

How Telerik's ASP.NET AJAX Controls Help You Build High-Performing Applications Faster

Contents

Optimization Techniques Utilized in RadControls for ASP.NET AJAX	2
• Content Delivery Network (CDN) Support	2
• HTTP Compression with RadCompression	2
• Performance Optimization Helper Controls	2
• Rich Client-side Capabilities	3
• Flexible Semantic Rendering	3
• CSS Sprites for Minimal HTML Footprint	3
Control-specific Optimization Techniques	4
• Grid	4
• Editor	6
• Scheduler	8
• TreeView	10
• Combobox	11
• Menu	12
• TabStrip	12
• ToolTip	12
• Splitter	12
• Calendar	13
• Input	13
Additional Performance Optimization Resources	13

Optimization Techniques Utilized in Rad-Controls for ASP.NET AJAX

At Telerik, we know that claims about offering “the fastest”, “the richest” or “the most complete” component suite mean nothing without facts to back them up. This is why we have assembled a host of facts and data about the optimization techniques utilized in RadControls for ASP.NET AJAX to actually show you they are the fastest, the richest, and the most complete .NET UI components on the market.

The facts below explain specific ways you can optimize the performance of your applications built with Telerik’s RadControls for ASP.NET AJAX and how we are attacking the performance issues for many of our key controls.

• Content Delivery Network (CDN) Support

Telerik’s CDN is hosted on the Amazon CloudFront service, a global content delivery service used to cache and deliver all types of content, including images, videos, CSS and JavaScript files. This way if your app is built with RadControls for ASP.NET AJAX, its client requests will automatically be redirected to the nearest server as opposed to the central web server and will be able to more quickly retrieve and download its content. What is more, Telerik covers the hosting expenses for your app’s resources stored on the CDN.

• HTTP Compression with RadCompression

RadControls allow you to achieve less traffic and faster page load for your app completely codelessly thanks to its HttpModule, RadCompression. In other words, RadCompression will intercept the bits that your server is sending back to a browser (or Silverlight-client, for that matter) and compress them. Once the compressed bits reach the browser, standard browser technology takes-over and decompresses the response so your application can work with it normally. The compression process is completely transparent to your client-side code (JavaScript or Silverlight) and your server-side code. It simply reduces the number of bits that must be sent over the wire (from your server to your client) and thus improves your page performance by reducing the TTLB (time to last byte) to up to 75%. [Read the full article and see test results »](#)

• Performance Optimization Helper Controls

Telerik’s ASP.NET AJAX suite ships with several components engineered to significantly improve your Telerik-based application’s performance.

• RadScriptManager and RadStyleSheetManager

Since the number of requests a page generates upon initial load is one of the most common performance bottlenecks, Telerik has devised the RadScriptManager and RadStyleSheetManager controls to help you combine all JavaScript and CSS requests to the server into a single request, thus greatly improving the overall page load time. [Read the full article for a complete analysis of how the RadManagers impact page performance »](#)

• RadInputManager

The RadInputManager control is an input validation control providing your end users with real-time feedback about the text they have entered. RadInputManager is the best choice for top performance and highly-optimized pages for a few simple reasons: it reduces the bytes sent over the wire and has a better caching story, it is much faster to configure than traditional ASP.NET validation controls and it improves overall page load performance. [Read the full article for a complete analysis of how the RadInputManager impact page performance »](#)

• RadAjaxManager:

The Telerik AjaxManager control offers a completely codeless approach for AJAX-enabling existing or new web applications, using Microsoft ASP.NET AJAX engine. First, you don't have to clutter your markup with the UpdatePanels normally required by ASP.NET AJAX, making your code easier to maintain and read. Next, you don't have to manually think through how Triggers should be defined on your UpdatePanels. The RadAjaxManager automatically figures that out based on your simple definition the controls that should be updated after specific control events fire. And finally, RadAjaxManager provides a complete client-side API that makes it easy to perform advanced ASP.NET AJAX operations without having to write a lot JavaScript manually. [Read how it works »](#)

• Rich Client-side Capabilities

RadControls for ASP.NET AJAX allow developers to fully leverage the widely supported client-oriented programming of the ASP.NET AJAX framework. The controls offer versatile client-side features which can be immediately persisted on the server. Create client-side scripts that modify the controls on the client and have the changes be reflected when accessing the same controls on the server. That means the controls can use JavaScript to render their UI in the browser, reducing trips to the server and speeding-up your application's performance.

