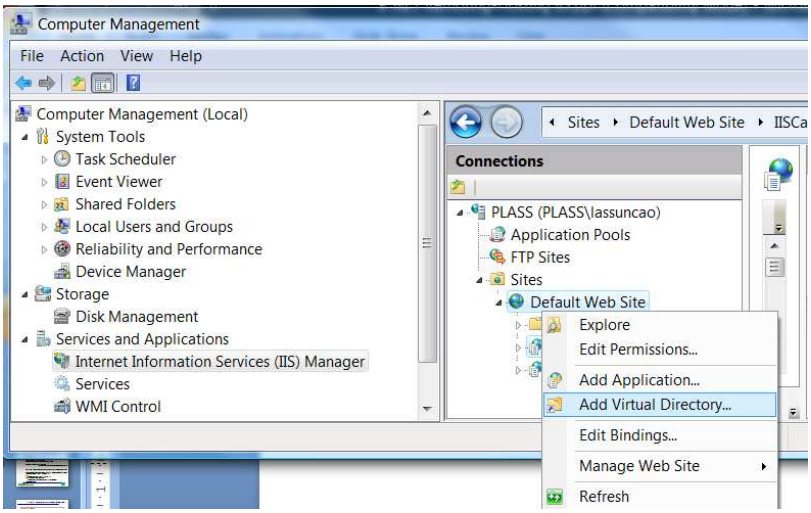
Criar uma directoria virtual a partir do IIS, definindo um alias (parte do *url*) e localizado num *folder* do *file system*



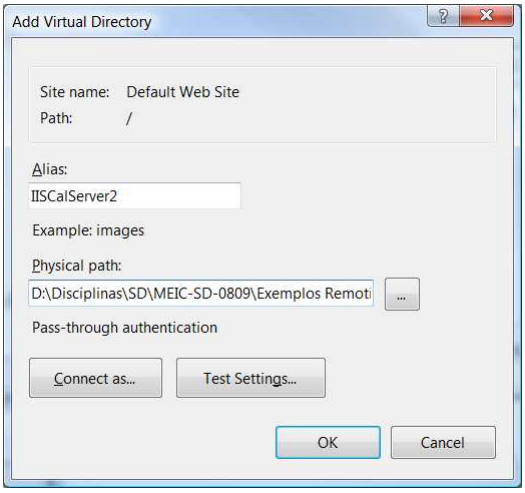
Não existe em Windows Vista e Windows 7 ?

Em *properties* de um *folder* é também possível criar, de forma mais simples, uma directoria virtual no IIS

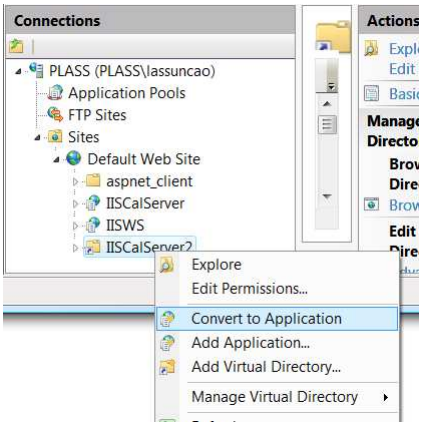
Criar uma directoria virtual em IIS7 (Windows Vista e Windows 7)

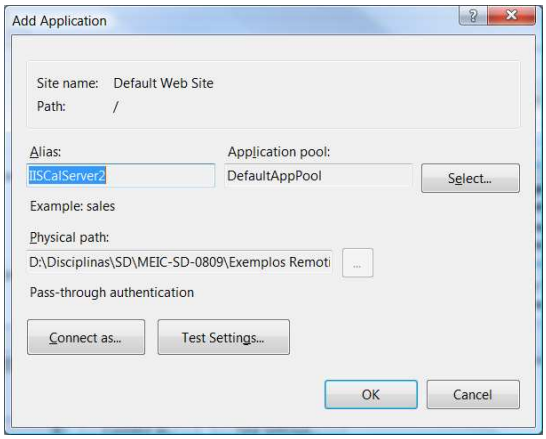


Criar uma directoria virtual em IIS7 (Windows Vista e Windows 7)



Criar uma aplicação em IIS7 (Windows Vista e Windows 7)

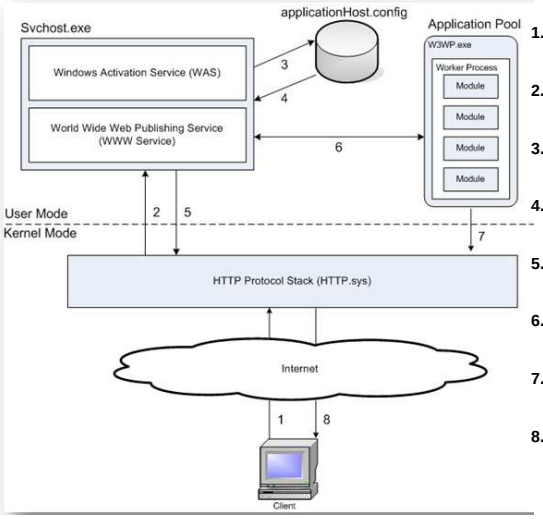




IIS7 Application pools

- Grupo de *URLs* (aplicações) servidos por um ou mais *worker processes*;
- Um *application pool* com múltiplos *worker process* é designada por *Web Gardens*
- Definem fronteiras para as aplicações, impedindo que sejam afectadas por outras aplicações fora da *application pool*;
- Aumentam a disponibilidade . Se uma *application pool* falhar, as aplicações noutras *application pools* não são afectadas;
- Melhoram a segurança. Ao isolarem as aplicações impedem que os recursos de uma aplicação possam ser acedidos por outra fora do *application pool*;
- No IIS 7, as *application pools* executam-se em dois modos:
  - *integrated mode* - o servidor usa o pipeline integrado ASP.NET para processar o pedido;
  - *classic mode* - o server continua a encaminhar o pedido para a componente *Aspnet_isapi.dll* como se a aplicação se executasse em IIS 6.0;

