Infragistics Application Styling

Collaborative Design for Killer Apps

Creating effective user experiences that set your killer application apart requires contributions from application developers and graphics artists alike. Developers may have the coding skills to write that great novel of an application, but it is the artists in your organization who have the brand awareness and media skills to transform these novels into major blockbuster motion pictures. Learn how Infragistics application styling tools facilitate greater collaboration among your organization's talented craftsman to deliver user experiences more quickly and at less cost.

Derek Harmon
4/26/2010

Contents

Table of Figures

Introduction

With its Application Styling Framework™ (ASF) and NetAdvantage® AppStylist®, Infragistics furnishes enterprises and organizations like yours with the tools, technology, and a selection of professionally designed style sets that will make creating expressive user experiences an effortless feat. By the end of this paper, readers will have a good understanding of what application styling is, why it is important to organizations, and how Infragistics addresses this need with their ASF and AppStylist tools.

Those who are already aware of the importance and value of application styling and the separation of concerns between developers and designers can skip the first section and read about the design goals that Infragistics used in building our application styling tools. After that, we cover at a high-level the mechanics involved in using each of our two current design tools—AppStylist for Windows Forms and AppStylist for ASP.NET. These tools are included as part of the NetAdvantage for .NET developer tools that can be downloaded for evaluation from our Web site (www.infragistics.com/downloads).

Application Styling

Applications are now as much about how you present information as what you present. In the past, it used to be enough for an application developer to choose what business data to present (quarterly revenue for instance) and select an appropriate user interface control fitting the job, such as a four-column list view. Technological advances like the AERO (Authentic, Energetic, Reflective, and Open) user experience in the Microsoft® Windows® 7 operating system shell have increased user expectations. Applications now need to be made more usable, more active, and more memorable than ever before.

Enter the graphic artist, a talented visual designer driven by a vision of your organization and its product branding, and proficient with a variety of graphical software tools. Whereas the application developer has the experience with the development frameworks and coding practices necessary to build applications, it is the graphic artist who is most capable of finishing the application's styling to ensure it is seen as user-friendly, attractive, interactive, and representative of the image you want your organization to present.

Creating compelling user experiences therefore increasingly requires these two distinct and often mutually-exclusive skill sets:

> **Application developers** have the experience constructing rigorous object-oriented applications to demanding business requirements.
>
> **Graphics artists** have the creative flair to deliver business information to users memorably and with the greatest impact that reinforces your brand.

The challenge of modern user interface development is in how to effectively, efficiently, and effortlessly bring both disciplines into collaboration on the finished product.

Key economic drivers are underlying this movement toward multi-disciplinary application development teams that make it more than superficial: a key requirement of the new digital economy is that it mandates increased

worker productivity from IT solutions. Organizations are measurably realizing competitive benefits through increased productivity by putting a high-fidelity user interfaces in front of their business applications, and application styling reduces the cost and shortens the turnaround to realizing these benefits.

Design Once, Style Everywhere

Figure 1 -- An application with four distinct styles.

The illustration above shows how the same Agent Management application has been designed once to provide its users all of the same functionality, but it easily runs with four distinct styles depending on the selected style set.

When building systems for both external *and internal* use, application styling establishes brand consistency across applications. Internally, this helps minimize the learning curve from one application to the next to enhance worker productivity; externally, this helps to send a powerful statement about your brand identity. And if you resell your application to many different customers, using Infragistics' ASF and AppStylist empowers you to easily leverage application styling to provide the same branding and productivity benefits for your customers without involved restyling efforts.

Why is the ASF and AppStylist Important to Your Organization?

Infragistics' application styling tools bridge the divide between your graphics artists and application developers enabling them to work together effectively, efficiently, and effortlessly. The Application Styling Framework (ASF) draws the distinction clearly and crisply between the developer space and the designer space. This separation in responsibilities of both disciplines helps keep them most effective by minimizing their tendency to task-switch and work outside of their natural element.

Neither discipline should have to concern itself with the other discipline's specialties. For example, a graphics artist shouldn't have to think in terms of how to load an embedded bitmap resource from an assembly, and the application developer shouldn't have to be concerned with the contrast of mouse-over effects. By reducing the inefficiencies of job cross-over and recall of seldom used activities, your developers and designers are better able to focus on their specific tasks.

A principle of Zen philosophy is that where there is strain, struggle or trying, there is something wrong. There's no denying that application design is hard work, but it should be hard work aligned with your organization's goals. Effort directed against the constraints of your tooling and architecture is wasted energy. Application developers work within the Microsoft Visual Studio 2010 Integrated Development Environment (IDE) with the NetAdvantage for .NET tools they're familiar with, while graphics artists enjoy the designer-oriented tooling of NetAdvantage AppStylist. In this way, all members on your application design team work and interact smoothly to reach the same goal.

Every organization is facing the challenge of modernizing their software development practices. Organizations are increasingly questioning how to deal with:

Corporate, application, and client **branding.**

Management and **reuse** of a growing library of stylistic assets.

Meeting and exceeding **rising user and market expectations.**

Coordinating design of the user experience using **multi-disciplinary teams** with **different skill sets.**

Infragistics' application styling tools solve these challenges and place your organization ahead of its competitors.

Design Goals

In light of the challenges facing the marketplace as it transitioned to achieving better user experience (UX) designs through multi-disciplinary teams of developers and designers, Infragistics saw the absence of tooling available from vendors in this important area as an opportunity to provide an innovative solution. Previous tools for UX design had focused disproportionately on the role of developers. Infragistics tasked its team of expert visual designers with driving the experience that our application styling tools would deliver to graphics artists. With their input, the Infragistics development team galvanized around the following set of core design goals that application styling support requires.

Separate Styling from Development

Infragistics application styling tools separate an application's code development from the visual design of its presentation. Now application developers and graphics artists can apply their specialized talents in parallel—instead of waiting on and interfering with each other—to create compelling applications. This eliminates the costs associated with hand-offs that had to take place between developer and designer team members using a non-separated architecture. This separation in the architecture is also more flexible because an application's styling can be changed after deployment without the need to rebuild and redeploy its executable code.

Tooling for Visual Designers

Graphics artists tend to look at and solve visual problems differently from application developers; therefore, making graphics artists conform to a developer-oriented tool like Visual Studio 2010 reduces their effectiveness. Conversely, our NetAdvantage AppStylist for Windows Forms and NetAdvantage AppStylist for ASP.NET tools aim to create a visual environment for graphics artists to work comfortably within without having to deal with programming language or the .NET Framework SDK concerns. These tools are very visually task-oriented to help graphics artists design an application's appearance, establish corporate branding across all applications, and maintain an organization's style library assets.