Additionally, RadControls expose a number of powerful databinding and caching techniques for taking full control over performance, lightweight JSON traffics, as well as complete support for ViewState-less operation. The integrated jQuery library is also an important element of our client-side databinding and we use it for advanced calling of PageMethods and WebServices. Check out the following blogs posts and examples for more details.

- [Optimizing ViewState usage](#)
- [RadGrid's Client-side databinding help topic](#)
- [RadGrid .NET 3.5 Client-side Binding demo](#)
- [jQuery Integration](#)
- [Grid client-side data-binding performance with 1 mil. Records](#)

• Flexible Semantic Rendering

The Telerik ASP.NET AJAX navigation components use advanced CSS rendering with list items rather than HTML tables in order to keep the page output clean and minimal. The semantic rendering not only improves the responsiveness of your applications, but it also makes your web pages more easily understandable by screen readers and thus increases its accessibility to visually impaired users. There's another benefit coming from the semantic markup of our UI controls – the HTML is much more search engine (SEO) friendly.

• CSS Sprites for Minimal HTML Footprint

What is one of the most common bottlenecks that hurt web pages' load performance? It's concurrent connections to the web server. Modern browsers can only download a handful of items from the same domain at one time, which means pages with tons of images, CSS files, and JavaScript files take an exceptionally long time to download - not because of the bits, but because of the connection limit. Telerik RadControls for ASP.NET AJAX use CSS sprites to combine numerous images into one, so that you have only one request to the server. Then the relevant part of the sprite image is displayed using CSS. This decreases page load time between 20-60%. [Read full details »](#)

Control-specific Optimization Techniques

• RadGrid

When picking a grid control, it's important to focus on features, ease of configuration, page load time, performance with large data sets, and support for advanced optimization techniques. Let's see how RadGrid addresses each of these requirements.

Features

RadGrid has hundreds of built-in features spanning everything from pop-up data editing and filtering to client-side column reordering and a number of built-in skins. You can control almost all features by setting simple properties and thanks to the rich visual configurators and SmartTags in Visual Studio, this setup task is even easier. Telerik's online demos highlight many of the Grid's features and provide the complete source code for them, so that you can see for yourself that there is no magic happening "behind the curtain".

[See RadGrid features in action »](#)

[Read about RadGrid performance tips and tricks »](#)

Page Load Performance

Good features aren't good enough to make a grid great. They need to load quickly and improve the user experience rather than interrupting it. Exactly for this reason RadGrid is built in a way, so that its total JavaScript payload for all features is as small as possible – it's only 174KB. RadGrid also leverages CSS and sprites to minimize its HTML footprint and to reduce the number of requests the Grid requires to load. In fact, RadGrid renders the smallest HTML footprint and generates the smallest number of request compared to any other commercial data grid.

Handling Large Data Sets

Grids need to be able to handle large amounts of data on the web. Telerik's RadGrid is optimized for these scenarios. In .NET 2.0 scenarios RadGrid for ASP.NET AJAX is optimized to load, page, sort, and filter up to 100,000 records without performance problems. In .NET 3.5 scenarios, though, the control automatically starts using LINQ-based techniques to easily handle 1 million records.

[See RadGrid large data processing in action »](#)

The screenshot shows two panels: 'Settings' and 'Measurements'. The 'Settings' panel includes three checkboxes: 'Enable caching' (checked), 'Let RadGrid do sorting instead of SQL Server' (unchecked), and 'Show aggregates' (unchecked). The 'Measurements' panel displays performance metrics: 'Time to retrieve data: 0 ms (from Selecting to Selected.LinqDataSource events)', 'Time to bind, aggregate(if needed) and render data: 363 ms', 'Server time: 363 ms (from Init to Render)', and 'Total time: Perform sorting/paging/filtering operation to see total time'.