HTTP Request Processing in IIS 7



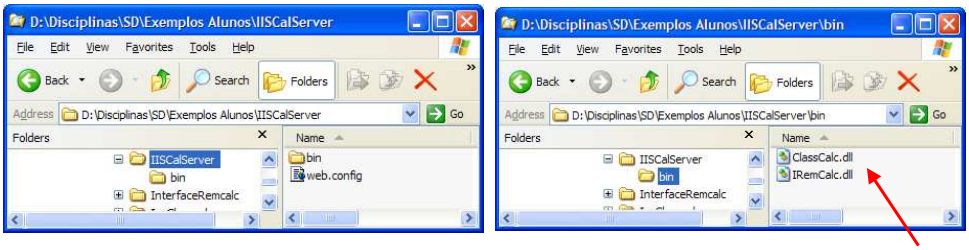
1. **HTTP.sys intercepts a client HTTP request for a resource on the Web server.**
2. **HTTP.sys contacts WAS to obtain configuration information store.**
3. **WAS requests configuration information from the configuration store, *applicationHost.config*.**
4. **The WWW Service receives configuration information, such as application pool and site configuration.**
5. **The WWW Service uses the configuration information to configure HTTP.sys.**
6. **WAS starts a worker process for the application pool to which the request was made.**
7. **The worker process processes the request and returns a response to HTTP.sys.**
8. **The client receives a response.**

http://learn.iis.net/page.aspx/101/introduction-to-iis-7-architecture/

Publicar o objecto remoto (igual em XP IIS 6.0 e em Vista/Win 7 com IIS 7)

A partir do *folder* que é directoria virtual:

1. **Criar uma directoria de nome *bin*;**
2. **Em /bin colocar o *assembly* que implementa o objecto remoto e os *assemblies* de que depende;**
3. **Na directoria virtual colocar o ficheiro de configuração, com o nome *web.config*, que publica o objecto remoto**



assemblies* do exemplo *CalcSimplesConfig

Ficheiro de configuração *web.config*

```
<configuration>
  <system.runtime.remoting>
    <application>
      <!-- Note a ausencia de Channel e port -->
      <!-- Como o objecto esta hosted no IIS, o canal: http e porto 80 -->
      <service>     <!-- Namespace.Type, Assembly  -->
        <wellknown type="ClassCalc.Calc, ClassCalc"
                   mode="Singleton"
                   objectUri="RemoteCalc.soap" />
      </service>
    </application>
  </system.runtime.remoting>
</configuration>
```


Cliente (idêntico ao já apresentado no exemplo *CalcSimplesConfig*)

```
class Cliente   {

      static void Main()     {
         string configfile = "Client.exe.config";
         RemotingConfiguration.Configure(configfile, false);

         WellKnownClientTypeEntry[] entries =
                                   RemotingConfiguration.GetRegisteredWellKnownClientTypes();


         ICalc robj = (ICalc)Activator.GetObject(entries[0].ObjectType, entries[0].ObjectUrl);

         if (RemotingServices.IsTransparentProxy(robj)) Console.WriteLine("robj é remoto");
         Console.WriteLine("5+8={0}", robj.Add(5, 8));
         try   {
             Console.WriteLine("5/2={0}", robj.Div(5, 0));
         } catch (DivideByZeroException e)   {
             Console.WriteLine("Divisão por zero: {0}", e.Message);
         }
         Console.ReadLine();
      }
  }
```

Ficheiro de configuração do Cliente - Client.exe.config

```
<configuration>
  <system.runtime.remoting>
    <application>
      <channels>
        <channel ref="http" port="0" />
      </channels>
    <client>
      <!-- Partilhar a Interface -->
      <wellknown type="IRemCalc.ICalc, IRemCalc"
                 url="http://localhost/IISCalServer/RemoteCalc.soap" />
                 <!-- objecto hosted no IIS no site IISCalServer (virtual directory) -->
    </client>
    </application>
  </system.runtime.remoting>
</configuration>
```


Autenticação no acesso a objectos remotos

Embora em .Net Remoting não esteja implementado nenhum mecanismo de autenticação de utilizadores, podemos recorrer aos mecanismos básicos de autenticação Windows baseados no conceito de *Windows Identity* e *Windows Principal*.
No método abaixo apresentado, os utilizadores que não pertençam ao grupo *Administrators* recebem uma excepção.

```
public string only4Admins(string msg)       {
        Console.WriteLine("Execução com verificação de Identitidade");
        Console.WriteLine("Identidade Corrente: {0}",WindowsIdentity.GetCurrent().Name);
        IPrincipal cliRem = System.Threading.Thread.CurrentPrincipal;   //Cliente Remoto
        Console.WriteLine("Identidade Cliente Remoto: {0}",cliRem.Identity.Name);
        WindowsPrincipal wprinc = new WindowsPrincipal((WindowsIdentity)cliRem.Identity);
        if (!wprinc.IsInRole(@"BUILTIN\Administrators"))
           throw new Exception("Não tem privilegio de Administrator!");
        else   {
           Console.WriteLine("Remote user is Administrator");
           return "Remote object answer: ola Administrator " + msg;
        }
}
```

Analise e teste o exemplo apresentado nas aulas: TcpSecureIdentity.zip
Note que a configuração do canal (cliente e servidor) tem de ser feita com *ensureSecurity* a *true*: RemotingConfiguration.Configure("Cliente.exe.config", true);