

ASP.NET 4.0 Beta1 - What's new in System.Web.Dll Assembly outlook

Introduction

This guideline provides detailed overview of System.Web assembly and the new classes and methods in ASP.NET 4.0, which were released with Visual Studio 2010 Beta1.

Document is subject to change – we are trying to find detailed information about all new stuff and update description column. The latest document version is available [there](#). Send us your comments and suggestion via [this online form](#).

Description

ASP.NET 4.0 introduces several changes to the web assemblies and also merges several dlls that existed in .NET 2.0 - 3.5 SP1. For example, new features from ASP.NET 3.5 SP1 System.Web.Abstractions.dll and System.Web.Routing.dll are merged with System.Web.dll, under System.Web namespace.

Changes

Beta1 of ASP.NET 4.0 introduced **27** new classes/interfaces/enums and **22** modifications of the existing ASP.NET 3.5 SP1 functionality.

Take into account, that Beta1 doesn't provide the complete list of the new features – they can add new stuff or remove existed one prior to the final release.

Legend

[black text] – existing items from version 2.0/3.0/3.5/SP1

[green text] – new added class, method, interface and etc

[gray text] – removed items, which existed before

SystemWeb.Dll

Assembly Name: System.Web, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a

System.Web namespace

```
public class HttpApplication : IHttpAsyncHandler,  
                                IHttpHandler, IComponent, IDisposable  
{  
    public virtual String GetOutputCacheProviderName(HttpContext context);  
}
```

```
[SerializableAttribute]  
public class HttpException : ExternalException  
{  
    public Int32 WebEventCode { get; }  
}
```

Description

ASP.NET 4.0 adds an extensibility point to output caching that enables you to configure one or more custom output-cache providers. Output-cache providers can use any storage mechanism to persist HTML content. This makes it possible to create custom output-cache providers for diverse persistence mechanisms, which can include local or remote disks, cloud storage, and distributed cache engines.

More details there:

<http://www.asp.net/learn/whitepapers/aspnet40/>

The *WebEventCodes* class contains codes that identify types of health-monitoring events. Two types of codes are defined within the class: major codes, which identify the ASP.NET health-monitoring events; and detail codes, which provide more information about a related major code. These codes are implemented as integers, rather than as an enumeration, to allow for extensibility.

NOTE: was sealed in .NET 3.5

<http://msdn.microsoft.com/en-us/library/system.web.management.webeventcodes.aspx>

System.Web.Caching namespace

<pre>[SerializableAttribute] public class FileResponseElement : ResponseElement { public FileResponseElement(String path, Int64 offset, Int64 length); public Int64 Length { get; } public Int64 Offset { get; } public String Path { get; } }</pre>	Extensible output cache related classes
<pre>[SerializableAttribute] public sealed class HeaderElement { public HeaderElement(String name, String value); public String Name { get; } public String Value { get; } }</pre>	n/a
<pre>public interface IOutputCacheEntry { List<HeaderElement> HeaderElements { get; set; } List<ResponseElement> ResponseElements { get; set; } }</pre>	n/a
<pre>[SerializableAttribute] public class MemoryResponseElement : ResponseElement { public MemoryResponseElement(Byte[] buffer, Int64 length); public Byte[] Buffer { get; } public Int64 Length { get; } }</pre>	
<pre>public static class OutputCache { public static OutputCacheProviderCollection Providers { get; } public static String DefaultProviderName { get; } public static Object Deserialize(Stream stream); public static void Serialize(Stream stream, Object data); }</pre>	Actual class of the output cache
<pre>public abstract class OutputCacheProvider : ProviderBase { protected OutputCacheProvider(); public abstract Object Add(String key, Object entry, DateTime utcExpiry); public abstract Object Get(String key); public abstract void Remove(String key); public abstract void Set(String key, Object entry, DateTime utcExpiry); }</pre>	Output cache provider
<pre>public sealed class OutputCacheProviderCollection : ProviderCollection, ICollection, IEnumerable { public OutputCacheProviderCollection(); public OutputCacheProvider this[String name] { get; } public override void Add(ProviderBase provider); public void CopyTo(OutputCacheProvider[] array, Int32 index); }</pre>	Extensible output cache related classes
<pre>[SerializableAttribute] public abstract class ResponseElement { protected ResponseElement(); }</pre>	Extensible output cache related classes
<pre>public sealed class SqlCacheDependency : CacheDependency, IDisposable { public static CacheDependency CreateOutputCacheDependency(String dependency); }</pre>	Extend existing class Note: SqlCacheDependency existed in 3.5SP1
<pre>[SerializableAttribute] public class SubstitutionResponseElement : ResponseElement { public SubstitutionResponseElement(HttpResponseSubstitutionCallback callback); public HttpResponseSubstitutionCallback Callback { get; } }</pre>	Extensible output cache related classes