ProductID	Product name	Unit price	Quantity per unit	Units in stock	Discontinued
1	Chai	\$18.00	10 boxes x 20 bags	39	<input type="checkbox"/>
2	Chang	\$19.00	24 - 12 oz bottles	17	<input type="checkbox"/>
3	Aniseed Syrup	\$10.00	12 - 550 ml bottles	13	<input type="checkbox"/>
4	Chef Anton's Cajun Seasoning	\$22.00	48 - 6 oz jars	53	<input type="checkbox"/>
5	Chef Anton's Gumbo Mix	\$21.35	36 boxes	0	<input checked="" type="checkbox"/>
6	Grandma's Boysenberry Spread	\$25.00	12 - 8 oz jars	120	<input type="checkbox"/>
7	Uncle Bob's Organic Dried Pears	\$30.00	12 - 1 lb pkgs.	15	<input type="checkbox"/>
8	Northwoods Cranberry Sauce	\$40.00	12 - 12 oz jars	6	<input type="checkbox"/>
9	Mishi Kobe Niku	\$97.00	18 - 500 g pkgs.	29	<input checked="" type="checkbox"/>
10	Ikura	\$31.00	12 - 200 ml jars	31	<input type="checkbox"/>

Navigation: 1 2 3 4 5 6 7 8 9 10 ... Page size: 10 335104 items in 33511 pages

Advanced Optimization Techniques

There is always a trade-off between the ease of use of a UI component and its performance optimization. The key is to find a component that makes it easy to add advanced performance optimizations when you need them. RadGrid exposes a number of techniques for taking full control over performance, such as powerful client-side data binding and complete support for ViewState-less operation. With client-side binding, for example, RadGrid can page, sort, and filter data with near client-like responsiveness.

[See RadGrid virtual scrolling with client binding in action »](#)

[See RadGrid client-side data binding in action »](#)

[See RadGrid ViewState optimization techniques »](#)

[RadGrid Client-side Binding and Caching »](#)

RadGrid's ViewState size is 3372 bytes

RadGrid's ViewState on/off

Ship Name	Ship Address	Ship City	Ship Postal Code	Ship Count
Toms Spezialitäten	Luisenstr. 48	Münster	44087	Germany
Toms Spezialitäten	Luisenstr. 48	Münster	44087	Germany
Victuailles en stock	2, rue du Commerce	Lyon	69004	France
Victuailles en stock	2, rue du Commerce	Lyon	69004	France
Victuailles en stock	2, rue du Commerce	Lyon	69004	France
Ernst Handel	Kirchgasse 6	Graz	8010	Austria
Ernst Handel	Kirchgasse 6	Graz	8010	Austria
Ernst Handel	Kirchgasse 6	Graz	8010	Austria
Ottilies Käseladen	Mehrheimerstr. 369	Köln	50739	Germany
Ottilies Käseladen	Mehrheimerstr. 369	Köln	50739	Germany
Ottilies Käseladen	Mehrheimerstr. 369	Köln	50739	Germany

Page size: 50 600 items in 12 pages

Accessibility

People often forget the importance of accessibility in web development. If you are building applications for large enterprises or governments, though, you can't afford not to address it. RadGrid is fully compliant with Section 508 accessibility guidelines, which means it can even be accessed by clients that don't have JavaScript enabled! While many grids claim to be accessible, few truly take the accessibility support as far as RadGrid.

[See RadGrid running in no JavaScript mode »](#)

CustomerID	Contact Name	Company	Address
ALFKI	Maria Anders	Alfreds Futterkiste	Obere Str. 57
ANATR	Ana Trujillo	Ana Trujillo Emparedados y helados	Avda. de la Constitución 2222
ANTON	Antonio Moreno	Antonio Moreno Taquería	Mataderos 2312
AROUT	Thomas Hardy	Around the Horn	120 Hanover Sq.
BERGS	Christina Berglund	Berglunds snabbköp	Berguvsvägen 8
BLAUS	Hanna Moos	Blauer See Delikatessen	Forsterstr. 57
BLOMP	Frédérique Citeaux	Blondesdélis père et fils	24, place Kléber
BOLID	Martin Sommer	Bólido Comidas preparadas	C/ Araquil, 67
BONAP	Laurence Leblan	Bon app'	12, rue des Bouchers
BOTTM	Elizabeth Lincoln	Bottom-Dollar Markets	23 Tsawassen Blvd.