Ability to Embed the Style Tooling Within an Application

Style tooling can be embedded within a WinForms application, allowing the application to be styled within the running application itself. Our customers can now offer application styling customization for their applications without the need to rebuild and redeploy. This would allow an organization using NetAdvantage AppStylist for Windows Forms to realize a competitive advantage by extending to its customers the capability of rebranding any of its applications.

One Point Application Style Management

Application styling can be switched from one style set to another through declaring one property or making one API call. End users of the application can be allowed to change its look and feel interactively, without the developer having to make wholesale changes across many individual controls.

Styling Similar Elements

AppStylist provides a way of styling "like" elements. For example, a graphics artist might want the column headers on a table to appear like the column headers in a list or tree without having to re-specify these settings manually for each control. Because all of these column headers belong to a common header style role, the graphics artist can apply styling and have it shared with like elements, even across control types.

Target Components by Type

The ASF permits a regular appearance to be applied across all controls of a particular control type. For instance, graphics artists can use AppStylist to affect the appearance on all grids or toolbars by customizing style roles scoped to the grid or toolbar control types.

Target Individual Instances

Infragistics controls make properties available to application developers that allow individual instances of a control type to use a different style set for its appearance. Consequently, designers have the freedom to style an individual grid instance using one particular style set (e.g., "Blue Gel") while using another style set across the rest of their application (e.g., "Electric Blue").

Minimize the Impact on Execution Speed and Memory Footprint

The infrastructure necessary for the ASF keeps its impact on an application's execution speed and memory footprint to a minimum, especially when it is not in use.

Standards-based Serialization

All settings for a style library are self contained and persistent with industry standard serialization formats (XML and CSS). This allows graphics artists to manipulate files using supplemental tooling that supports these file formats, while application developers can dynamically restyle the application at runtime by working with files that conform to these known industry standards.

Coexistence with the Infragistics Object Model - No Property Settings

The ASF co-exists with the object model exposed by Infragistics controls, and the ASF infrastructure does not set any properties on the controls. In Windows Forms applications it instead merges its style settings with those of the control at appearance resolution time. In Web applications the appearance resolution occurs on the client machine as part of their browser's built-in CSS class resolution.

Flexible Appearance Resolution

Our application styling tools encompass the flexibility to decide what precedence will be used when resolving appearances. Expressly assigning appearance settings through the control's object model should supersede conflicting style set settings to allow application developers to react in code to user demands such as an "increase font size" menu command without requiring manipulating the style set. Similarly, CSS style rules

marked by the !important token can be added through an external style sheet (for example, to improve contrast and readability of text), readily overriding font settings in the style set.

NetAdvantage AppStylist for Windows Forms

Background

NetAdvantage AppStylist has been available to graphics artists for several years, as it accompanies the NetAdvantage Windows Forms controls. With each successively enhanced release of the NetAdvantage tool set, NetAdvantage AppStylist has added new features to enhance the experience graphics artists have when using it. Recently added new features have included an eyedropper to ease color matching tasks, improvements to the Resource tab of the Style Explorer such as showing a list of UI roles that use each particular resource and instant preview, and there is now more selective resource importation from your organization's existing style library assets.

Soon after the success of AppStylist for Windows Forms, we introduced a parallel edition of NetAdvantage AppStylist for ASP.NET applications that is bundled with releases of the NetAdvantage tool set. We've distinguished these two editions by qualifying the Windows Forms edition as the NetAdvantage AppStylist for Windows Forms, and the ASP.NET edition as the NetAdvantage AppStylist for ASP.NET.

NetAdvantage AppStylist for Windows Forms in NetAdvantage for .NET 2010 Volume 1 features exporting image assets from style libraries, personalization of style libraries, and loading multiple style libraries into memory at once. The feature drawing the greatest clamor from visual designers has been the capability to generate new style libraries based on templates. It's a significant time saver in creating high impact style sets from a stock appearance without having to adjust that many style roles.

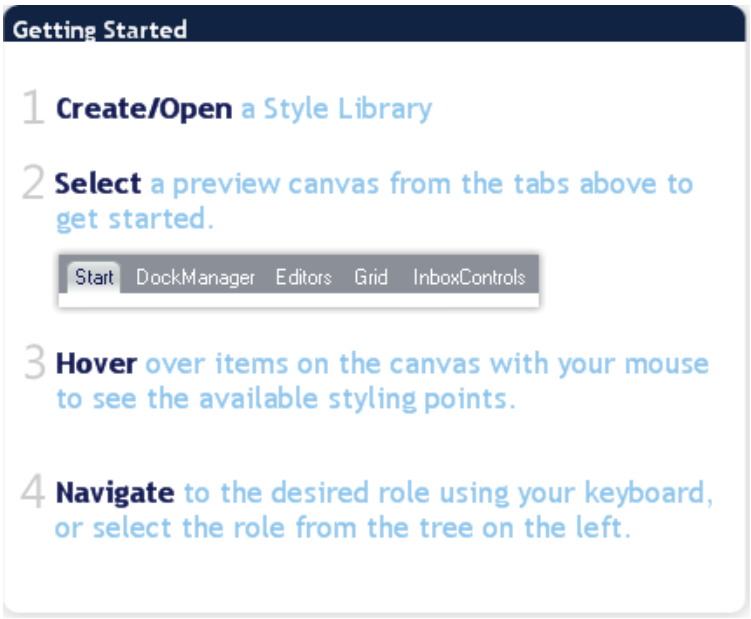

Figure 2 -- Graphics artists are guided to immediate productivity with the NetAdvantage AppStylist® Getting Started panel.

Four Steps to Style

NetAdvantage AppStylist for Windows Forms makes your graphics artists immediately productive through tooling that's been designed to have a similar look and feel to graphics media programs with which they're probably already familiar. As soon as you launch NetAdvantage AppStylist for Windows Forms you'll see a "Getting Started" panel that describes the 4-step routine which soon becomes second nature.

Your first step is to Create/Open a Style Library. Select the File | New Style Library | New Style Library from Template … menu command to base a new style library on an existing style library as a template.

Using a style template such as the "Template_Office2007.isl," one of several professionally designed templates included with the Windows Forms UI controls in NetAdvantage for .NET 2010 Volume 1, gives your application's user interface immediate embellishments like those in Microsoft Office® 2007. Notice the drag handles on the menu and toolbars, the drop down end cap on the toolbar, the rounded tabs and the hot tracking effect over the tabs, toolbar buttons, menu commands, and explorer bar views.