New = 9
Modified = 1

System.Web.Compilation namespace

Description

```
public sealed class BuildManager
{
    public static FrameworkName TargetFramework { get; }
    public static IWebObjectFactory GetObjectFactory(String virtualPath,
                                                    Boolean throwIfNotFound);
}
```

Added new methods to compile for the specific platform

Note: [BuildManager](#) existed before

```
public sealed class ClientBuildManager : MarshalByRefObject, IDisposable
{
    public ClientBuildManager(String appVirtualDir,
                              String appPhysicalSourceDir,
                              String appPhysicalTargetDir,
                              ClientBuildManagerParameter parameter,
                              TypeDescriptionProvider typeDescriptionProvider);
}
```

Extended the existing class, providing additional constructs

Note: [ClientBuildManager](#) existed before

```
public class RouteUrlExpressionBuilder : ExpressionBuilder
{
    public RouteUrlExpressionBuilder();
    public override Boolean SupportsEvaluate { get; }
    public override Object EvaluateExpression(Object target,
                                             BoundPropertyEntry entry,
                                             Object parsedData,
                                             ExpressionBuilderContext context);
    public override CodeExpression GetCodeExpression(BoundPropertyEntry
                                                    entry,
                                                    Object parsedData, ExpressionBuilderContext context);
    public static String GetRouteUrl(Control control, String expression);
    public static Boolean TryParseRouteExpression(String expression,
                                                  RouteValueDictionary routeValues, out String routeName);
}
```

Built-in support for using Routing together with WebForms.

More details are there
<http://www.mostlylucid.net/archive/2009/01/25/asp.net-4.0-webform-routing-quick-rsquo-dirty-version.aspx>

```
public class RouteValueExpressionBuilder : ExpressionBuilder
{
    public RouteValueExpressionBuilder();
    public override Boolean SupportsEvaluate { get; }
    public override Object EvaluateExpression(Object target,
                                             BoundPropertyEntry entry,
                                             Object parsedData, ExpressionBuilderContext context);
    public override CodeExpression GetCodeExpression(BoundPropertyEntry
                                                    entry, Object parsedData, ExpressionBuilderContext context);
    public static Object GetRouteValue(Page page, String key, Type
                                       controlType, String propertyName);
}
```

Built-in support for using Routing together with WebForms.

More details are there
<http://www.mostlylucid.net/archive/2009/01/25/asp.net-4.0-webform-routing-quick-rsquo-dirty-version.aspx>

New = 2
Modified = 2

System.Web.Hosting namespace

Description

```
public class HostSecurityPolicyResolver
{
    public HostSecurityPolicyResolver();
    public virtual HostSecurityPolicyResults ResolvePolicy(Evidence
                                                         evidence);
}
```

n/a

```
public enum HostSecurityPolicyResults
{
    DefaultPolicy,
    FullTrust,
    AppDomainTrust,
    Nothing
}
```

n/a

```
public interface IApplicationPreloadManager
{
    void SetApplicationPreloadState(String context,
                                    String appId,
                                    Boolean enabled);
}
```

Auto-Start type.