RadGrid By the Numbers

The following numbers are results of a test performed on a simple page with RadGrid using default settings bound to a LinqDataSource with a query that returns 1,000,000 records, each with 5 columns.

RadGrid for ASP.NET AJAX

Avg. load time

63ms

JavaScript size

74KB

• RadEditor

Among the most important criteria for picking a rich text editor for the web are richness and page load time. It also helps to find an editor that has proven itself in the “real world” and since Telerik’s RadEditor is the preferred choice of Microsoft’s MSDN, CodePlex, and SharePoint teams, we think you’ll agree you’re in good company.

Page Load Time

Telerik’s Editor for ASP.NET AJAX is one of the richest and fastest loading rich text editors on the market, a combination you don’t often find. When our engineers set out to build the best rich text editor for the web, they recognized that one of the most important requirements is that it supports a sub-second page load time. And that’s exactly what you get with RadEditor for ASP.NET AJAX.

RadEditor loads on all major browsers in 0.75 seconds on average thanks to two very important features: it loads only the JavaScript for the toolbar features used at the moment, and it uses CSS sprites for skinning. Furthermore, tools that are not immediately used by the Editor automatically employ a “lazy loading” technique to ensure that even the most complex of the Editor’s configurations do not hurt your page load time.

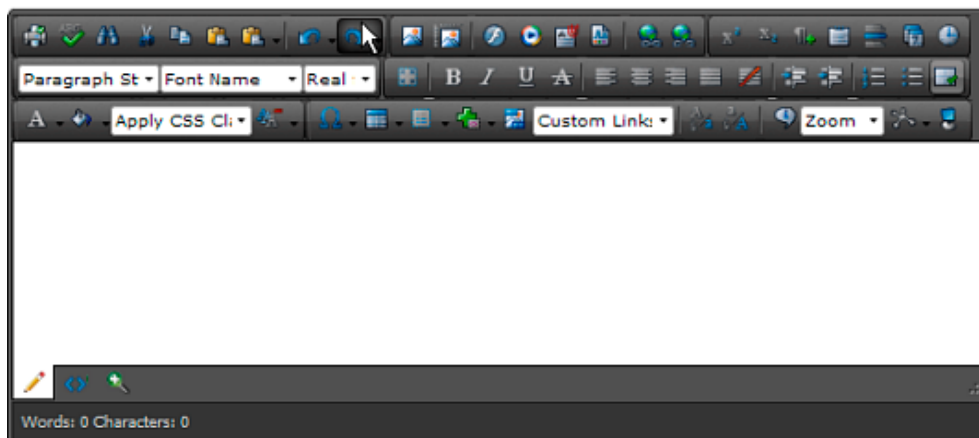
Need a lot of Editors on your page at one time (SharePoint developers, I’m looking at you)? That’s usually a performance killer for rich text editors, but RadEditor provides optimization techniques allowing sharing toolbars between Editor instances making it possible to have more than 20 Editors on a page at one time.

