Оглавление

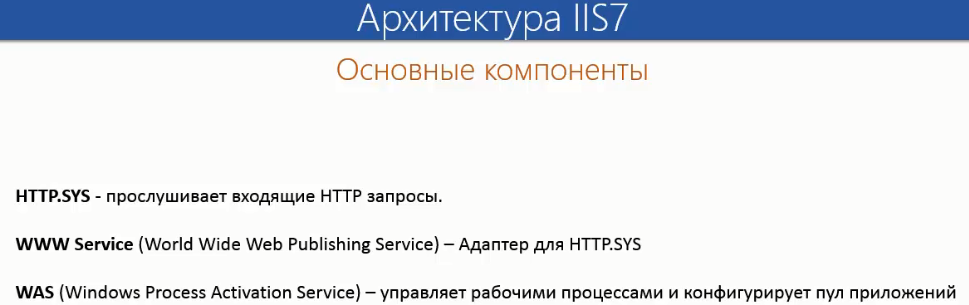
[ISS 7 Архитектура 2](#_Toc426482181)

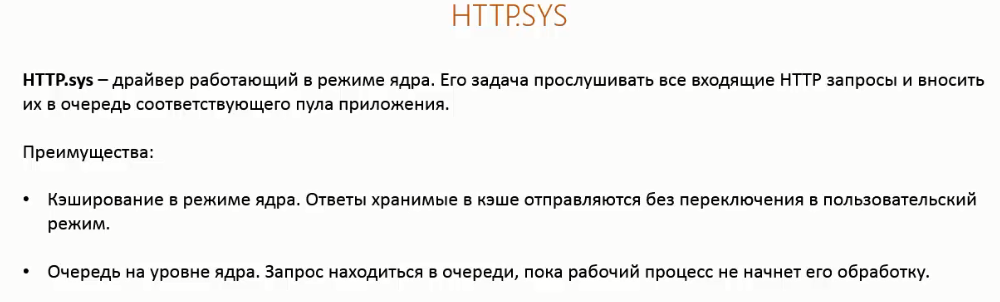
[Жизненный цикл HttpApllication 3](#_Toc426482182)

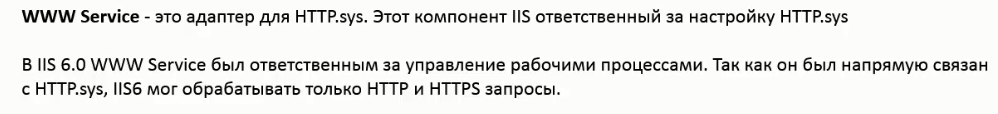
[Жизненный цикл 4](#_Toc426482183)

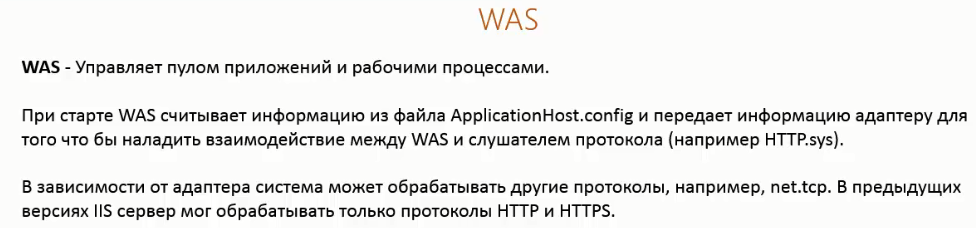
[Настройка IIS сервера 9](#_Toc426482184)