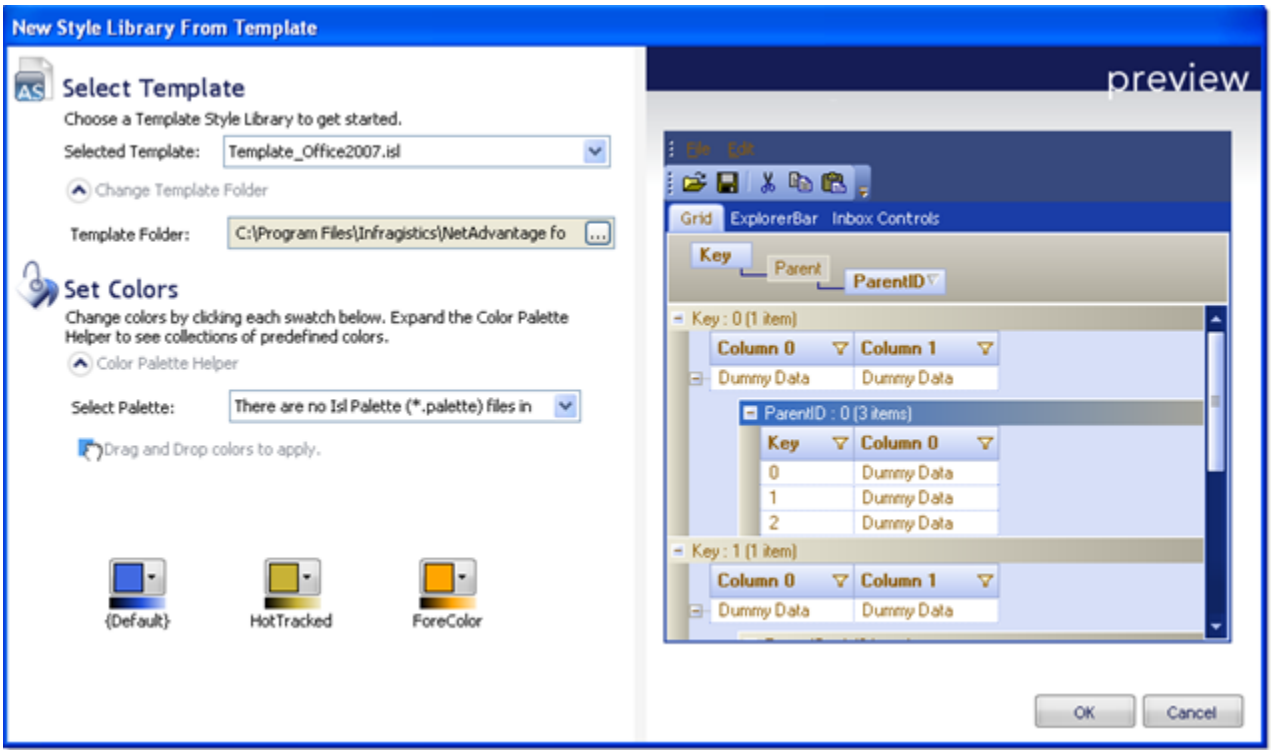

Figure 3 -- Starting Style Libraries from a Template instantly presets all style roles to that look and feel.

After you select the template there will be several color swatches that allow you to customize the color palette of the style library. For the Office 2007 template illustrated above, I set my swatches to Royal Blue, Gold, and Orange. The effects are instantly applied to scores of style roles throughout the Infragistics controls, previewed for you on the right-hand side. This preview is fully interactive, so you can move and group columns on the Infragistics WinGrid™ control to see them just as an end user might. If at any point you change your mind, it takes only seconds to select different colors.

Graphics artists often needed application developers' help to change the property settings throughout an application to make simple color changes like this in the past, or they had to learn to work with Microsoft Visual Studio 2010 and the complex control object models to make the changes themselves, which is often very time consuming. NetAdvantage AppStylist for Windows Forms makes such changes for the designer effortless.

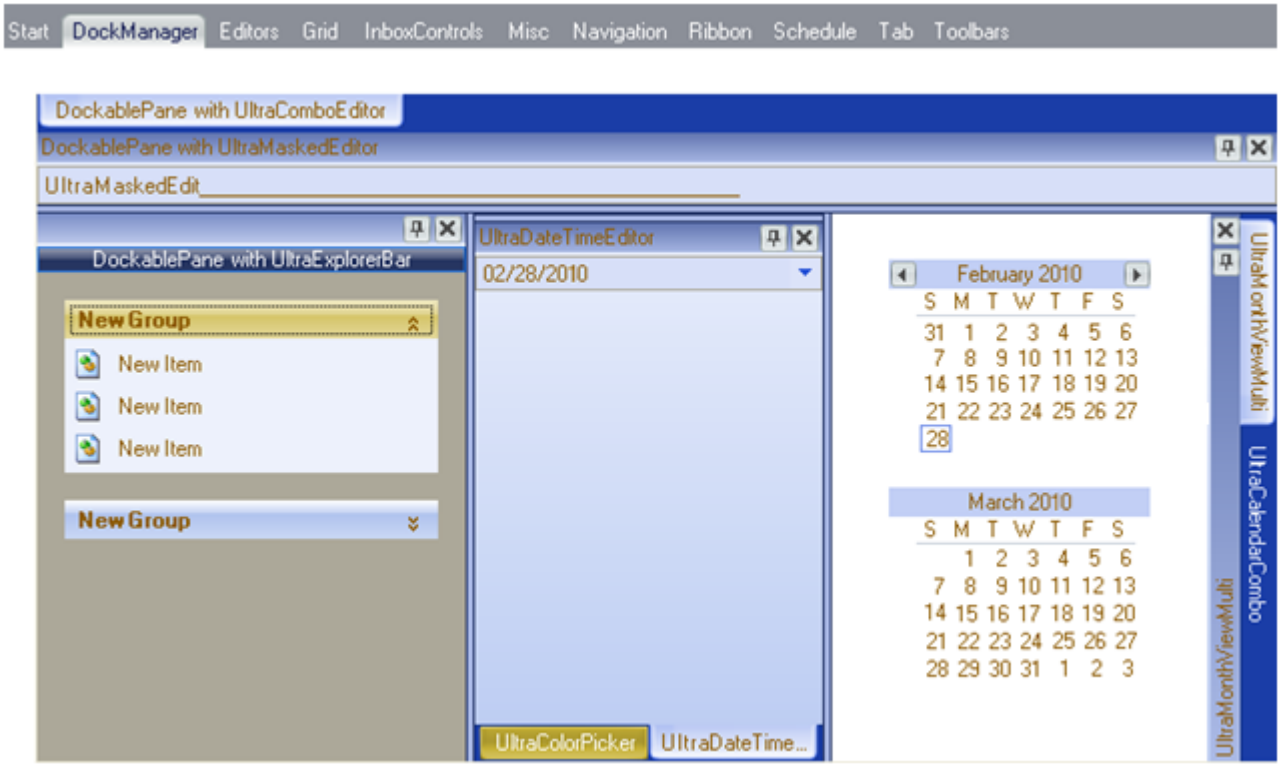

Figure 4 -- Selecting a preview tab displays an arrangement of related controls giving you broad supervision over the style library's look and feel.