[See RadEditor optimization techniques for many Editors on one page »](#)

[See RadEditor work with large content without any performance problems »](#)

- Use Tool Provider
- Use ShowOnFocus Toolbar Mode

Apply Settings

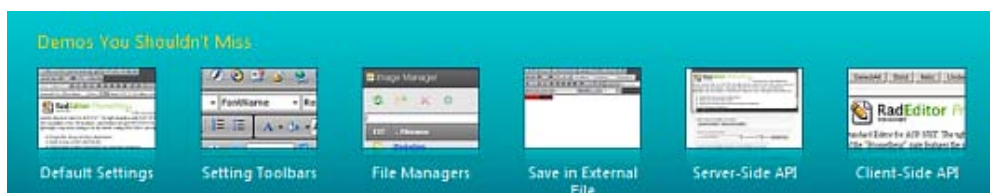


Richness

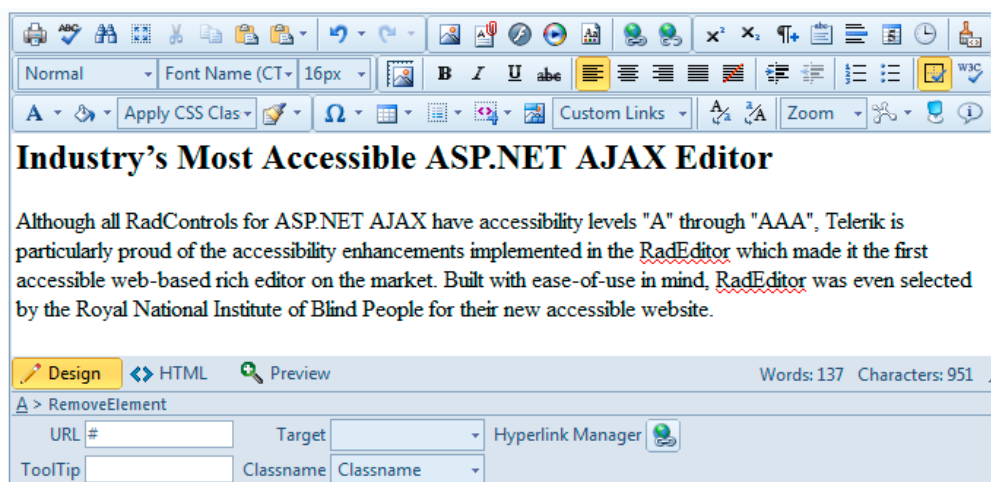
There is no question that Microsoft Word is the de facto standard for word processing on the desktop. With RadEditor, you get a comparable level of richness in a web-based environment and support for complex text editing in a standards-based, XHTML-compliant UI component. RadEditor even supports many of the keyboard shortcuts Word users are accustomed to, making it easy for your end-users to work with RadEditor (read: less training and customer support).

It would be impossible to capture all of the Editor's features in a simple paragraph, but highlights include a built-in inline spellchecker, a rich client-side and server-side extensibility API, a built-in image uploader and editor, and (like all Telerik controls) more than a dozen (and highly optimized) CSS skins. Spend some time with the online demos to see all of these features first hand.

[See RadEditor features in action »](#)



[See RadEditor accessibility features »](#)



RadEditor by the Numbers

The following numbers are based on a simple page with a RadEditor using default settings loaded in Internet Explorer 7.

RadGrid for ASP.NET AJAX	
Avg. load time	680 ms
Number of requests	21 Requests (w/o RadManagers) 15 JS files (12 for RadEditor and RadWindow, and 3 for ASP.NET AJAX), 5 CSS files (3 for RadEditor, 2 for RadWindow), 4 images files (together for RadEditor and RadWindow)
JavaScript size	Variable RadEditor, RadWindow, RadSpell files only – 94.1 Kb

