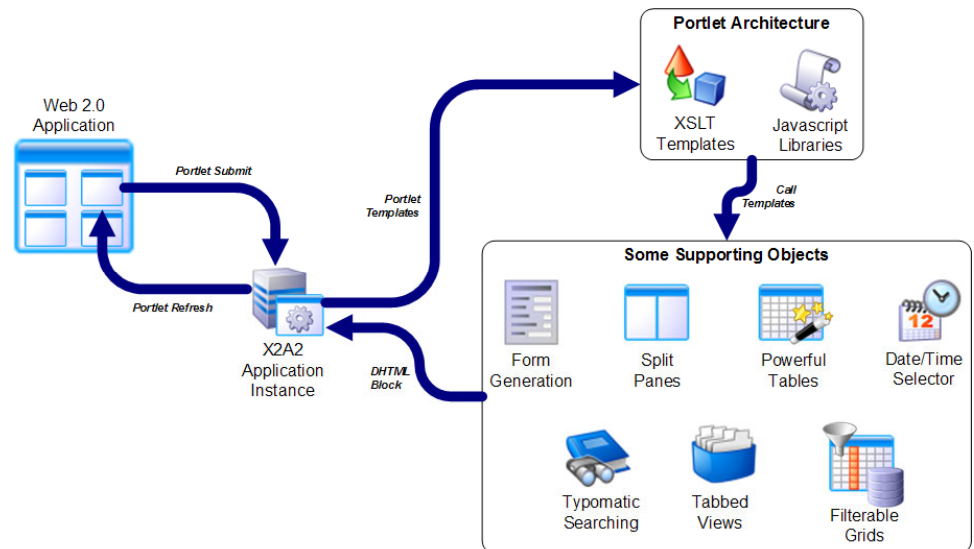


# Portlets

*“Vision is the Ability to See What is Possible Before it Becomes Obvious.”*

*-RelWare Development*

## Put the Rapid Back into Your Application Development by Using the X2A2 and Our Portlet Architecture



### RelWare Solution

The Portlet Architecture is a small Javascript library paired with a robust XML/XSLT combination that provides a basis for building AJAX/Web 2.0 applications.

We took years of website and enterprise Web application building experience and created a methodology for developers to transform data quickly from any source into a single, unified user-experience.

Further, over the past 5 years, we have built a comprehensive library of additional XSLT templates and supporting Javascript libraries that turn complex Web functions into a simple call of a template.

Customization was considered a core requirement at every level of our architecture. With the flexibility of our libraries and the abilities of CSS and HTML, we are turning Web applications into art instead of science.

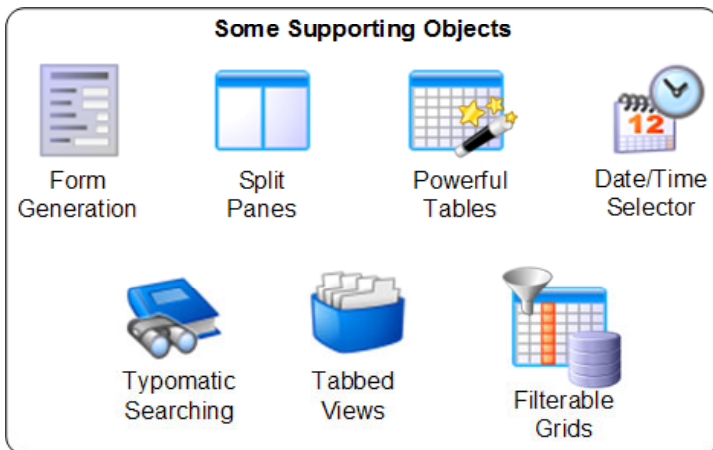
### History

RelWare has been building enterprise Web applications since 1998. We incorporated AJAX technology into our application in 2001. As we continued to build applications for the Web, we tried various, different approaches and techniques.

In 2003, we came up with a patent-pending methodology for rendering webpages or portions, named the X2A. Then, again in 2005, we rewrote this as a full-blown application engine, the X2A2.

At that time, we took all of our lessons-learned, and designed the Portlet Architecture. In conjunction with the X2A2, this is the base platform for our web-based enterprise electronic health record application.





### Supporting Objects

Our library of supporting objects started with some standard, simple, "core" Javascript functions, which were the basis for any DHTML manipulation.

From then on, as we have continued to build more complex applications, we have made a point to determine what new, generic component is required to add any new feature to the application.

This stance has provided us with a library of generic, yet highly customizable objects that have evolved into easy-to-use, yet very flexible object types.

After the Portlet object, the first supporting object that we built is called a "QuickTable". At the time, it was simply a quick way to generate a table from a given dataset. The point was to allow a developer to use any XML dataset composed of rows (*child nodes*) and columns (*attributes*) to generate an HTML table quickly. We also wanted to give the developer the ability to customize, easily, the look, feel, and indeed, features of the table. The QuickTable object today allows for paging of its data, sorting by column, nested children, and asynchronous loading of child data, amongst numerous other features. Each of QuickTable's features can be enabled or disabled, or simply not specified, and QuickTable will decide.

### Supporting Objects (cont.)

After QuickTable, we quickly began adding other standard application objects. At first, we expected that we would be looking to duplicate many of the concepts that we see in a standard development environment. Obviously, this was not necessary for a majority of those items, as they already exist in HTML (*e.g. checkboxes, and text fields*).

What we found was more of a need for complex objects that performed a seemingly simple concept. For instance, Tabbed Views are a concept with which we are all familiar. However, the ability to turn this into a reusable object has allowed us to add tabs to any Portlet, regardless of whether they are used in a "full-screen" view, such as a Home page, or in a Wizard-style within a "popup".

We continued to expand our library, adding as we have found needs within the context of building Web applications. Some of these, like the "splitter", make the addition of a "slider" bar for pane splitting as simple as a template call. Others, like the Super Calendar, provide a unified mechanism for collecting data from the end user. Again, all objects are highly-customizable, allowing the developer to change the look or even features completely from one instance to another.

While the above list is short, it is also far from complete. These are just some of our more popularly used supporting objects.

### RelWare's X2A2

The X2A2 is a lightweight, yet powerful application engine that is the core of our Web and workflow engines. It was built using XML from the ground up, and has been RelWare's primary development platform for the past 5 years.

RelWare's constant commitments to technology and excellence arrived on this patent-pending programming methodology in 2003. We realized then that XML was the data medium of the future, and had already known that the Web would continue to be the delivery mechanism.

The X2A2 merges these into a powerful, flexible engine, with built-in and external connectors that give them the ability to do almost anything.

### About RelWare Technology

At RelWare, we realized in 1998 that the Internet, and specifically, Web-based technology was the future. We built our company with the continued motto: "Every application is a Web application." This meant that anytime we looked at writing a new application, we first asked ourselves, "Can I write this as a Web application?" What we soon found was the answer was invariably "Yes" . . . every time.

It has been 10 years since we started, and we are still saying "Yes" . . . every time.