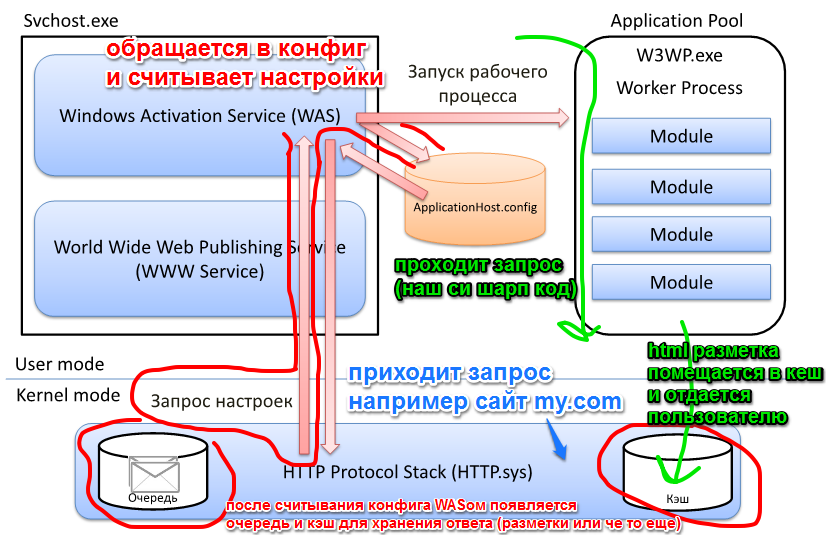
# ISS 7 Архитектура





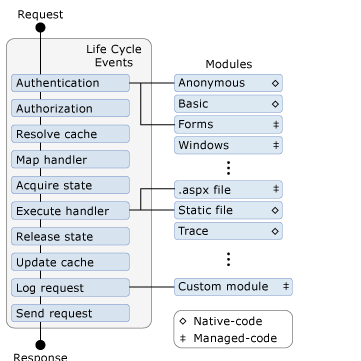




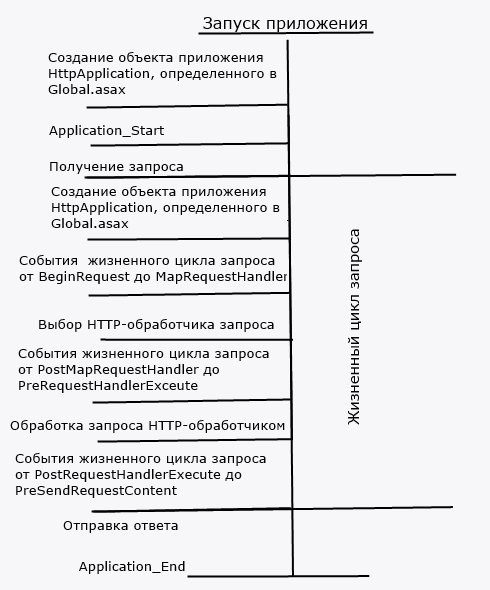


# Жизненный цикл HttpApllication

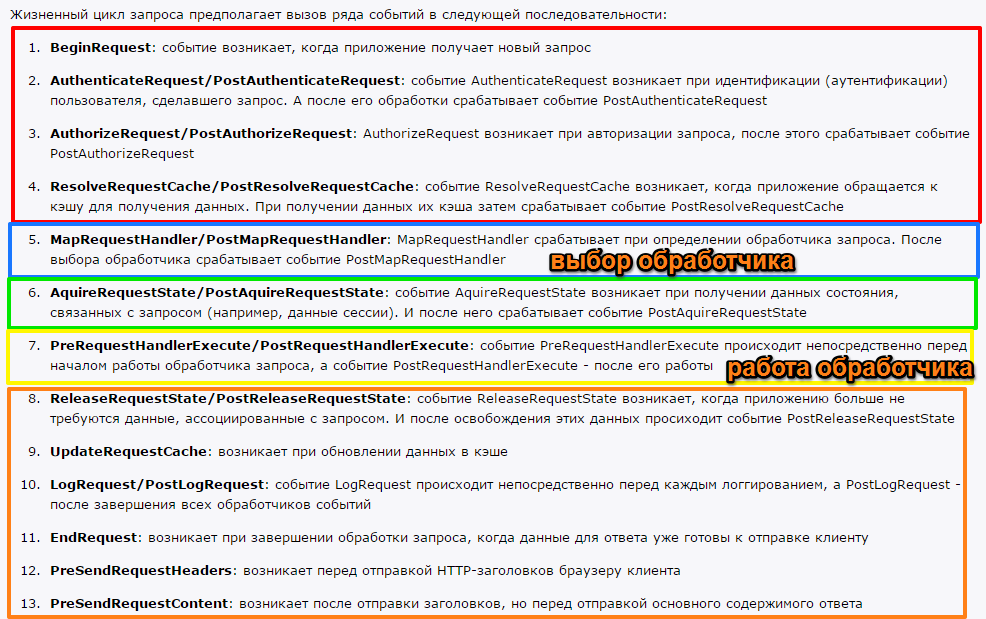


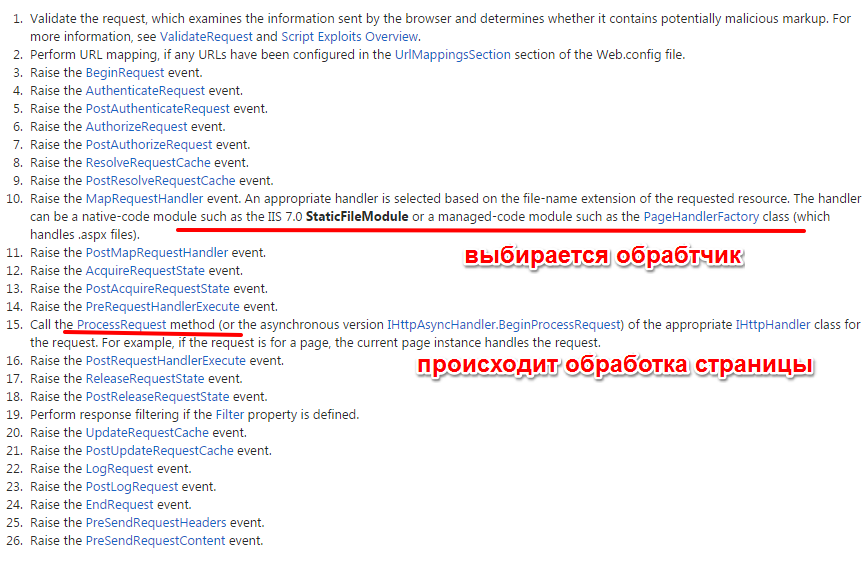


## Жизненный цикл



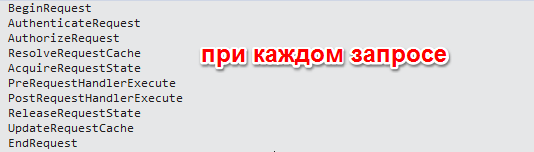
В IIS-7 26 событий



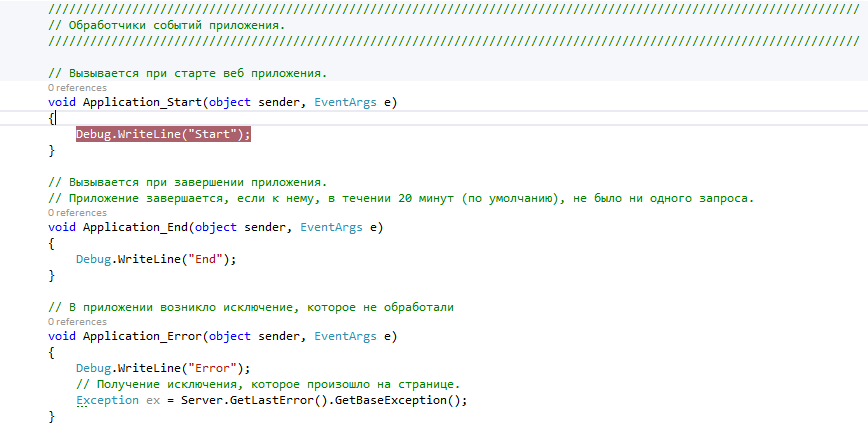




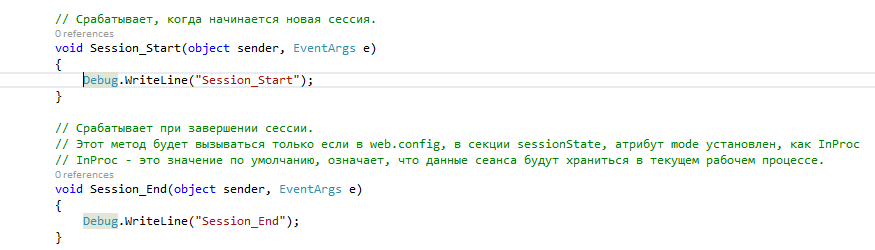
ОБРАТИТЬ ВНИМАНИЕ ЧТОНЕ ВСЕ СОБЫИЯ СРАБАТЫВАЮТ ПРИ КАЖДОМ ЗАПРОСЕ, А ТОЛЬКО ЛИШЬ ЭТИ



А эти единожды (если будет не отработанное исключение то его можно перехватить в событии Error)



Если нужно то можно отследить сколько пользователей на сайте через событие сессии



using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Web;

using System.Web.Security;

using System.Web.SessionState;

namespace ApplicationLifeCycle

{

public class Global : System.Web.HttpApplication

{

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Обработчики событий приложения.

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Вызывается при старте веб приложения.

void Application\_Start(object sender, EventArgs e)

{

Debug.WriteLine("Start");

}

