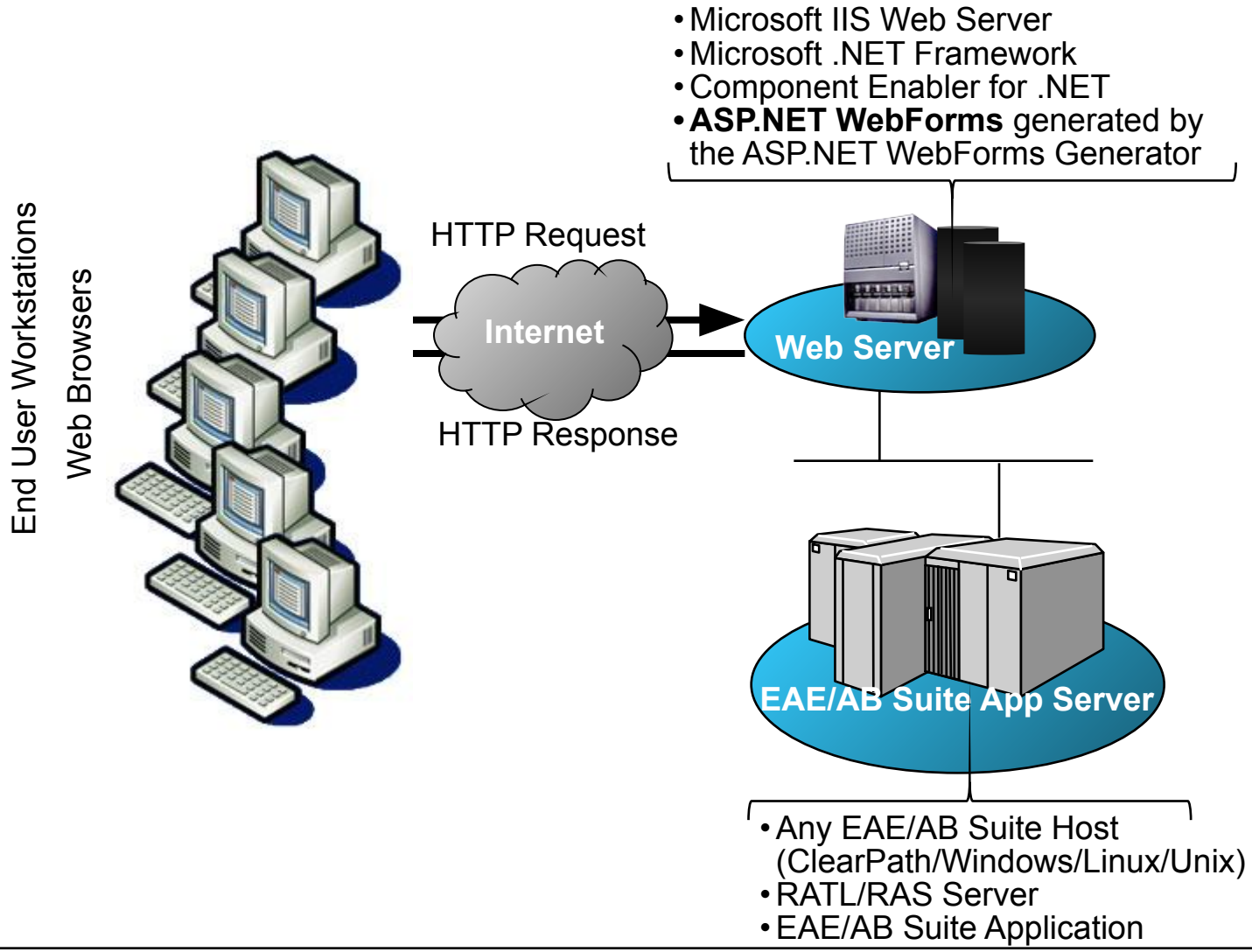
Component Enabler for .NET

ASP.NET WebForms Generator

Overview

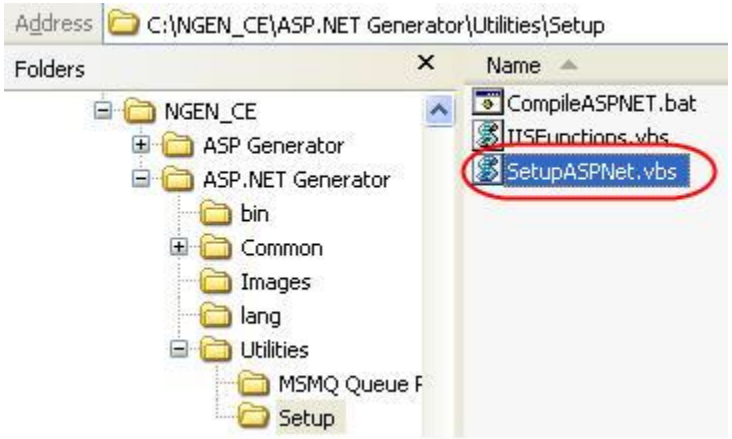
- ASP.NET WebForms environment
- ASP.NET WebForms Generator overview
- ASP.NET WebForms Generator setup
- Generating an ASP.NET WebForms bundle
- WebForm Renderer
- Customizing a generated ASP.NET WebForms client
- Renderer Control events
- Extending generated forms
- Using multiple Renderer Controls