The most powerful surface in NetAdvantage AppStylist for Windows Forms is where the controls used within your application are arranged with a preview of their application styling applied. Every NetAdvantage Windows Forms visual control is present and accessible by selecting any of the tabs running along the top in groups of commonly related controls. The preview canvas strives to have high fidelity in its rendition of the final appearance, but also (noting the highly interactive nature of applications) it is necessarily dynamic. Graphics artists can and should select, highlight, expand, and collapse the many interactive UI elements that provide these behaviors to see all aspects of the style library applied.

Previously, graphics artists would often wait until a test build could be produced by your organization's application developers. The designer then faced a protracted period of trying out every facet of the application's user interface to ensure everything end users could reach has been styled consistently and appropriately. This sometimes required the application developers to leave in "back doors" or vestigial code used by designers to reach all screens within the application but that needed to be removed before shipment. These workflows just do not perform well for developing today's user experiences, and the NetAdvantage AppStylist for Windows Forms design model allows your designers to accomplish these same tasks much more efficiently.

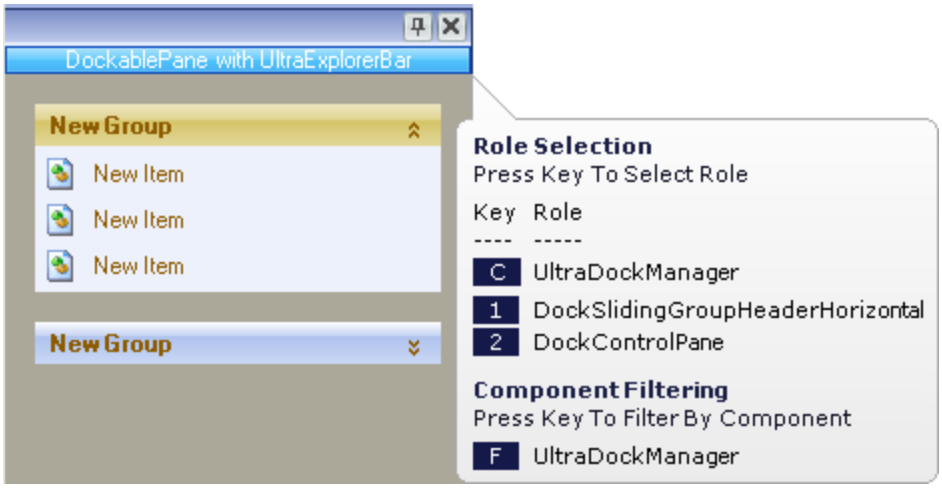

Figure 5 -- Modern user interfaces remain as sophisticated as ever, but role selection tool tips clarify what your choices are at any given moment.

When every user interface control has numerous features that can slide, glide, pop up, cascade, dock, float, collapse, or expand in front of users' eyes, it is challenging to give every option its due design attention. There is after all only so much a flat canvas can display at one time, but that's why context-sensitive role selection tool tips are available by default (experienced designers can disable them through a Tools | Options… checkbox). As you hover over UI elements on the preview canvas these tool tips pop-up with actionable keyboard shortcuts to select any variety of roles associated with the UI element beneath the mouse pointer.

Figure 6 -- Style Explorer catalogs all roles, resources, and their dependent controls.

Sometimes there's that one style role that needs to be modified, and its presence in the controls laid out on the preview canvas proves elusive. Instead of hovering over stylable elements, your designers can turn to the Style Explorer which presents a hierarchical catalog of stylable UI roles, component roles, shared (or global) roles, and resources. The visual tree in the Style Explorer parallels how designers tend to think about visual style inheritance. For example, in the preceding figure (Figure 6) you can see how DropDownEditorButton is a more specialized style than DropDownButton, which is a more specialized style than Button. Another powerful feature of the Style Explorer is its "Used By" list for the currently selected role. This conveys what controls used in your application can have the appearance of the currently selected role.

One thing graphics artists are familiar with from using products such as Adobe® Photoshop® is the notion of having a large palette of tools at their disposal: brushes, pens, pencils, curves, polygons, eyedroppers, etc. The Properties ribbon in NetAdvantage AppStylist for Windows Forms organizes the tools available to customize the appearance properties of style roles. A partial screenshot (shown in Figure 7) shows some of the properties that can be set on the DropDownEditorButton style role.

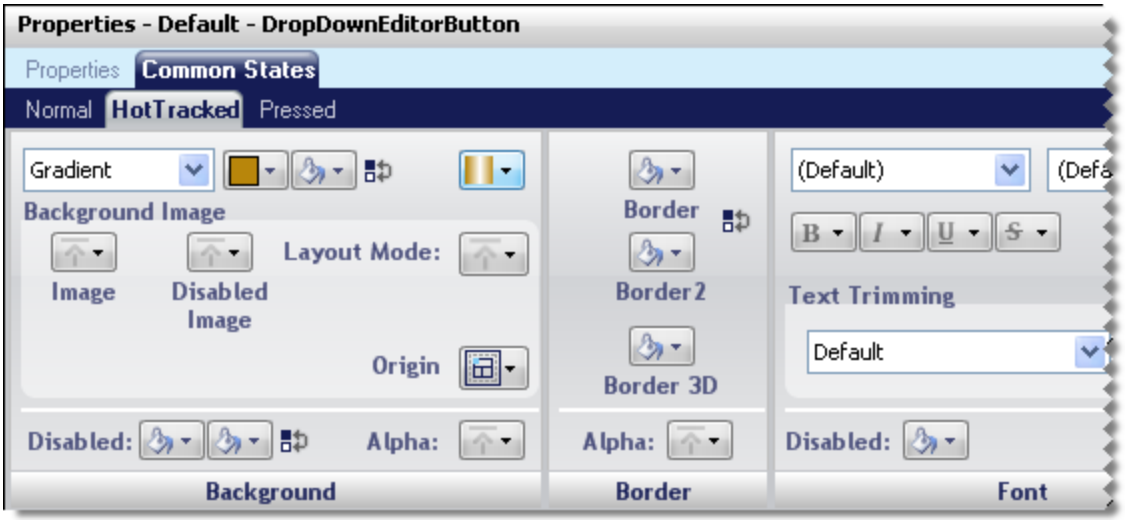

Figure 7 -- It's easy to customize the hot tracked DropDownEditorButton role's gradient background texture through the Properties ribbon.