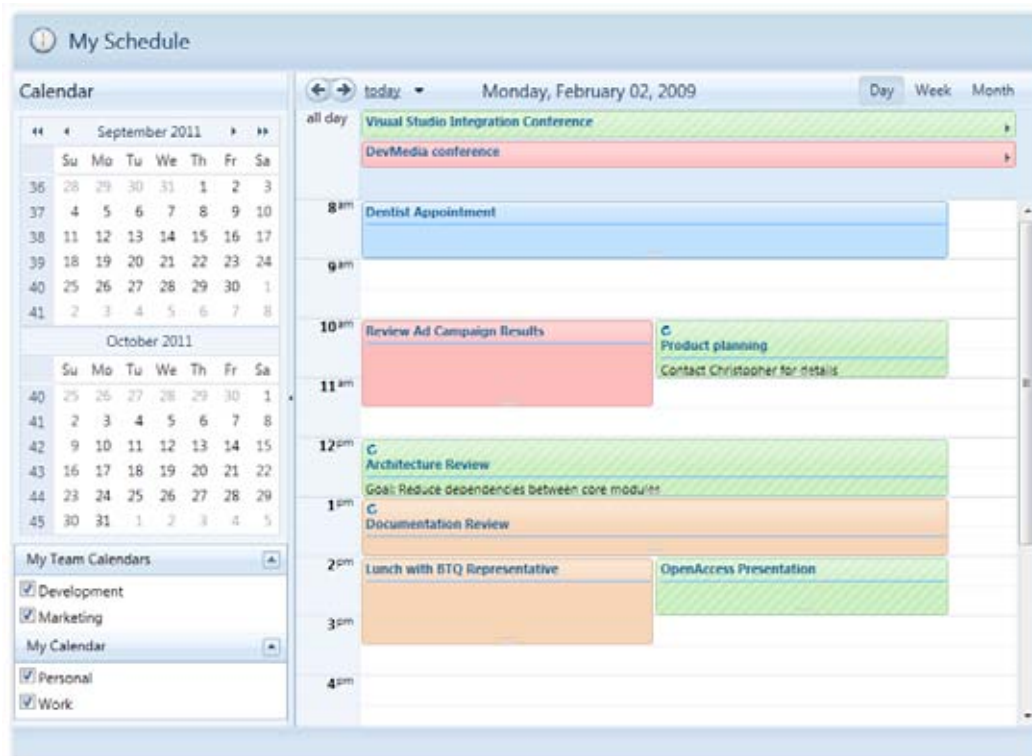
• RadScheduler

When picking a scheduler control, you should look for one that loads quickly, has a rich, interactive UI, and allows flexible binding to any type of scheduling data you have.

Page Load Time

Similar to RadEditor, RadScheduler for ASP.NET AJAX only loads the script files for the features that are absolutely necessary during the initial page load. Lazy loading is used to load additional scripts as they're needed, so the end result is a very rich scheduling component that does not impact negatively your page's initial loading time. RadScheduler's load time also benefits from the highly optimized HTML output, ensuring only the bare minimum HTML is rendered so your pages stay lightweight and fast.

[See RadScheduler page loading in action »](#)



Flexible Data Binding

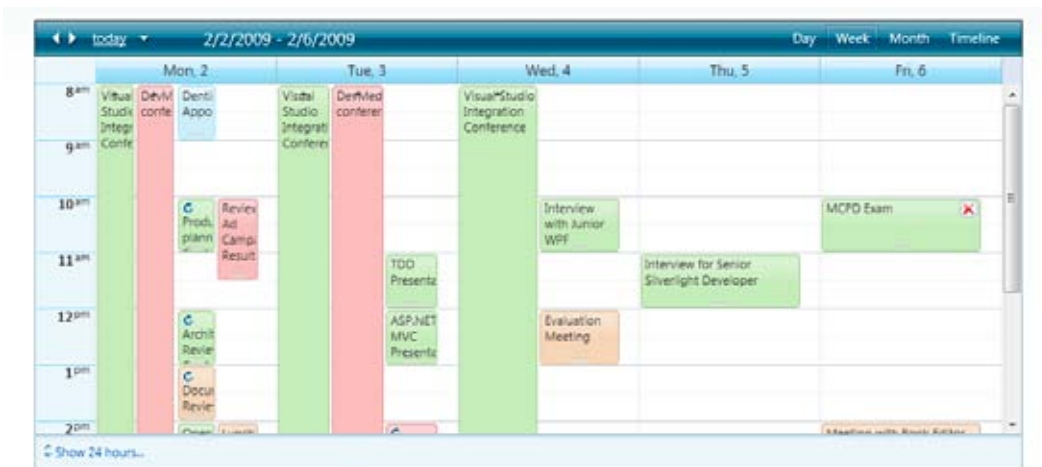
One of the biggest problems with most web-based scheduling components is that they require you to bind to objects or data sources that implement pre-defined patterns or interfaces. This is okay for new projects, but an absolute showstopper for existing development. Fortunately, with RadScheduler for ASP.NET AJAX you don't have to worry about where your data is stored. You can bind the Scheduler to any data source or object that contains your scheduling data and still take advantage of all of the rich UI features like drag-and-drop appointments and recurrence.