ASP.NET WebForms Environment



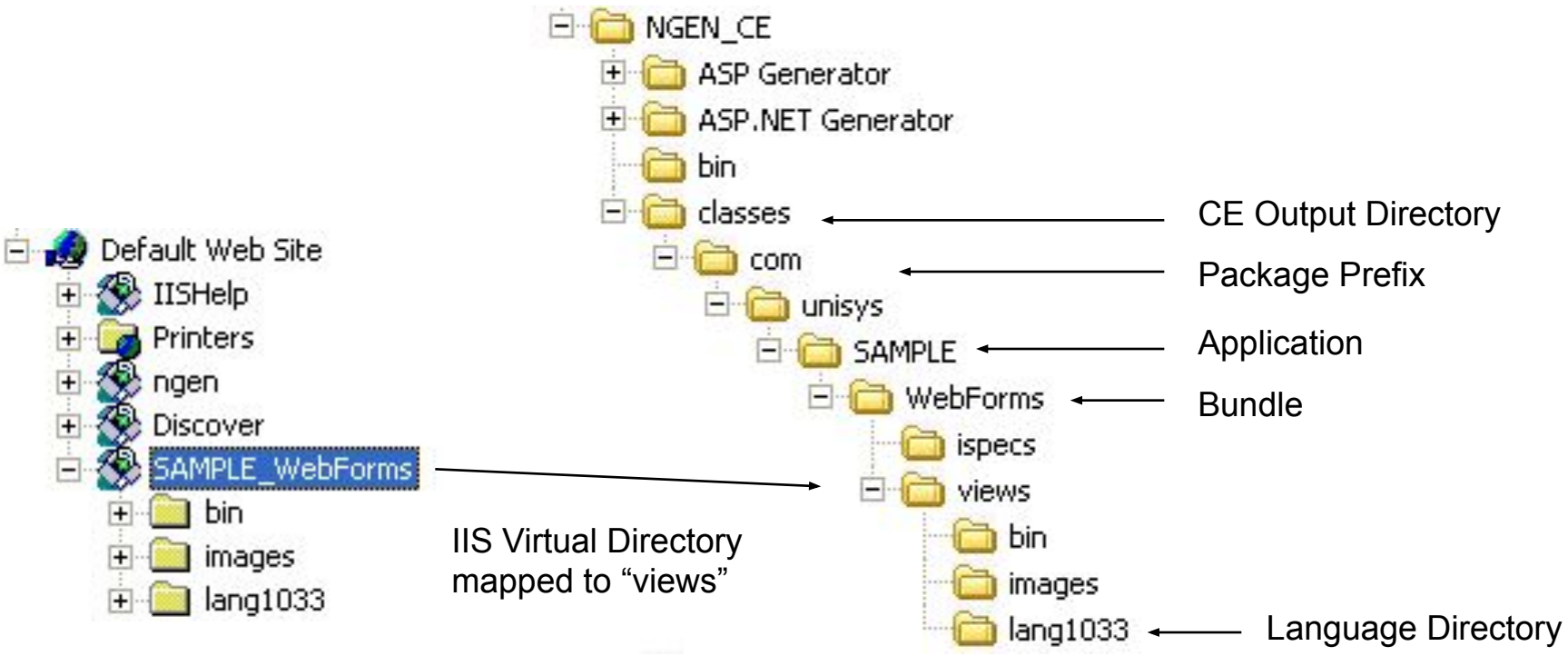
ASP.NET Generator Setup

- Run the ASP.NET WebForm Generator Setup wizard:
  <CE installation directory>\ASP.NET Generator\Utilities\Setup\SetupASPNet.vbs
- The wizard will prompt you for:
  - Output directory
  - Package prefix
  - Application name
  - Bundle name
  - Number of languages for the system
  - Locale identifiers (optional)
  - Virtual directory to create
- Wizard creates the directory structure for your ASP.NET Web application, copies the required infrastructure files, and registers the virtual directory



ASP.NET Generator Setup Wizard Output

Wizard creates the directory structure for your ASP.NET Web application, copies the required infrastructure files, and registers the virtual directory.



Generating an ASP.NET WebForms Bundle

Sample Bundle Configuration Properties

Category	Property	Value
Build Target Filter	Deploy Component Enabler User Interface	True
Component Enabler User Interface	Generate Views	True
	User Defined View Generator	GenerateFormASPdotNET.dll
	Ispec Models Source Language	C#
General	Deployable	True
Installation	Package Name	WebForms

Configuring the ASP.NET WebForms Application

1. Open <Views>\Web.config in Visual Studio (or any text editor).
2. Supply host details:
   - HostURI
   - HostViewName
   - HostLogin, HostPassword, and HostDomain
3. (Optional) Modify application settings, such as:
   - Logging
   - Object pooling
   - Security
4. Save and close the file.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

 <appSettings>
    <add key="DefaultSystem"            value="" />

    <add key="LINCEnvironmentProgId"    value="" />

    <add key="ApplicationName"          value="SAMPLE" />
    <add key="BundleName"               value="WebForms" />
    <add key="DisplayName"              value="SAMPLE" />
    <add key="PackagePrefix"            value="com.unisys" />