// Вызывается при завершении приложения.

// Приложение завершается, если к нему, в течении 20 минут (по умолчанию), не было ни одного запроса.

void Application\_End(object sender, EventArgs e)

{

Debug.WriteLine("End");

}

// В приложении возникло исключение, которое не обработали

void Application\_Error(object sender, EventArgs e)

{

Debug.WriteLine("Error");

// Получение исключения, которое произошло на странице.

Exception ex = Server.GetLastError().GetBaseException();

}

// Срабатывает, когда начинается новая сессия.

void Session\_Start(object sender, EventArgs e)

{

Debug.WriteLine("Session\_Start");

}

// Срабатывает при завершении сессии.

// Этот метод будет вызываться только если в web.config, в секции sessionState, атрибут mode установлен, как InProc

// InProc - это значение по умолчанию, означает, что данные сеанса будут храниться в текущем рабочем процессе.

void Session\_End(object sender, EventArgs e)

{

Debug.WriteLine("Session\_End");

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Обработчики событий запроса. (В хронологическом порядке.)

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// Вызывается при каждом запросе.

protected void Application\_BeginRequest(object sender, EventArgs e)

{

Debug.WriteLine("BeginRequest");

}

// Срабатывает после аутентификации пользователя, выполняющего запрос. (аутентификация - определение принадлежности к ресурсу, поиск учетной записи)

void Application\_AuthenticateRequest(object sender, EventArgs e)

{

Debug.WriteLine("AuthenticateRequest");

}

// Срабатывает после авторизации пользователя, выполняющего запрос. (авторизация - определение наличия прав доступа)

void Application\_AuthorizeRequest(object sender, EventArgs e)

{

Debug.WriteLine("AuthorizeRequest");

}

// Модули отвечающие за кэширование, извлекают ранее обработанный запрос, минуя вызов обработчика события.

void Application\_ResolveRequestCache(object sender, EventArgs e)

{

Debug.WriteLine("ResolveRequestCache");

}

// Срабатывает после получения состояния сессии (Session state) для текущего запроса.

protected void Application\_AcquireRequestState(object sender, EventArgs e)

{

Debug.WriteLine("AcquireRequestState");

}

// Срабатывает до того как начнется вызов обработчика.

protected void Application\_PreRequestHandlerExecute(object sender, EventArgs e)

{

Debug.WriteLine("PreRequestHandlerExecute");

}

// ВЫЗОВ ОБРАБОТЧИКА.