Application developers know that many of these settings are represented as .NET Framework value types or enumeration constants (for example, whether a background is “Solid,” “Gradient,” or “Hatch”). We found that there’s really no reason why graphics artists had to learn the .NET Framework 4 SDK value types before they could become productive. NetAdvantage AppStylist for Windows Forms eliminates this barrier to their success by making all appearance properties intuitively available to the designer without them needing to be aware of the underlying programming model.

You’ve witnessed how these four steps to get started with NetAdvantage AppStylist for Windows Forms quickly lead to some very compelling style sets for your application:

1. Create/Open a Style Library.
2. Select a preview canvas from the tabs above to get started.
3. Hover over items on the canvas with your mouse to see the available styling points.
4. Navigate to the desired role using your keyboard or select the role from the tree on the left.

Similar achievements using Microsoft Visual Studio 2010 alone would have demanded a greater knowledge of the .NET Framework 4 SDK and control object models. Typically this knowledge only rested in the hands of your application developers – not your creative graphics artists. Consequently, the workflows many organizations undertake are driven from the application development perspective with the inherent inefficiencies, ineffectiveness, and extra effort that such a vantage point places on the graphics artists who are visually designing your user experience.

One Step to Go Live

While the visual design work of your organization's application styling has been going on in parallel, your application developers have been implementing the new or enhanced release of your latest application. Very early on, and perhaps continuously throughout their development process, the application developers will receive style libraries packaged as .isl files from the graphics artists.

There are a variety of deployment options available for loading the Style Library (.isl file) that include embedding it as a resource within an application assembly or retrieving it from a Web service at run-time. One of the simplest deployment options any developer can perform is to add the .isl file to the project, placing it alongside the executable, and then add this one line of code to the application to turn on the ASF and load the desired style:

VB.NET code

```
Infragistics.Win.AppStyling.StyleManager.Load("BlueGold.isl")
```

C# code

```
Infragistics.Win.AppStyling.StyleManager.Load("BlueGold.isl");
```

Listing 1 -- The static Load method on the StyleManager can load an .isl by filename or stream.

Instantly, the application starts running using the appearances defined by the new Style Library. Depending on how many controls are present on each Form, this may have required hundreds—if not thousands—of tedious and error prone property assignments in the past to accomplish the same result. The ASF gives immediate visual feedback that lets you evaluate the successes and failures of your application's user experience incrementally throughout development so that you can turn that information around to your visual design team members faster than previously possible.

NetAdvantage AppStylist for ASP.NET

Background

As with Windows Forms applications, ASP.NET applications face similar challenges in creating compelling user experiences on the Web. Following the highly successful design model of the NetAdvantage AppStylist for Windows Forms, Infragistics released NetAdvantage AppStylist for ASP.NET to rave reviews from the Web design community.

The following sections compare and contrast this new technology to existing ones to help readers quickly gain an understanding through analogy and to see the value added in contrast to existing technologies as appropriate. After that, we'll dive right in to illuminate how one goes about taking advantage of AppStylist for ASP.NET.

Differences from Application Styling in Windows Forms

The alternative media (cascading style sheets, HTML markup, image files) making up Web applications required an evolution in the Application Styling Framework™ (ASF). There were new challenges to facilitate design work under different browser interpretations (including different interpretations within the same browser, such as the "Quirks" versus "Standards" mode in Internet Explorer) of markup and style inheritance. At the same time, the Web platform freely and efficiently performs appearance resolution when using technologies such as the World Wide Web Consortium (W3C) Cascading Style Sheet (CSS) recommendation. The following are some of the more significant differences in application styling for ASP.NET applications compared to Windows Forms applications.

Style libraries are folder-based instead of file-based. Designers work with .wsl (Web style library) files that describe the contents of their style library, but because the style library is now composed of separate .css and image files (.gif, .png, .jpg) organized by style set and by control type, style libraries can span multiple files in a nested folder structure. This simplifies the import and export of graphic files from other tools for designers. Run-time bandwidth consumption improves performance because only the .css and image files necessary for the end user based on the control types present on a form need to be sent over the network.

Cascading Style Sheet (CSS) standard. Application styling for Windows Forms serialized appearance settings and image resources in XML format. In the Web environment, leveraging cascading style sheets proves to be an advantage for the widespread support and Web designer experience it enjoys. Those appearance settings that can be styled for the Web conform to the style properties as defined by CSS and supported by major browser vendors.

Configuration through "web.config." Application developers don't need to write code to activate application styling for their ASP.NET applications as it can become instantly activated by setting the `enableAppStyling`, `styleSetName` and `styleSetPath` attributes in your application's "web.config" file. A Web site's configuration file can be updated to use a new style set without ever taking the site down or rebuilding it.

Application Styling Compared to CSS-Only Web design

A big question on every Web developer and designer's mind is how does application styling for ASP.NET improve upon the common use of CSS in an ASP.NET application today. The answer is that application styling for ASP.NET offers greater benefits in terms of the **organization**, **integration,** and **design automation** benefits provided by its architecture, leveraging the CSS standard when styling entire Web applications.

By elevating the focus of design above the details of applying CSS, which requires tedious attention across every control on every page of the application, NetAdvantage AppStylist for ASP.NET raises designer and developer productivity. Less time is spent mechanically assigning CSS classes, and more time is spent imagining the look of the application.

For instance, using ten different controls across a Web site might entail referencing over a hundred CSS classes. These CSS classes need meaningful names and a management scheme to prevent naming collisions and to facilitate quickly finding and editing particular CSS style definitions that correspond to pieces of HTML in the browser. The naming taxonomy that the Application Styling Framework uses will produce unique CSS class names based on organization, style set, control type, and style role to eliminate collisions and make lookup a cinch. The folder-based storage of Web Style Library assets aggregates CSS style definitions common to a particular control type in one style sheet. Since each control appearing on the Web site references its own control type's CSS style sheet, this cuts down on the unnecessary overhead being transmitted to clients.

In addition to being organized from the outside of the controls, the ASF is also highly integrative on the inside of the controls. Style roles correspond to the independent stylable UI elements that constitute the control's appearance. When the control renders itself it will know what pieces of HTML require the CSS class of a specific style role (selected, disabled, an alternate row, a filter drop down item, etc.), and it can specify that CSS class reference on the spot, eliminating the need for developers or designers to manually apply styles to each control. Further, applying styles and CSS classes to all of the individual stylable elements of a control can be complex, time-consuming, and sometimes not even possible, so having the ASF do this for developers is a benefit that will be recognized immediately, reducing implementation and maintenance time and simplifying developer-designer interaction..