    <add key="HostURI"                  value="x-ratl:localhost:2449" />
    <add key="HostViewName"             value="SAMPLE" />
     …
    <add key="HostLogin"                value="ALPublic" />
    <add key="HostPassword"             value="ALPublic" />
    <add key="HostDomain"               value="." />
     …
    <add key="LoggingEnabled"           value="true" />
    <add key="LogFile"                  value="C:\Temp\ASPNet.log" />
    <add key="LogLevel"                 value="7" />
     …
 </appSettings>
```

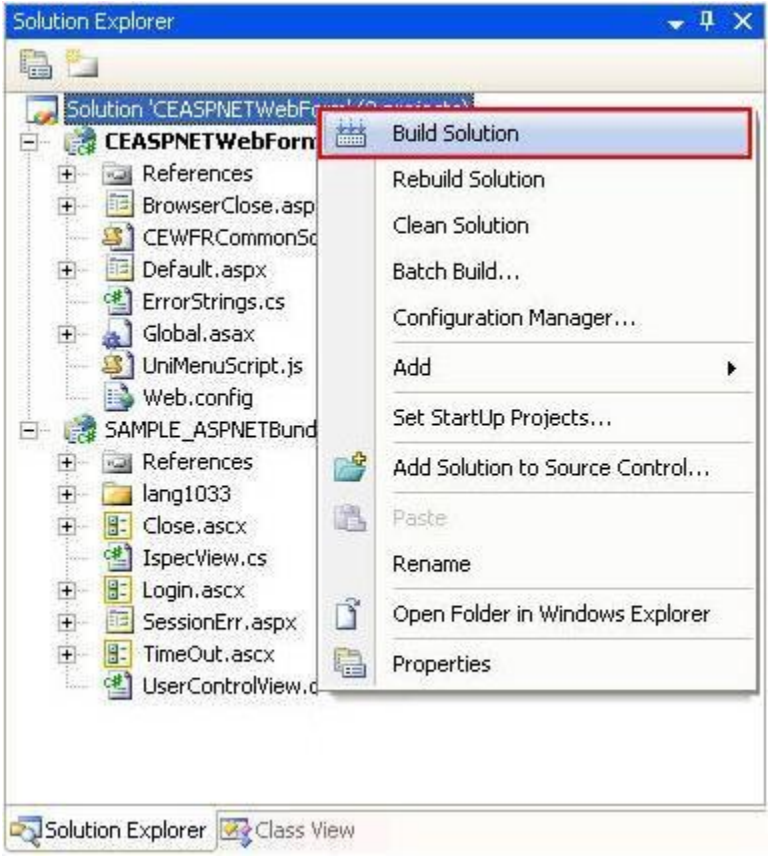
Building the WebForms Application

To build the default ASP.NET WebForms application:

Run Views\CompileASPNET.bat

OR

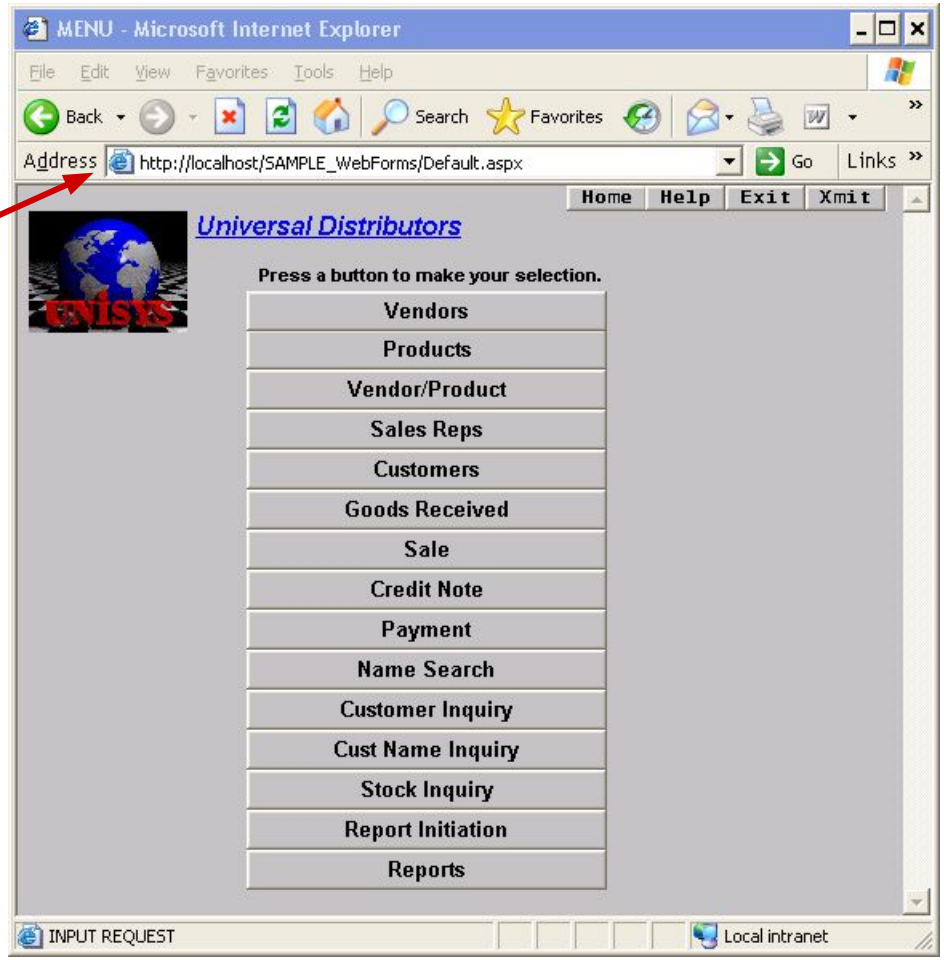
1. In Visual studio, open CEASPNETWebForm.sln.
2. In the Solution Explorer, right-click the Solution and select Build.



Default ASP.NET WebForms Application

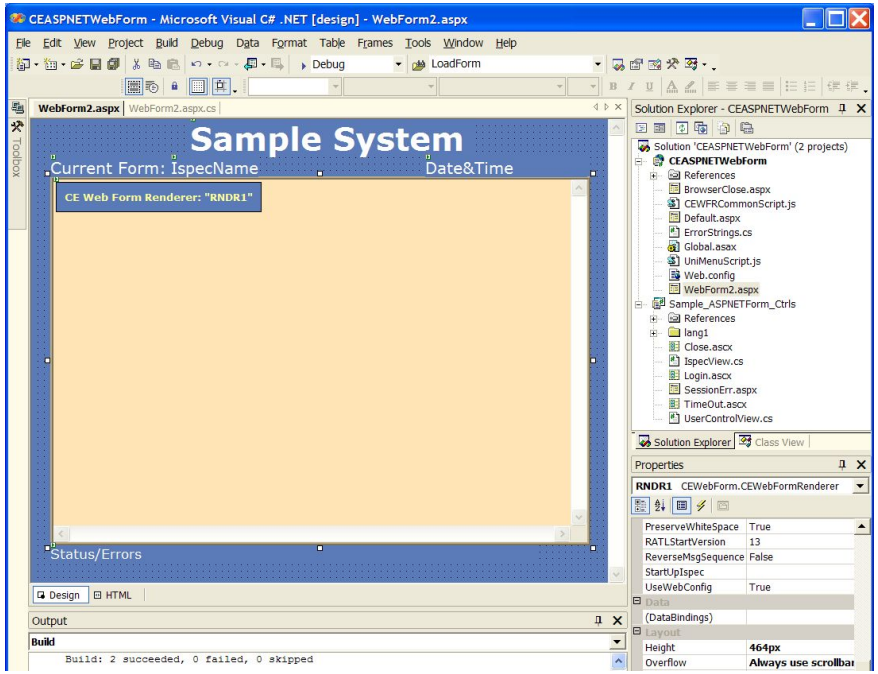
Access your generated ASP.NET WebForms application via a browser.

http://<machine name>/
<virtual directory>/Default.aspx



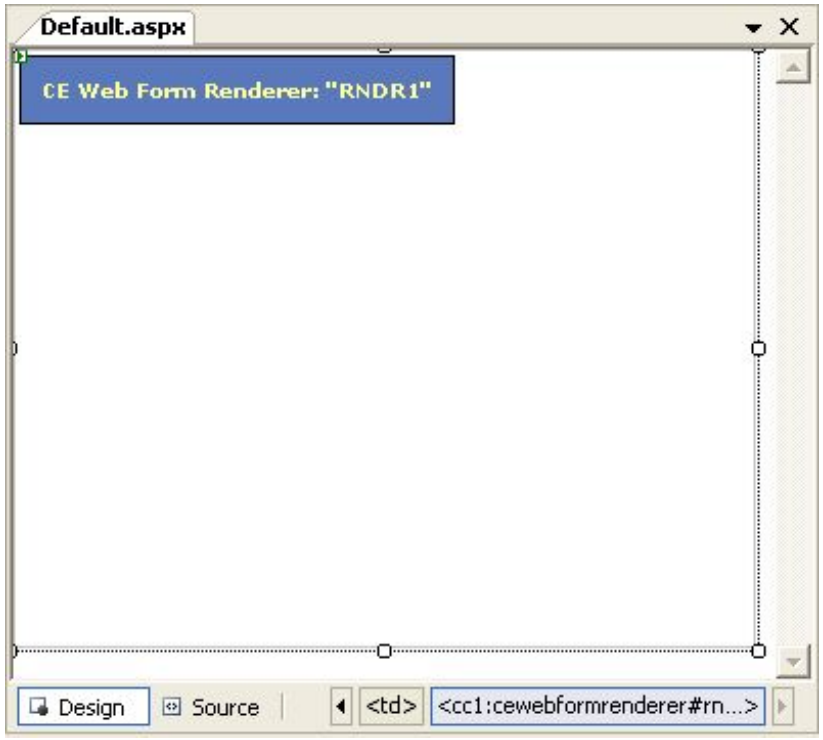
Customizing your Generated Application

- Customize look and feel