RadScheduler supports Web Service binding which allows the developer to improve performance by minimizing the HTML output and removing the need of a page refresh. In this mode RadScheduler does not postback to the page and the appointments are rendered on the client.

Furthermore, the Scheduler loads only the appointments that are "in view" from your data source. That means you only load data as it's needed by the Scheduler UI, significantly reducing potentially performance-crushing queries to your data source for data.

[See RadScheduler Web Service data binding in action »](#)

[See RadScheduler WCF Web Service data binding in action »](#)



[See RadScheduler data optimizations in action »](#)



RadScheduler by the Numbers

The following numbers were collected by loading a default RadScheduler on a simple page, bound to 10 appointment items displayed in Week View. The tests were conducted in IE 7.

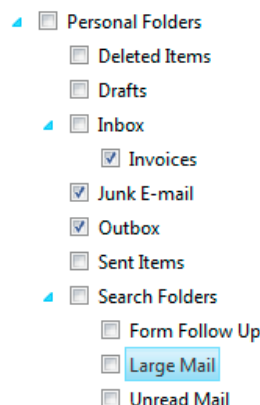
RadScheduler for ASP.NET AJAX	
Number of requests	420 ms
Number of requests	20 (w/o RadManagers)
JavaScript size	190 KB (Varies between view types)

When picking a treeview control, the most important factors to consider are rich client-side features and optimized performance when dealing with large node collections.

Rich, client-side features

TreeView controls are a key UI concept in all major operating systems, and as such users - and developers - have a certain level of expectation about what a tree should be able to do. RadTreeView rises to the challenge by supporting all of the rich interactions users expect from a tree in a standards-based, XHTML-compliant UI component. Things like node drag-and-drop (in the same tree, between trees, even between trees and grids), node editing, and node context menus are supported right out of the box. All of this, of course, while rendering highly optimized HTML and loading lightning fast on the page.

[See RadTreeView client-side features in action »](#)



Page Load Time

The most common performance problem trees encounter on the web is dealing with large node collections. Telerik's engineers recognized this problem early on and provided [a number of "load-on-demand" modes](#) to loading nodes as they are needed. For top performance, developers can leverage the RadTreeView's built-in support for web services that enables easy codeless binding to fast and efficient services. With this approach, developers can load nodes as needed without transmitting ViewState and without executing the full ASP.NET page lifecycle, meaning nodes load with small, unnoticeable delay.

[See RadTreeView web service optimization »](#)

RadTreeView (as well as most of our ASP.NET AJAX controls) renders some JavaScript code in JSON format required to initialize the nodes. At the time being the text of the node is not serialized in by default which saves a lot of output.

RadTreeView by the Numbers

To collect these numbers, a simple page with a RadTreeView using default settings is bound to 1,000 nodes (10 x 10 x 10), with all sub-nodes loaded on demand via web services. Tests were conducted in IE 7.

RadTreeView for ASP.NET AJAX	
Avg. Page load time	470 ms
Number of requests	23 (w/o RadManagers)
JavaScript size	180 KB
Time to load sub-nodes	70 ms per node

• RadComboBox

Combobox controls deliver more functionality than the standard HTML select or ASP.NET DropDownList can provide, so it's most important to look for a control that supports flexible rendering, while not significantly impacting page load performance.

Flexible Semantic Rendering

More often than not, you'll use a rich combobox control like RadComboBox when you need to render a drop down that supports more than simple string values. That means you need a tool that can render any HTML you want in the drop down list - and that's exactly what RadComboBox allows you to do. From multi-column support to template support, anything you need from a drop down can be rendered in RadComboBox.