What all of these CSS class assignments made throughout your application automatically mean for your organization's graphics artists is that their design choices are applied automatically. Without AppStylist, if a designer experiments with a particular look, it may take a number of property settings to reference the correct CSS across the controls on a representative page in the application. Should that style not turn out as well as had been hoped, the designer has to manually back out of the changes they had made which can be frustratingly nonproductive. With NetAdvantage AppStylist for ASP.NET, changes having ramifications on tens or even hundreds of CSS classes take only seconds to make (and un-make), which reduces risk and increases productivity while experimenting to find the best appearance.

Application Styling Compared to Presets

Six years ago, Infragistics released Presets as a way for NetAdvantage ASP.NET controls to be given a consistent appearance. Presets were XML files that, when deserialized, would assign their values to the properties and style sub-objects of the control. In addition to producing an initial appearance, Presets could also establish the behavior of the control by setting non-appearance (behavioral) properties. For example, organizations could load a standard Preset to initialize a WebGrid™ control that allowed rows to be inserted, updated, and deleted.

Presets filled a need at a time when there were few other viable solutions, and they still have a valuable use to initialize property values – especially behavioral property values. When the look and feel of controls was designed by setting tens or hundreds of properties through Visual Studio 2010, it was important to save that work into a Preset so the property assignments could be replicated onto other instances of that control. Presets however did not force a consistent or personalized look and feel on applications because they had to be loaded for every instance of a control. Another drawback is that Presets directly affect properties on the controls

themselves, which increases maintenance if one wants to change the preset and can cause some blending when applying multiple presets.

One of the major design goals for application styling–not to affect the properties on the controls' object models–was learned from this experience with Presets. The ASF resolves appearance properties on a just-in-time basis whether at appearance resolution time in Windows Forms or through CSS class inheritance in ASP.NET without affecting the pristine property values on the controls. The ASF supports the appearance capabilities that developers used to have with Presets and much more, so Application Styling Framework should be used in favor of Presets for defining the appearance of controls. On a per-instance control basis for initializing property values the use of behavioral Presets is still indicated.

Application Styling Compared to ASP.NET Themes

Themes in ASP.NET 2.0 was a mechanism based largely on the markup representation of ASP.NET controls after removing certain non-themeable attributes like a control's ID and runat="server". As promising as this may have looked, the graphic artist using it to style an application required at least minimal ASP.NET knowledge to design a control by itself and then basically cut and paste this control's markup representation into a .skin file to be deployed to a special App_Themes folder.

In contrast, AppStylist lets the designer focus on the visual representation of the controls in an application, rather than technical platform details for ASP.NET. Designing a style set should be efficient and effortless, and with the strong tool support in the NetAdvantage AppStylist for ASP.NET and Application Styling Configuration Add-In, it is.

Still Four Steps to Style

Graphics artists already experienced with other Web design tools will quickly become productive working with NetAdvantage AppStylist for ASP.NET. Its "Getting Started" panel that you see on launch mirrors the familiar 4-step routine used by NetAdvantage AppStylist for Windows Forms. Immediate visual feedback and easily made changes will spur creative progress so that your graphics artists won't need very much time to reach a polished and professional application style.

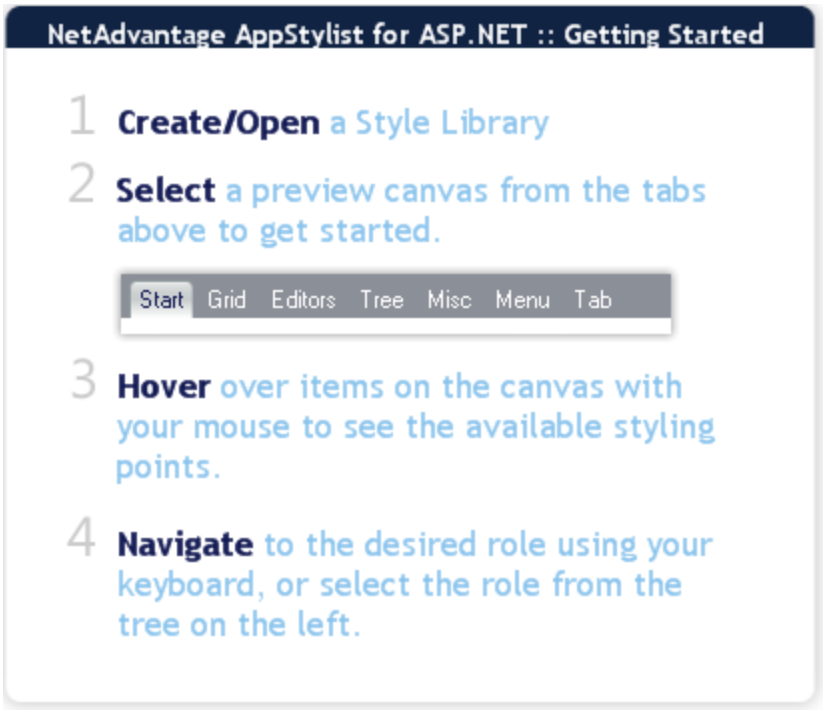

Figure 8 – "Getting Started" process familiar to visual designers who have used NetAdvantage AppStylist® for Windows Forms.

As before, your first step will be to Create/Open a Style Library. Knowing that not every organization can afford a fully outfitted visual design department, a large number of professionally designed style sets have been packaged with NetAdvantage AppStylist for ASP.NET, and those who do have graphics artists can use these supplied style sets as a springboard for further customization. Open the "Infragistics Style Library.wsl" file this time and select its "Office 2007 Blue" style set from the "Active Style Set" drop down in the Style Explorer to serve as the ground work for your application's appearance.

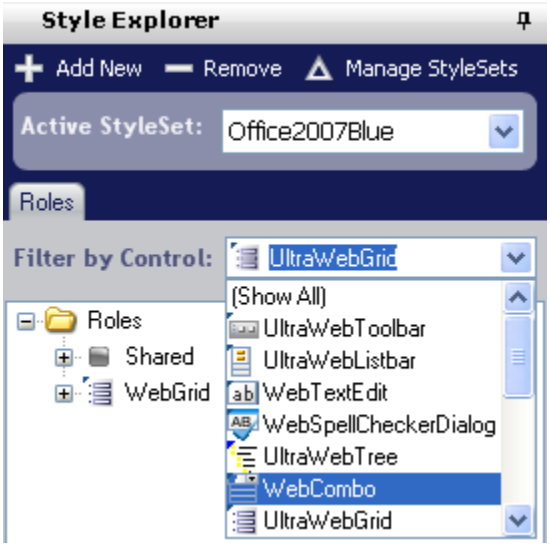

Figure 9 -- Style Explorer lets you narrow down the active style set and control types currently under design.