  - Add controls to WebForm
  - Modify display properties
- Customize behavior
  - Perform client-side validation
  - Set/modify field values
  - Customize error handling
  - Dynamically modify display properties of controls
- Extend generated ispec forms
- Display multiple ispec forms on a single WebForm
  - Multiple views of your EAE/AB Suite Runtime system
  - Multiple EAE/AB Suite Runtime systems



The CE WebForm Renderer

- Component of the Client Tools ASP.NET WebForms Generator
- ASP.NET Server Control
- Specifically for rendering forms in a Web browser interacting with EAE or AB Suite systems
- Container for displaying ispecs
- Handles communication with host application



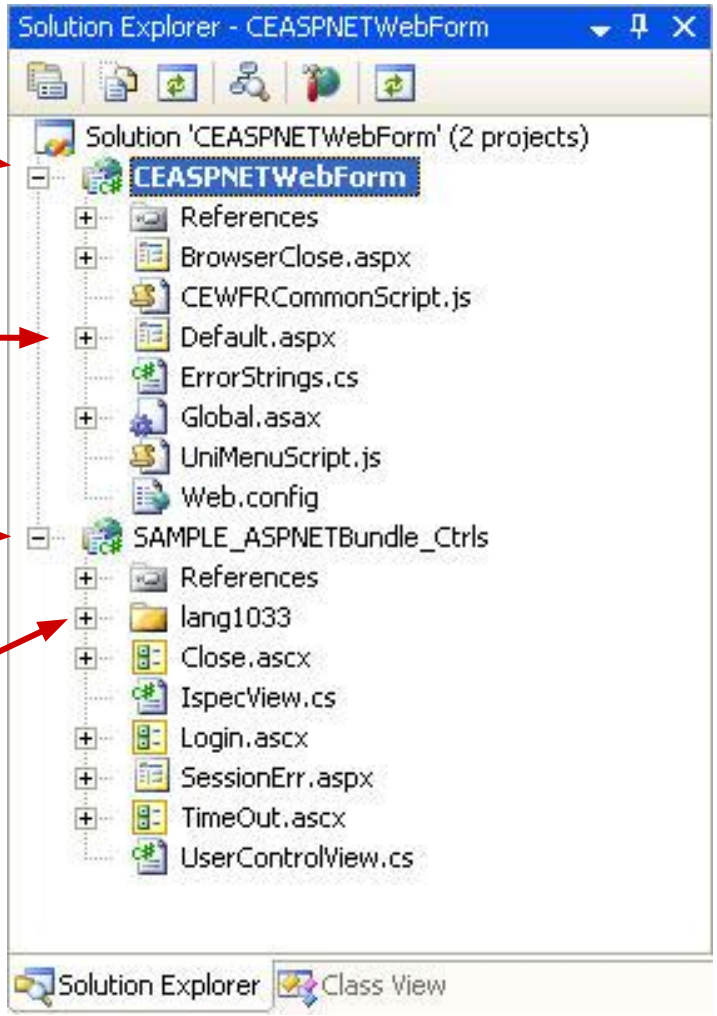
CEASPNETWebForm Solution

CEASPNetWebForm project
Files required for default ASP.NET Web Application

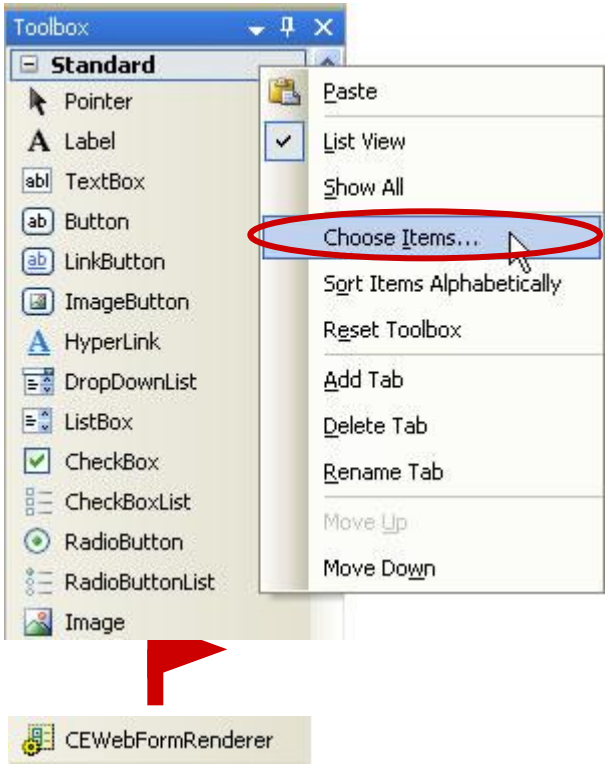
Default.aspx
Hosts WebForm Renderer

[Virtual directory]_Ctrls project
Generated User Controls

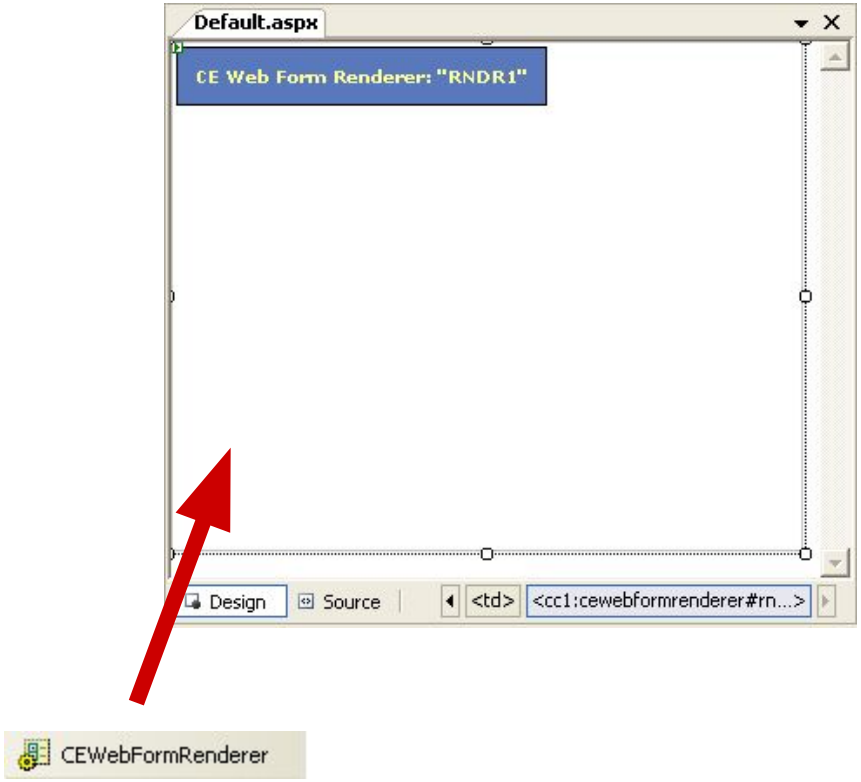
lang1-n directories
Generated ispec User Controls



Using the CE WebForm Renderer Control

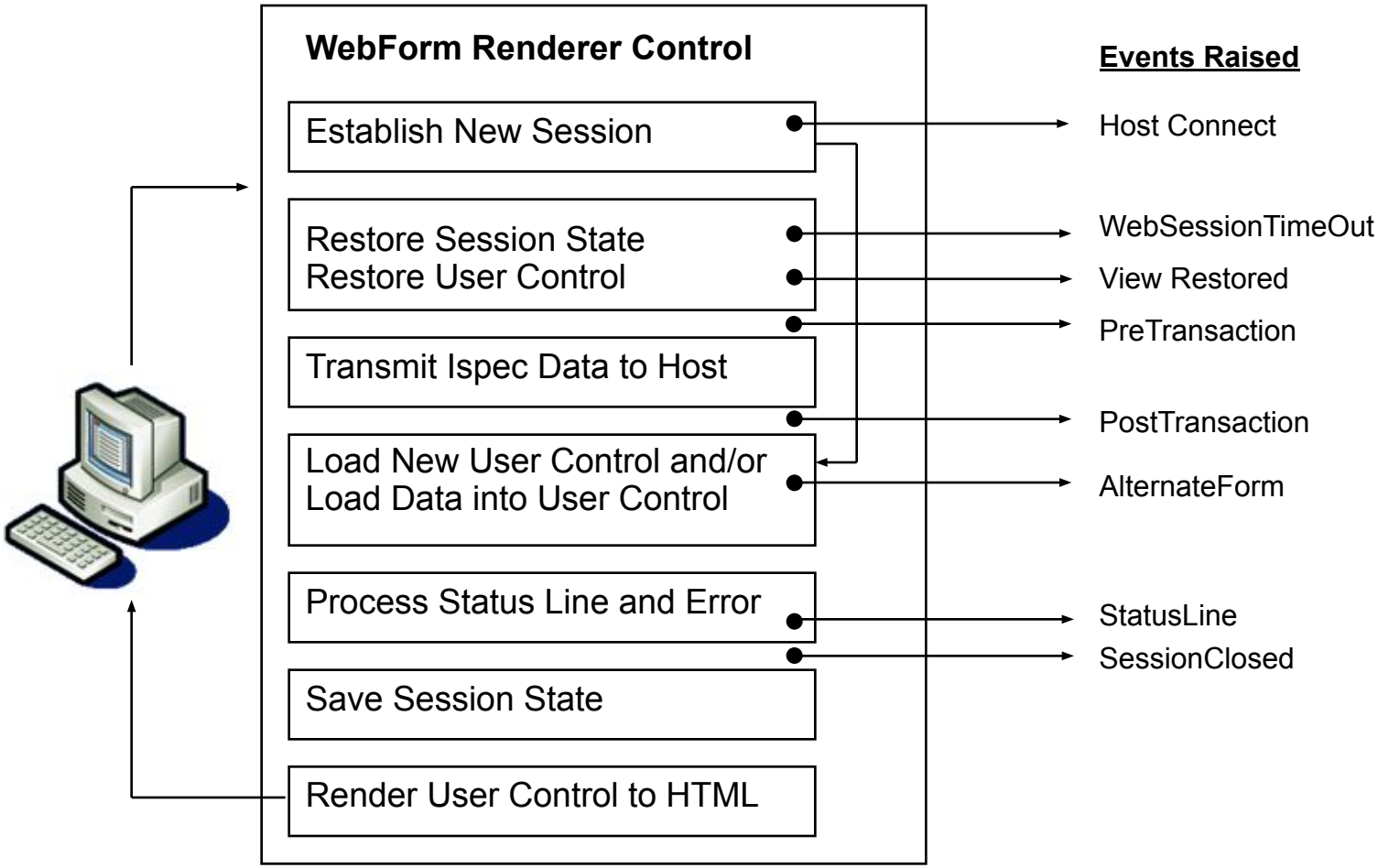


Add the CE WebForm Renderer Control to the Toolbox



Drag and drop the CE WebForm Renderer Control from the Toolbox onto a WebForm

Renderer Control Events



CE WebForm Renderer Event Argument Object

- CEWebFormRendererEventArgs — class with a collection of objects that is passed to each event
- Important members include:
  - HostSession — the LINCEnvironment object used by the Renderer Control for maintaining the session with the Runtime host
  - WebSession — the session object used by the Renderer Control for maintaining the web session
  - ReturnCode — a value containing the most recent ResponseCode from the host
  - Status — LINCStatus line object for the most recent transaction
- Not all properties of the CEWebFormRendererEventArgs are applicable for every type of event

PreTransaction Event

- Raised after form data transferred to ispec model
- Opportunity to validate or modify data before sending transaction to host
- Add logic in PreTransaction event handler to:
  - Perform extra client-side validation of user input
  - Auto fill fields
  - Modify field values
  - Bypass transaction and return error message

PreTransaction Event Handler — Example