More details there:
<http://www.asp.net/learn/whitepapers/aspne>

```

    void SetApplicationPreloadUtil(IApplicationPreloadUtil preloadUtil);
}

public interface IApplicationPreloadUtil
{
    void GetApplicationPreloadInfo(String context,
        out Boolean enabled, out String
        startupObjType,
        out String[] parametersForStartupObj);
    void ReportApplicationPreloadFailure(String context, Int32 errorCode,
        String errorMessage);
}

public interface IProcessHostPreloadClient
{
    void Preload(String[] parameters);
}

public sealed class ProcessHost : MarshalByRefObject, IProcessHost,
    IAdphManager,
    IPphManager,
    IProcessHostIdleAndHealthCheck,
    IApplicationPreloadManager
{
    public void SetApplicationPreloadState(String context, String appId,
        Boolean enabled);
    public void SetApplicationPreloadUtil(IApplicationPreloadUtil
        applicationPreloadUtil);
}

```

[t40/](#)

n/a

Managed Auto-Start type, which allows customize application auto-start. More details there: <http://www.asp.net/learn/whitepapers/aspnet40/>

ProcessHost has been extended to include methods, which control application auto-start

New = 5
Modified = 1

System.Web.Routing namespace

Description

```

public class PageRouteHandler : IRouteHandler
{
    public PageRouteHandler(String virtualPath);
    public PageRouteHandler(String virtualPath, Boolean
        checkPhysicalUrlAccess);
    public Boolean CheckPhysicalUrlAccess { get; }
    public String VirtualPath { get; }
    public virtual IHttpHandler GetHttpHandler(RequestContext requestContext);
    public String GetSubstitutedVirtualPath(RequestContext requestContext);
}

```

Routing support for a Web Forms Application [http://msdn.microsoft.com/en-us/library/system.web.routing.pageroutehandler\(VS.100\).aspx](http://msdn.microsoft.com/en-us/library/system.web.routing.pageroutehandler(VS.100).aspx)

Note: this namespace has been merged from ASP.NET 3.5 System.Web.Routing.dll

New = 1
Modified = 0

System.Web.Security namespace

Description

```

public sealed class FormsAuthentication
{
    public static TicketCompatibilityMode TicketCompatibilityMode { get; }
    public static TimeSpan Timeout { get; }
}

public abstract class MembershipProvider : ProviderBase
{
    protected virtual Byte[] EncryptPassword(Byte[] password,
        MembershipPasswordCompatibilityMode legacyPasswordCompatibilityMode);
}

[SerializableAttribute]
public sealed class RolePrincipal : IPrincipal, ISerializable
{
    protected RolePrincipal(SerializationInfo info,
        StreamingContext context);
    protected virtual void GetObjectData(SerializationInfo info,
        StreamingContext context);
}

```

No description available for the new methods yet

Added overloaded method to provide compatibility mode for the encrypted password
No more detail.

Represents security information for the current HTTP request, including role membership.
Not sealed class anymore, so can be inherited to be serialized to different contexts

New = 0
Modified = 3

System.Web.SessionState namespace

Description

```
public interface IPartialSessionState
{
    IList<String> PartialSessionStateKeys { get; }
}
```

Granular session update functionality, to provide distributed cache (came from Velocity project)

New = 1
Modified = 0

System.Web.UI namespace

Description

```
public enum ClientIDMode
{
    Inherit,
    Legacy,
    Predictable,
    Static
}
```

Provides different modes of ID for the client components, which might not depend on the actual container location and naming.

More details there

<http://cs.vbcity.com/blogs/mike-mcintyre/archive/2009/02/09/visual-studio-2010-web-development-clientidmode-and-clientid-dropdownlist-example.aspx>

```
public class Control : IComponent, IDisposable, IParserAccessor,
    IUrlResolutionService, IDataBindingsAccessor, IControlBuilderAccessor,
    IControlDesignerAccessor, IExpressionsAccessor
{
    public virtual ClientIDMode ClientIDMode { get; set; }
    public Control DataItemContainer { get; }
    public Control DataKeysContainer { get; }
    public virtual String AdapterGroup { get; set; }
    public virtual ViewStateMode ViewStateMode { get; set; }
    public String GetUniqueIDRelativeTo(Control control);
    protected void ClearCachedClientID();
    protected void ClearCachedClientIDMode();
    protected void ClearChildAdapter();
    protected void ClearChildAdapterGroup();
}
```

Additional methods that provide new cachability functionality and item containers features.

```
public sealed class ControlCachePolicy
{
    public String ProviderName { get; set; }
}
```