The preview canvas continues to be the focal point for design work providing *critical and immediate visual feedback* on the effects of style changes on the controls comprising your application. Notice the Canvas Options button--this allows the graphic artist to activate the many features of the NetAdvantage controls for ASP.NET that ordinarily an application developer would have turn on through Visual Studio 2010. Designers become more independent from application developers because they can see the styling of controls based on knowing what control features are called for in the Web application *without waiting for developers to expose those features* in the application itself.

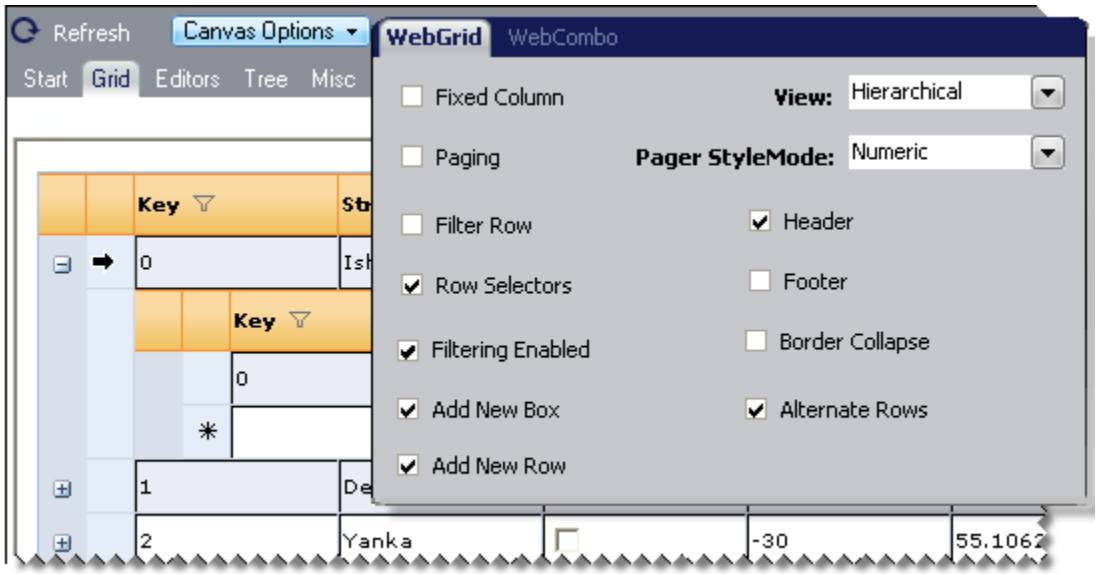

Figure 10 -- Canvas options let the graphic artist preview the appearance of NetAdvantage control features that the application developer may enable programmatically.

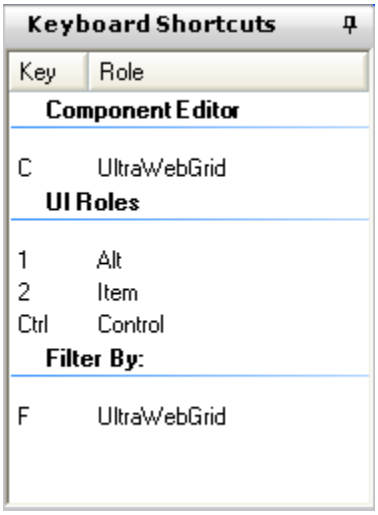

Figure 11 -- Keyboard shortcuts are equivalent to role selection tool tips.

As you hover over stylable UI elements on the preview canvas, context-sensitive keyboard shortcuts are displayed in a separate task panel. Designers can reposition or dock this task panel where they feel most comfortable. A task panel was used in NetAdvantage AppStylist for ASP.NET instead of the pop-up role selection tool tips shown in Figure 5 because the preview canvas is an embedded Microsoft Internet Explorer browser window. This is because any mouse over interactivity observed on the preview canvas must remain a high fidelity representation of the same behavior end users will see when viewing the controls with this styling in Microsoft Internet Explorer themselves.

Fashioning cross browser styles that appear equivalent requires care due to differing degrees of browser conformance to the current Web standards for styles and markup. For best results, the use of DOCTYPEs that place the browser into what is called “Standards mode” is strongly recommended. Application developers can add these DOCTYPE declarations at the top of their

ASP.NET .master or .aspx files. Whether a Web application has been designed with "Standards mode" in mind or not, NetAdvantage AppStylist for ASP.NET simplifies previewing controls that have application styling in both "Standards mode" and "Quirks mode" through a setting for DOCTYPE available on the Tools | Options… dialog.

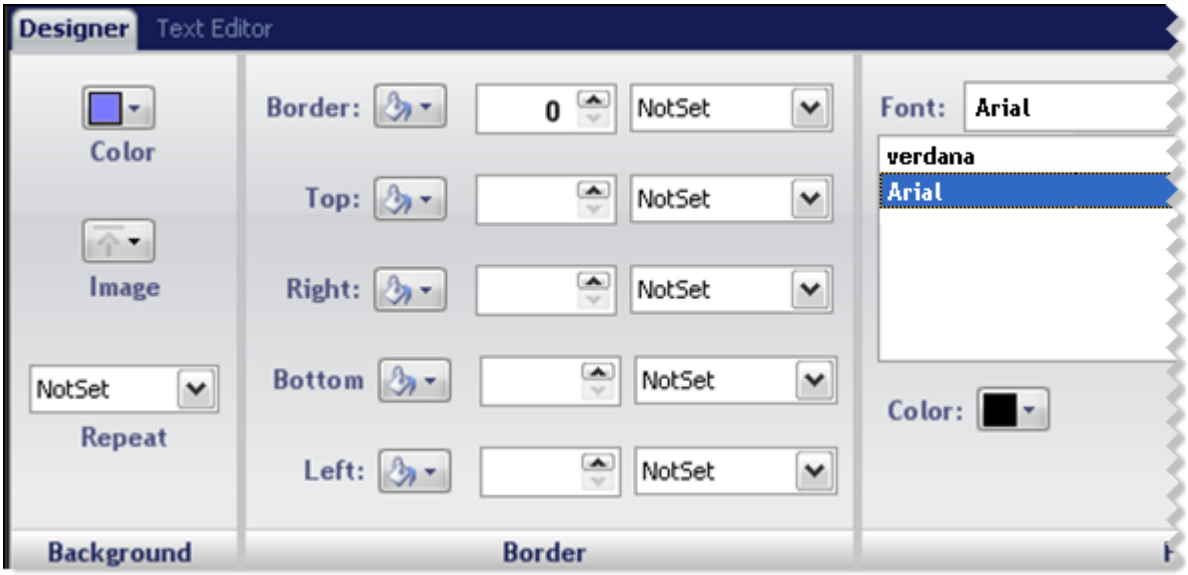

Figure 12 -- Properties ribbon features a Designer tab cataloging CSS style properties that can be set and a Text Editor tab that allows advanced designers to manipulate the CSS directly.