```
protected void RNDR1_PostTransaction(object sender, CEWebFormRendererEventArgs e)
{
    // Obtain the name of the current ispec from the HostSession object
    CEWindowsAPI.IspecModel currIspec = e.HostSession.GetCurrentIspec();

    // if the current ispec is "SREP" and UserMaint = "ADD" or "CHG"
    if (currIspec.IspecModelName == "SREP")
    {
      string maint = currIspec.GetFieldValue("_UserMAINT");
      if (maint == "ADD" || maint == "CHG")
      {
            //Bypass the transaction and set the ReturnCode
            e.BypassTransaction = true;
            e.ReturnCode =
    CEWindowsAPI.ResponseCodes.ERR_TRANSACTION_FAILED;
      }
    }
}
```

PostTransaction Event

- Raised after host runtime has processed transaction and returned ispec
- Opportunity to process response from host runtime
- Add logic in PostTransaction event handler to:
  - Change field values
  - Pre-fill fields
  - Display own custom Web User Control form

PostTransaction Event Handler — Example

```
protected void RNDR1_PostTransaction(object sender, CEWebFormRendererEventArgs e)
{
    // Obtain the name of the current ispec from the HostSession object
    CEWindowsAPI.IspecModel currIspec = e.HostSession.GetCurrentIspec();

    // If the current ispec is "CUST" and the "CUSTOMER" field is blank
    if (currIspec.IspecModelName == "CUST")
    {
      if (currIspec.GetFieldValue("CUSTOMER") != "")
      {
            // Prevent users from seeing the postal address of the customer.
            currIspec.SetFieldValue("POSTADD1", "Not Available");
            currIspec.SetFieldValue("POSTADD2", "Not Available");
            currIspec.SetFieldValue("POSTADD3", "Not Available");
      }
    }
}
```

StatusLine Event

- Raised immediately before form is sent to browser
- Opportunity to:
  - Customize display of transaction status
  - Influence look and feel of form
- Add logic in StatusLine event handler to:
  - Display transaction status/error messages
  - Change display attributes of form or controls

StatusLine Event Handler — Example