Providers for the control policies

```
public sealed class DataKeyPropertyAttribute : Attribute
{
    public DataKeyPropertyAttribute(String name);
    public String Name { get; }
    public override Boolean Equals(Object obj);
    public override Int32 GetHashCode();
}
```

n/a

```
public interface IDataKeysControl
{
    DataKeyArray ClientIDRowSuffixDataKeys { get; }
    String[] ClientIDRowSuffix { get; }
}
```

```
public class Page : TemplateControl, IHttpHandler
{
    public RouteData RouteData { get; }
    public String MetaDescription { get; set; }
    public String MetaKeywords { get; set; }
}
```

Routing data and META keywords support for web page.

```
public sealed class PartialCachingAttribute : Attribute
{
    public String ProviderName { get; set; }
}
```

Specifying provider for cache via attributes

```
public class StaticPartialCachingControl : BasePartialCachingControl
{
    public StaticPartialCachingControl(String ctrlID, String guid, Int32
```

n/a

```

duration, String varyByParams, String varyByControls, String varyByCustom,
String sqlDependency, BuildMethod buildMethod, String providerName);
    public static void BuildCachedControl(Control parent, String ctrlID,
String guid, Int32 duration, String varyByParams, String varyByControls, String
varyByCustom, String sqlDependency, BuildMethod buildMethod, String
providerName);
}

```

```

public enum ViewStateMode
{
    Inherit,
    Enabled,
    Disabled
}

```

New modes of ViewState

```

New      = 4
Modified = 5

```

System.Web.UI.HtmlControls namespace

Description

```

public sealed class HtmlHead : HtmlGenericControl
{
    public String Description { get; set; }
    public String Keywords { get; set; }
}

```

Head control has Description and Keywords fields

```

New      = 0
Modified = 1

```

System.Web.UI.WebControls.WebParts namespace

Description

```

public class WebPartManager : Control, INamingContainer, IPersonalizable
{
    protected virtual PermissionSet MediumPermissionSet { get; }
    protected virtual PermissionSet MinimalPermissionSet { get; }
}

```

n/a

```

New      = 0
Modified = 1

```

System.Web.UI.WebControls namespace

Description

```

public class DataKey : IStateManager, IEquatable<DataKey>
{
    public Boolean Equals(DataKey other);
}

```

Extended class to provide equality of instances

```

public class DetailsView : CompositeDataBoundControl, IDataItemContainer,
    INamingContainer, ICallbackContainer, ICallbackEventHandler,
    IPostBackEventHandler, IPostBackContainer, IDataBoundItemControl,
    IDataBoundControl, IFieldControl
{
    String[] IDataBoundControl.get_DataKeyNames();
    String IDataBoundControl.get_DataMember();
    Object IDataBoundControl.get_DataSource();
    String IDataBoundControl.get_DataSourceID();
    IDataSource IDataBoundControl.get_DataSourceObject();
    void IDataBoundControl.set_DataKeyNames(String[] value);
    void IDataBoundControl.set_DataMember(String value);
    void IDataBoundControl.set_DataSource(Object value);
    void IDataBoundControl.set_DataSourceID(String value);
    DataKey IDataBoundItemControl.get_DataKey();
    DataBoundControlMode IDataBoundItemControl.get_Mode();
    IAutoFieldGenerator IFieldControl.get_FieldsGenerator();
    void IFieldControl.set_FieldsGenerator(IAutoFieldGenerator value);
}

```

n/a

```

public class FormView : CompositeDataBoundControl, IDataItemContainer,
    INamingContainer, IPostBackEventHandler, IPostBackContainer,
    IDataBoundItemControl, IDataBoundControl
{
    public virtual Boolean RenderTable { get; set; }
}