When you have selected a stylable UI element of the WebGrid control to customize, NetAdvantage AppStylist for ASP.NET fills the intuitive Properties ribbon docked at the bottom of the screen with all of the current style settings. For the more advanced Web designer with an intimate knowledge of CSS style rules or as a learning aid for more novice Web designers, there is a Text Editor tab that opens a CSS editor with syntax coloring. Directly editing CSS allows designers extremely fine-grained control over each style role's appearance and the ability to take advantage of new visual styles supported by future versions of the CSS standard.

Configure to Go Live

Application developers can still programmatically enable (and disable so as to opt-out portions of the Web site) the ASF through the EnableAppStyling property and set their initial StyleSetName and StyleSetPath properties at the scope of the page or user control by using the non-visual WebPageStyler control. The best way to affect application-wide styles is through the web.config file for your Web site. When activated through Web site configuration, changes to the default style set name and path can be made without having to take down the Web site for maintenance.

Simplifying this configuration for application developers is the Application Styling Configuration Add-in for Microsoft Visual Studio 2005/2008/2010 which comes packaged with NetAdvantage ASP.NET controls. Developers can install the Application Styling Configuration Add-in to Microsoft Visual Studio .NET manually.

Once installed, the add-in can be accessed within the Visual Studio .NET IDE through the Tools | Application Styling Configuration… menu command. The add-in simplifies two common developer tasks:

1. It updates the web.config file with ASF configuration information such as the name and path to the style set, whether the ASF is enabled, and potentially an override of the image directory where image files are stored.
2. It imports the contents of style sets into subfolders of your Web application project (or Web site). Recall that in ASP.NET, the Style Library is folder-based and must contain multiple .css and image files that controls will reference. This feature makes file management easier.

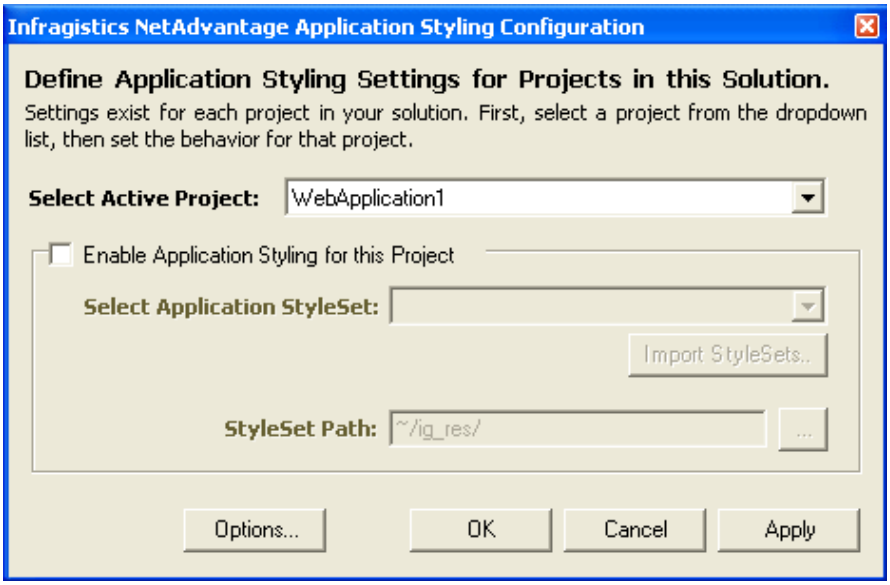

Figure 13 -- Developers enable application styling in the web.config by checking a checkbox.

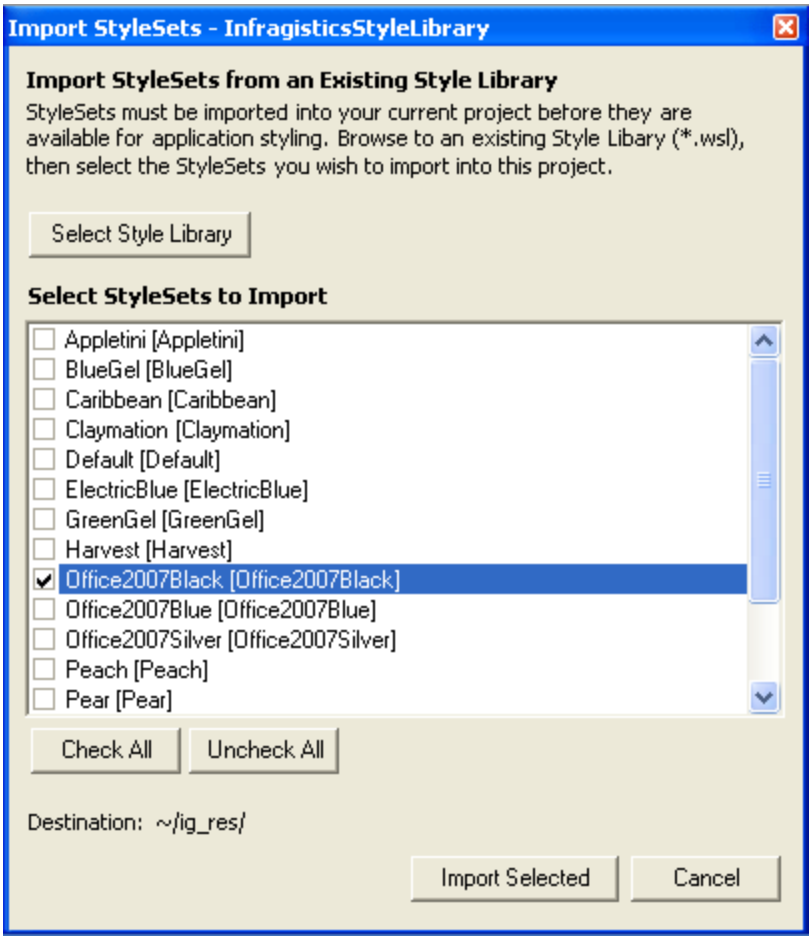

Figure 14 -- Developers can import style set files from a style library into the Web application.

Summary

NetAdvantage AppStylist for Windows Forms and NetAdvantage AppStylist for ASP.NET empowers your development and visual design teams to collaborate effectively, efficiently, and effortlessly as they drive the creation of your rich client applications to successful fruition. In this paper you've seen how the barriers to collaboration between developers and designers can be torn down through adoption of Infragistics application styling tools. Designers can work in parallel with developers and garner instant feedback that expedites the turnaround of styling issues. Infragistics empowers some of your organization's most creative people–its graphics artists–to become partners in branding your applications to be more user-friendly, memorable, and above all reflective of your product or service's primary vision.