```
protected void RNDR1_StatusLine(object sender, CEWebFormRendererEventArgs e)
{
      // Display name of current Ispec.
      Lbl_IspecName.Text = "";
      if (e.HostSession != null)
      {
        CEWindowsAPI.IspecModel currIspec = e.HostSession.GetCurrentIspec();
        if (currIspec != null)
              Lbl_IspecName.Text = currIspec.IspecModelName;
      }
}
```

AlternateForm Event

- Raised when a new form is about to be loaded
- Opportunity to specify an alternate form to load
- Add logic in AlternateForm event handler to:
  - Check if a particular form is about to be loaded
  - Display an alternate form
- Provides a mechanism for displaying extended forms

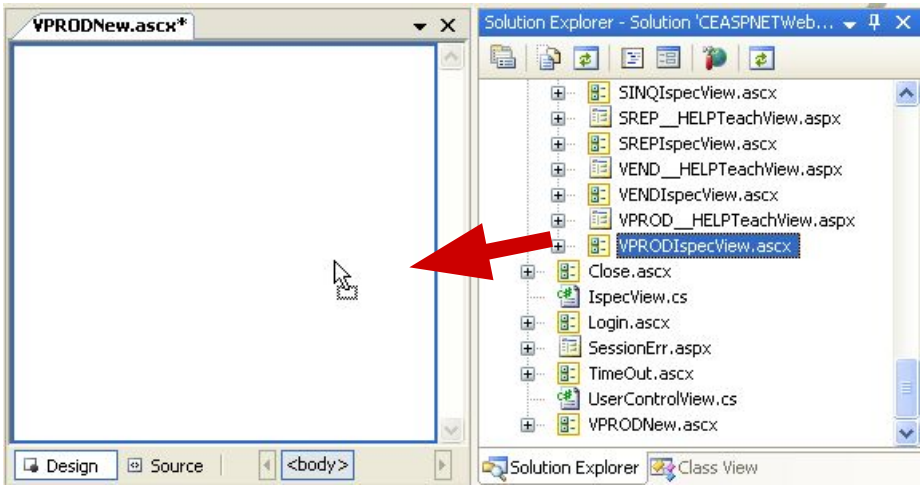
AlternateForm Event Handler — Example

```
protected void RNDR1_AlternateForm(object sender, CEWebFormRendererEventArgs e)
{
    // Get current language number
    int langNo = e.WebSession.LangNo;

    string formNameToAlter = "lang" + langNo + "/VPRODIspecView.ascx";

