This text was added by using code.Несколько сайтов на одном биндинге.

Виртуальные директории,

Applications,

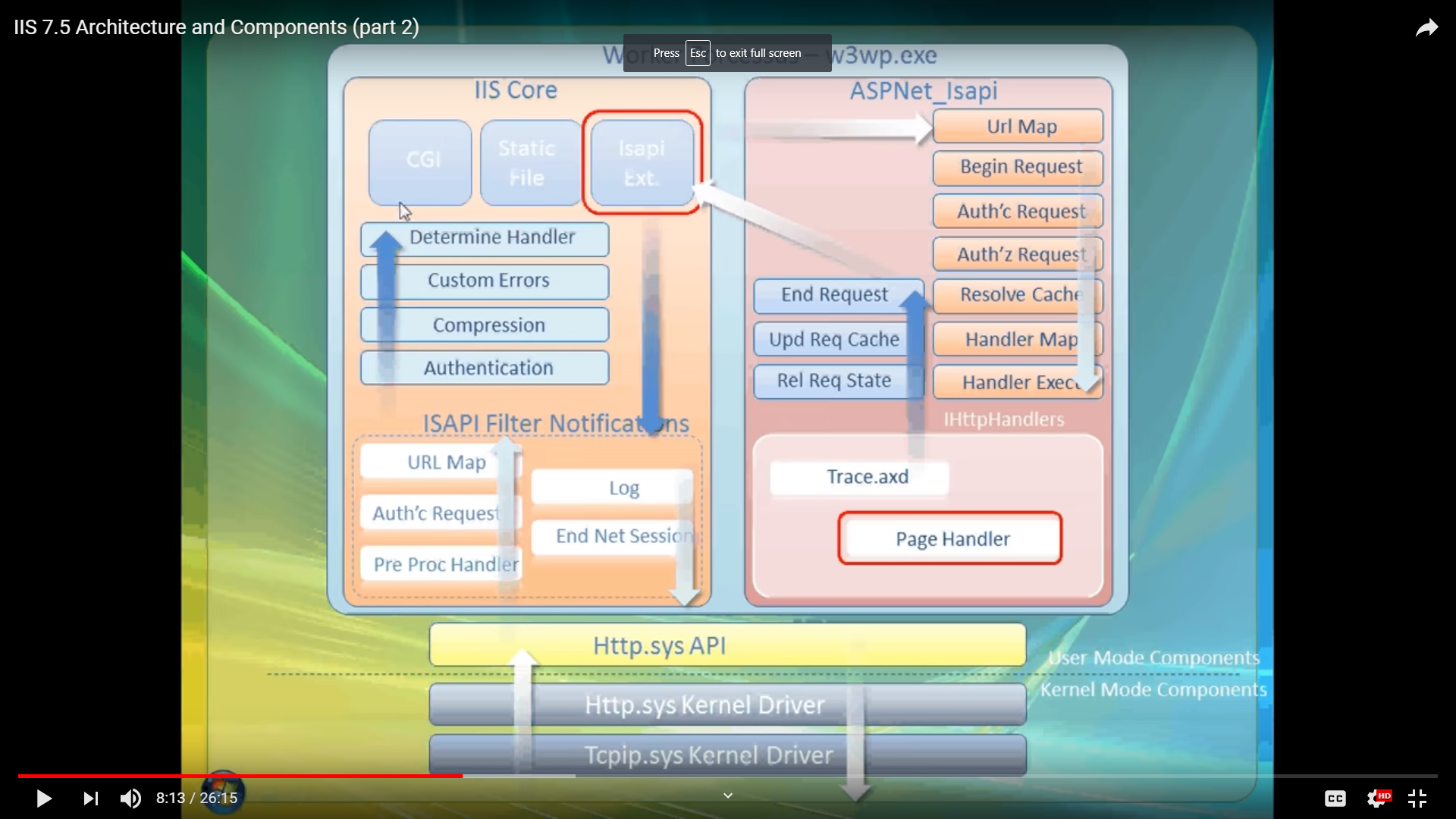
Деплоинг,

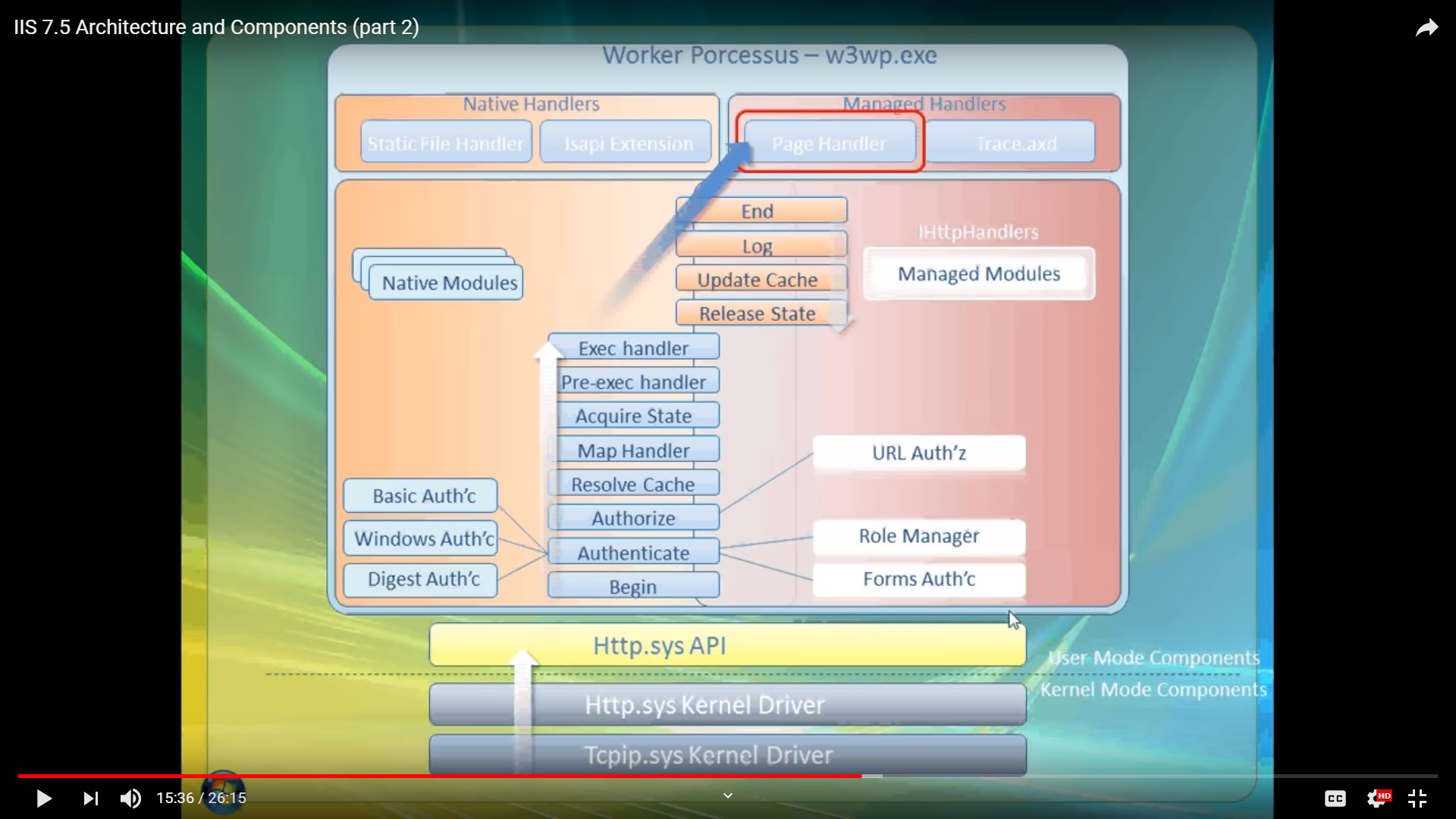
Дефолтные странитцы,

Конфигурация,

Как создать распределенные сайты

HttpHandler





**Notes**

C:\Windows\System32\inetsrv\Config\applicationhost.config

C:\Windows\System32\drivers\etc\hosts

* **system.applicationHost** - Contains configuration settings for sites, applications, virtual directories, and application pools. These are centralized settings that cannot be distributed.