// Срабатывает после завершения обработчика события.

protected void Application\_PostRequestHandlerExecute(object sender, EventArgs e)

{

Debug.WriteLine("PostRequestHandlerExecute");

}

// Вызывается тогда, когда информация сеанса сериализуется из коллекции Session, чтобы стать доступной в следующем запросе

protected void Application\_ReleaseRequestState(object sender, EventArgs e)

{

Debug.WriteLine("ReleaseRequestState");

}

// Вызывается перед добавлением информации в кэш выходных данных, если он был разрешен

protected void Application\_UpdateRequestCache(object sender, EventArgs e)

{

Debug.WriteLine("UpdateRequestCache");

}

// Последнее событие, которое происходит при обработке запроса.

protected void Application\_EndRequest(object sender, EventArgs e)

{

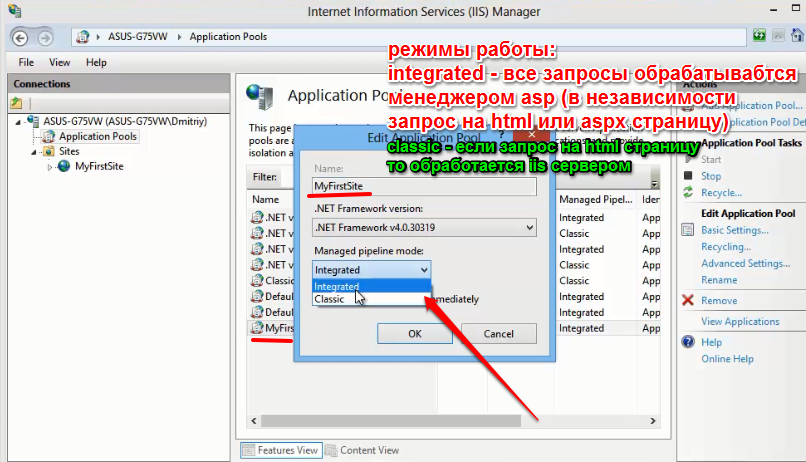
Debug.WriteLine("EndRequest");

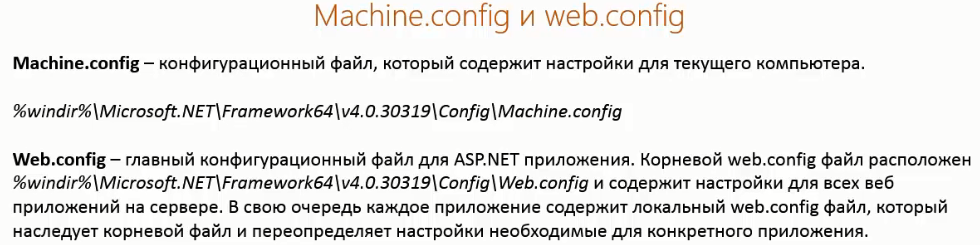
}

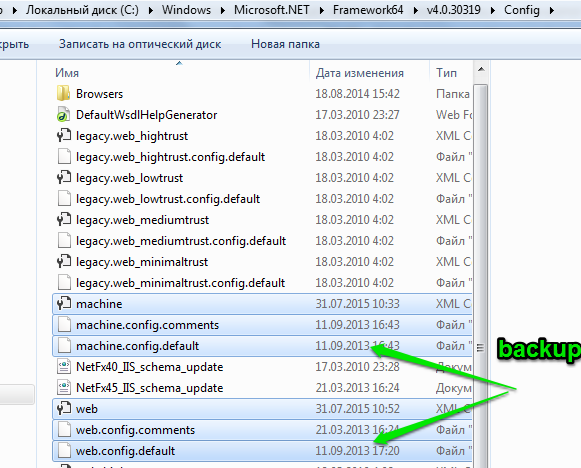
}

}

# Настройка IIS сервера







<customErrors mode="RemoteOnly"> только для пользователей который заходят с других компьютеров если хотим чтобы и для нас то mode="On"