    // Check the form about to be loaded.
    if (e.FormName == formNameToAlter)
    {
      // Specify an alternate form.
      e.FormName = "lang" + langNo + "/VPRODNew.ascx";
    }
}
```

Special Case Events

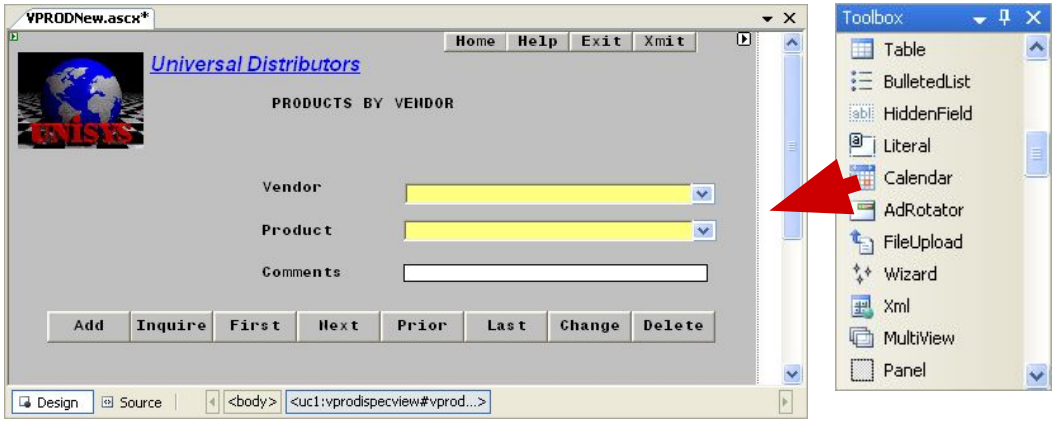
- WebSessionTimeOut Event
  - Raised when Web session with end user times out
- SessionClosedEvent
  - Raised when host runtime application closes session
- ViewRestored Event
  - Raised when form returned by browser does not match last form sent to it
- HostConnect Event
  - Raised when Renderer Control acquires HostSession object from object pool
  - Only applies if you enable object pooling of the HostSession object

Extending Generated Forms



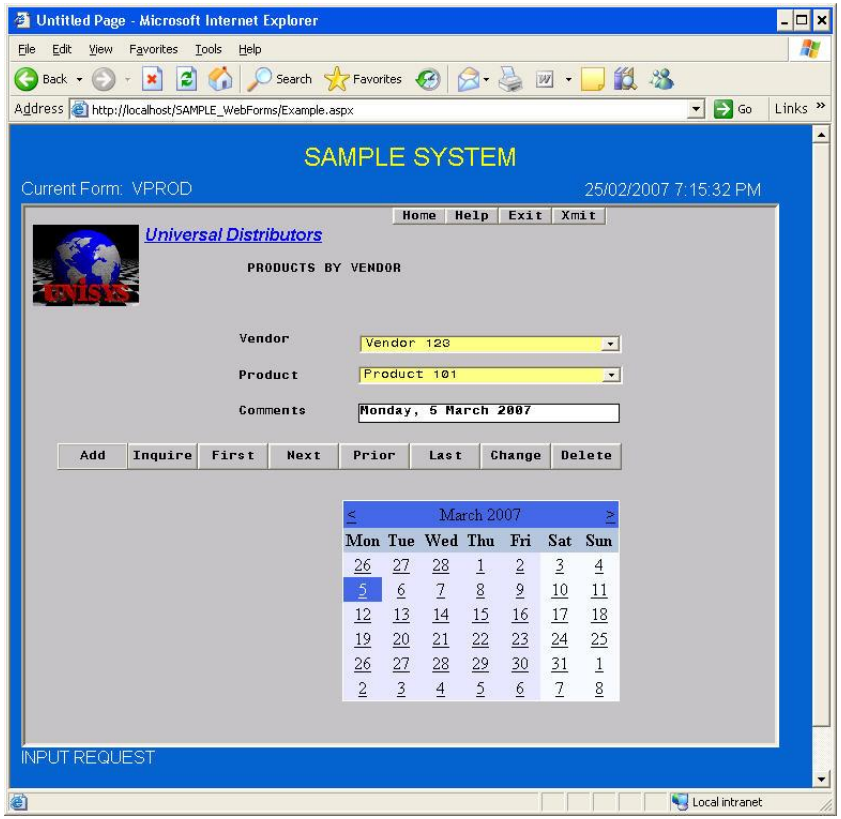
Drag and drop generated control onto Design surface of new Web User Control

Drag and drop additional controls onto the new Web User Control



Extending Generated Forms — Example

1. Add a new Web User Control called NewVPROD.ascx
2. Drop the generated VPROD.ascx Control onto NewVPROD.ascx
3. Drop a Calendar Control onto NewVPROD.asxc
4. Add a SelectionChanged event handler to the Calendar Control
5. Add an AlternateForm event handler to the WebForm

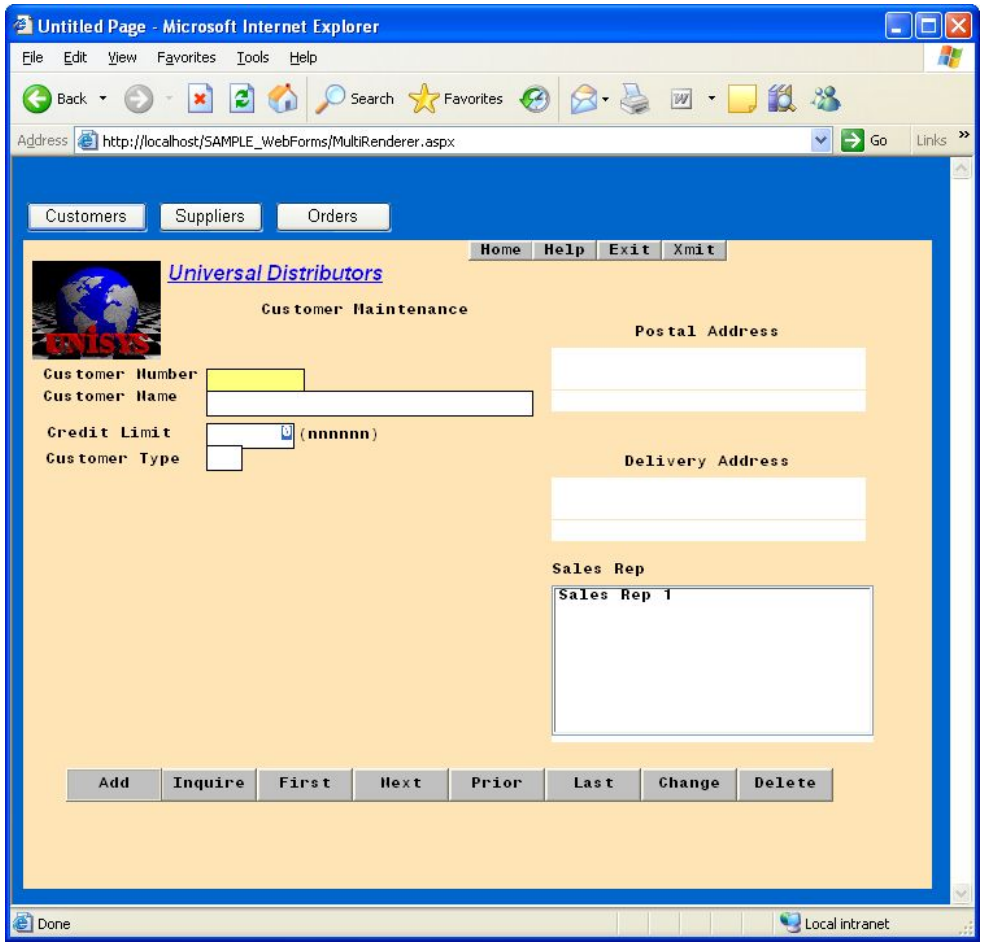


Using Multiple WebForm Renderer Controls in an Application

- Two or more Renderer Controls on the same ASP.NET Web page
- Renderer Controls are independent of each other
- Each Renderer Control:
  - Establishes a separate connection with the host Runtime application
  - Maintains its own session state with the browser user and with the host
- May be configured to access different host Runtime applications or different functional areas of the same application

Multiple Renderer Controls (One Runtime System)

1. Generate system and configure Web application (as for default application)
2. Add new WebForm to CEASPNETWebForm project
3. Add Renderer Controls to WebForm
4. Add logic to show / hide Renderers as required



Accessing Multiple Systems from a Web Application

Overview of steps:

1. Generate bundles for each application
2. In Visual Studio, create an ASP.NET Web Application with multiple Renderer Controls
3. Create a virtual directory mapped to the project directory of the new Web application
4. Copy files from generated bundles to virtual directory of Multi Renderer Web Application
5. Configure the Multi Renderer Web Application (Web.config)

Summary

- ASP.NET WebForms
  - Event-driven OO programming model
  - Server Controls
  - Custom Controls
  - Cross-browser support
- ASP.NET WebForms Generator
  - Generates an ASP.NET WebForms project
  - Includes CE WebForms Renderer Server Control
  - Provides better customization capabilities than traditional ASP Generator
  - Less need for Generator Customization Kit
- CE WebForms Renderer
  - Specifically for rendering forms in a browser interacting with EAE or AB Suite systems
  - Implement your own event handlers for Renderer Control events
  - Extend generated forms
  - Use multiple Renderer controls in a Web application