Modules(IhttpModule methods Init, Dispose) – subscribe handlers on events lifecycle for (for example add some handlers to response) –> web.config(system.webserver -> modules) or iis manager -> <add/remove name=”” type=”{namespace.class}” precondition=” integratedMode,managedHandler”/> - precondition – defines use cases for module like (managedHandler use only by managed handlers)

Benefits of integrated modules

 The new request-processing architecture consists of an ordered list of native and managed modules that perform specific tasks in response to requests.

First, all file types can use features that were originally available only to managed code. For example, you can now use ASP.NET Forms authentication and Uniform Resource Locator (URL) authorization for static files, Active Server Pages (ASP) files, and all other file types in your sites and applications.

Second, this design eliminates the duplication of several features in IIS and ASP.NET. For example, when a client requests a managed file, the server calls the appropriate authentication module in the integrated pipeline to authenticate the client. In previous versions of IIS, this same request would go through an authentication process in both the IIS pipeline and in the ASP.NET pipeline.

Third, you can manage all of the modules in one location, instead of managing some features in IIS and some in the ASP.NET configuration. This simplifies the administration of sites and applications on the server.

public class SyncModule : IHttpModule{

public void Init(HttpApplication app)  
{  
 app.BeginRequest += new EventHandler(OnBeginRequest);  
}

public void Dispose(){ }

public void OnBeginRequest(Object s, EventArgs e)  
{  
 HttpApplication app = s as HttpApplication;  
 app.Context.Response.Write("Hello from OnBeginRequest in custom module.<br>");  
 if(\_eventHandler!=null)  
 \_eventHandler(this, null);  
}

}

HttpHandlers(IHTTPHandler methods ProcessingRequest(contetxt) prop IsReusable -> )

An application pool identity allows you to run an application pool under a unique account without having to create and manage domain or local accounts. The name of the application pool account corresponds to the name of the application pool. The image below shows an IIS worker process (W3wp.exe) running as the DefaultAppPool identity.

ApplicationPoolIdentity - When we create new Application pool IIs creates a virtual account with the name of this application pool and run the application pool worker process under this account. This is the least privileged account

<site name="MySite" id="3">

<application path="/" applicationPool="MySite">

<virtualDirectory path="/" physicalPath="d:\mysite\www" />

</application>

</site>

<site name="Contoso" id="2" serverAutoStart="true">

<application path="/">

<virtualDirectory path="/" physicalPath="C:\Contoso\Content" />

</application>

<application path="/CRM">

<virtualDirectory path="/" physicalPath="C:\Contoso\Content\CRM" />

<virtualDirectory path="/Images" physicalPath="E:\Images" />

</application>

<bindings>

<binding protocol="http" bindingInformation="\*:80:www.contoso.com" />

</bindings>

</site>

Каждый <site> содержит коллекцию <application>. Всегда будет по крайней мере одно приложение, которое определяет корневое приложение, /.

Атрибут applicationPool указывает, какой пул приложений использовать.

Обратите внимание, что существует единственный дочерний элемент: virtualDirectory.

Каждый application имеет дочерний набор элементов virtualDirectory, и в этой коллекции обычно будет хотя бы один элемент.

По умолчанию <virtualDirectory> в корневом приложении сообщает нам:

* это это корень (path="/") и
* что он физически находится в файловой системе в d:\mysite\www (physicalPath="d:\MySite\www").

path каждого virtualDirectory относится к path, указанному в родительском пути application.

Briefly, a site contains one or more applications, an application contains one or more virtual directories, and a virtual directory maps to a physical directory on a computer.