All this flexibility comes packaged in a control that renders lightweight, semantic HTML for optimum page performance and SEO. RadComboBox renders as an unordered list in the HTML, which means no nested tables are used to build the controls UI (a common disadvantage of other combobox controls).

[See RadComboBox flexible rendering in action »](#)



Page Load Time

Similar to the tree controls, the biggest problem comboboxes face is handling large collections of values. Browsers simply have limits on the number of HTML items they can process on a page before memory problems begin to seriously harm the page performance. To help fight those issues, RadComboBox supports several built-in "load-on-demand" binding modes that enable you to only load list items as they're needed. And just like RadTreeView, RadComboBox supports direct binding to web services for super optimized, JSON updates. It's still possible to hit browser memory limits with these optimizations, but RadComboBox makes it much easier to avoid those problems with smart loading of your data.

[See RadComboBox optimized web service binding »](#)

RadComboBox by the Numbers

The numbers below were collected by testing a simple page with a RadComboBox for ASP.NET AJAX bound to a web service that returns 500 values when the combobox is expanded. Tests were conducted in IE 7.

RadTreeView for ASP.NET AJAX	
Avg. Page load time	58 ms
Number of requests	22 (w/o RadManagers)
JavaScript size	188 KB
Time to load list values	203 ms

• RadMenu

RadMenu supports [web service load on demand](#) and lazy initialization. Additionally, RadMenu can seamlessly work with disabled ViewState.

• RadTabStrip

Having lots of page views inside RadMultiPage can slow down the switching between tabs. What is more, it generates big HTML output because of the controls contained in the pageviews. To tackle this problem we have an [online example](#) demonstrating how to load pageviews on demand via AJAX. The multipage also has a property "RenderSelectedPageOnly" which does exactly what it says. In this case switching to a new page view requires postback.

• RadToolTip

RadToolTip and RadToolTipManager are lightweight controls which generally cause no performance problems. However, in templated scenarios the number of tooltip controls on the page can easily go out of hand. We have seen scenarios involving 1000+ tooltips on a single page. Since each of them needs to be initialized on client page load, the system takes a lot of time to do it, especially if `<compilation debug=true>`. In such scenarios there is a better approach to the tooltips and that is using a couple of lines of client-side code that will create a tooltip only when the user needs to see it. The [following demo](#) demonstrates this approach.

• RadCalendar

RadDatePicker, RadDateTimePicker and RadTimePicker

Having many date pickers on a page might render too much HTML and impact performance. RadDatePicker instances can share a RadCalendar control and use it to pick dates.

[See an example of RadDatePicker sharing calendars in action »](#)

When you have many RadTimePicker and/or RadDateTimePicker controls on a page, this might render large amount of HTML and impact performance. For this case, to reduce the html and speed up the loading time, RadTimePicker/RadDateTimePicker controls can share an instance of TimeView control for picking times.

[See an example of RadTimePicker/DateTimePicker sharing TimeViews in action »](#)



• RadInput

RadDateInput, RadNumericInput, RadMaskedTextBox and RadTextBox

Performance problems can be caused by using several instances of RadInput textboxes. In case your scenario requires a lot of textbox controls on the page, it is recommended to use RadInputManager for better performance. Integrating RadInputManager with RadGrid, for example, will significantly decrease the input editors loading time since plain Microsoft TextBoxes will be created instead of the corresponding RadInput controls and the data entered by the end user will be automatically filtered by RadInputManager based on the input manager settings.

[See RadInputManager integration with RadGrid in action »](#)

Additional Performance Optimization Resources

Every tool that Telerik builds is designed with performance maximization in mind. [Telerik's documentation](#) features many articles focused specifically on helping you optimize your RadControl implementations.

For more optimization tips, check out these articles from the Telerik Optimization Tips blog series:

Optimization Tips Series (blog posts):

- [The RadManagers for ASP.NET AJAX](#)
- [RadCompression Module](#)
- [RadInput vs. RadInputManager](#)
- [Testing Page Performance](#)
- [Using RadAjaxManagerProxy Controls](#)
- [Using HTTP Compression](#)
- [Optimizing Custom Skins](#)