```

n/a

```

String[] IDataBoundControl.get_DataKeyNames();
String IDataBoundControl.get_DataMember();
Object IDataBoundControl.get_DataSource();
String IDataBoundControl.get_DataSourceID();
IDataSource IDataBoundControl.get_DataSourceObject();
void IDataBoundControl.set_DataKeyNames(String[] value);
void IDataBoundControl.set_DataMember(String value);
void IDataBoundControl.set_DataSource(Object value);
void IDataBoundControl.set_DataSourceID(String value);
DataKey IDataBoundItemControl.get_DataKey();
DataBoundControlMode IDataBoundItemControl.get_Mode();
}

public class FormViewRow : TableRow
{
    protected|internal override void Render(HtmlTextWriter writer);
}

public class GridView : CompositeDataBoundControl, IPostBackContainer,
IPostBackEventHandler, ICallbackContainer, ICallbackEventHandler,
IPersistedSelector, IDataKeysControl, IDataBoundListControl, IDataBoundControl,
IFieldControl {
    public virtual Boolean EnablePersistedSelection { get; set; }
    public virtual Boolean ShowHeaderWhenEmpty { get; set; }
    public DataKeyArray ClientIDRowSuffixDataKeys { get; }
    public virtual String[] ClientIDRowSuffix { get; set; }
    public TableItemStyle SortedAscendingCellStyle { get; }
    public TableItemStyle SortedAscendingHeaderStyle { get; }
    public TableItemStyle SortedDescendingCellStyle { get; }
    public TableItemStyle SortedDescendingHeaderStyle { get; }

    DataKeyArray IDataKeysControl.get_ClientIDRowSuffixDataKeys();
    String[] IDataBoundControl.get_DataKeyNames();
    String IDataBoundControl.get_DataMember();
    Object IDataBoundControl.get_DataSource();
    String IDataBoundControl.get_DataSourceID();
    IDataSource IDataBoundControl.get_DataSourceObject();
    void IDataBoundControl.set_DataKeyNames(String[] value);
    void IDataBoundControl.set_DataMember(String value);
    void IDataBoundControl.set_DataSource(Object value);
    void IDataBoundControl.set_DataSourceID(String value);
    String[] IDataBoundListControl.get_ClientIDRowSuffix();
    DataKeyArray IDataBoundListControl.get_DataKeys();
    Boolean IDataBoundListControl.get_EnablePersistedSelection();
    DataKey IDataBoundListControl.get_SelectedDataKey();
    Int32 IDataBoundListControl.get_SelectedIndex();
    void IDataBoundListControl.set_ClientIDRowSuffix(String[] value);
    void IDataBoundListControl.set_EnablePersistedSelection(Boolean value);
    void IDataBoundListControl.set_SelectedIndex(Int32 value);
    IAutoFieldGenerator IFieldControl.get_FieldsGenerator();
    void IFieldControl.set_FieldsGenerator(IAutoFieldGenerator value);
}

public interface IDataBoundControl
{
    IDataSource DataSourceObject { get; }
    Object DataSource { get; set; }
    String DataMember { get; set; }
    String DataSourceID { get; set; }
    String[] DataKeyNames { get; set; }
}

public interface IDataBoundItemControl : IDataBoundControl
{
    DataBoundControlMode Mode { get; }
    DataKey DataKey { get; }
}

public interface IDataBoundListControl : IDataBoundControl
{
    Boolean EnablePersistedSelection { get; set; }
    DataKey SelectedDataKey { get; }
    DataKeyArray DataKeys { get; }
    Int32 SelectedIndex { get; set; }
    String[] ClientIDRowSuffix { get; set; }
}

public interface IFieldControl
{
    IAutoFieldGenerator FieldsGenerator { get; set; }
}

```

```
public class RouteParameter : Parameter, ICloneable, IStateManager
{
    public RouteParameter();
    public RouteParameter(String name, String routeKey);
    public RouteParameter(String name, TypeCode type, String routeKey);
    public RouteParameter(String name, DbType dbType, String routeKey);
    protected RouteParameter(RouteParameter original);
    public String RouteKey { get; set; }

    protected override Parameter Clone();
    protected|internal override Object Evaluate(HttpContext context, Control
        control);
}

public class XmlDataSource : HierarchicalDataSourceControl, IDataSource,
    IListSource
{
    public virtual String CacheKeyContext { get; set; }
}
```

New = 5
Modified = 6

The detailed overview of all changes from ASP.NET team can be found there
<http://www.asp.net/learn/whitepapers/aspnet40/>