**Request Filtering**

<system.webServer>

<security>

<requestFiltering

allowDoubleEscaping="false"> - prevents attacks that rely on double-encoded requests

<requestFiltering

allowHighBitCharacters="true"> - **Filter High Bit Characters**

This feature either allows or rejects all requests to IIS that contain non-ASCII characters

<requestFiltering>

<fileExtensions allowUnlisted="true" >

<add fileExtension=".asp" allowed="false"/> - **Filter Based on File Extensions**

<requestFiltering>

<requestLimits

maxAllowedContentLength="30000000"

maxUrl="260"

maxQueryString="25" /> - **Filter Based on Request Limits**

<requestFiltering>

<verbs

allowUnlisted="false">

<add verb="GET" allowed="true" /> - list of verbs that IIS accept as part of a request.

<requestFiltering>

<denyUrlSequences>

<add sequence=".."/> - defines a list of sequences that IIS reject when it is part of a request.

<requestFiltering>

<hiddenSegments>

<add segment="BIN"/> (denying access to bin dirrectory)

</hiddenSegments> - **Filter Out Hidden Segments -**  allows you to define which segments are "servable."

**enhanced Request Filtering features**

<requestFiltering>

<filteringRules>

<filteringRule name="BlockFooInHeader">

<scanHeaders>

<add requestHeader="Foo-Header" />

</scanHeaders>

<denyStrings>

<add string="Foo" />

<add string="Bar" />

</denyStrings> - block strings "Foo" and "Bar" in header "Foo-Header"

<requestFiltering>

<filteringRules>

<filteringRule name="BlockSqlCommands" scanQueryString="true">

<appliesTo>

<add fileExtension=".asp" />

</appliesTo>

<denyStrings>

<add string="Insert" />

<add string="Table" />

</denyStrings> -- block strings "Insert" and "Table" in the query string sent with any ".asp" page

<requestFiltering>

<alwaysAllowedUrls>

<add url="Login.asp" />

</alwaysAllowedUrls>

<alwaysAllowedQueryStrings>

<add queryString="Allow=true" />

</alwaysAllowedQueryStrings> - This new feature allows you to specify safe URLs and query strings that will bypass all the deny rules defined.

<requestFiltering>

<denyQueryStringSequences>

<add sequence=".." />

<add sequence="./" /> - **Deny List of URL Sequences**

<requestFiltering unescapeQueryString="true">

<denyQueryStringSequences>

<add sequence="script" />

</denyQueryStringSequences> - **Checking for both Escaped and unEscaped Query String**

Dzmitry, 3:45 PM

Про фильтры я вот это отвечал

"Для веб-сайта или приложения можно указать Request Filtering, которая представляет собой набор правил, определяющих должен ли быть предоставлен доступ к тем либо иным ресурсам. Запрет на доступ к ресурсам может определяться на основании расширения файла, сегмента, урла, HTTP глагола, наличия указанного заголовка. "

**DOMAINS**

Связано с тем, если возникает необходимость развернуть несколько веб-приложений на одном айпишнике. Тогда для каждого можно использовать свое доменное имя и потом уже по этому имени определять к какому из них адресуется запрос.

# Security Isolation for Web Sites

if you create an application pool with the name "MyNewAppPool," a SID with the name "MyNewAppPool" is created in the Windows Security system. Resources can be secured using this identity. Note that the identity is not a real user account

Virtual Folder or Virtual Directory

A Virtual Folder or Virtual Directory is just a link to a physical folder somewhere on the server. This folder becomes part of the website structure and you can use the virtual directory in the path part of URLs. Code that executes in Virtual Directories will execute in the same "Application" as it's parent.

An Application is where the code that runs inside that "folder" has it's own Session state and Application state. It is in effect a new standalone application living underneath the root application.

An application is a group of files that delivers content or provides services over protocols, such as HTTP. When you create an application in IIS, the application's path becomes part of the site's URL.

In IIS 7 and above, each site must have an application which is named the root application, or default application. However, a site can have more than one application.

For example, if you were to deploy an ASP.NET application into a site that had an Application folder called /myapp then that application would have it's own application domain, session state, application state completely separate from another ASP.NET application running in /. For example: if you set an Application value Application["Thing"] = 123 in the root application and then did the same but with a different value in /myapp then Application["Thing"] in the root would not be overwritten by the assignment in /myapp.

Another thing you can do with Application's is specify a different Application Pool to run under.

Орор