

The Jaxcent Java-AJAX Framework

Jaxcent is a Java AJAX framework. This paper outlines the philosophy of Jaxcent as a Java AJAX framework, and advantages of Jaxcent.

Learning, Learning, Learning...

The idea is to avoid learning.

That may not sound very wholesome. After all, learning is GOOD. As programmers, we all appreciate learning new and cool methods and technologies.

Yes, learning can be good. But there is only so much time and so much to learn. Face it, everything we learn is not going to be rewarding. Those who have been in the industry for a while can undoubtedly recall time they spent learning things that were never useful, or at least not for long. Surely that time could have been put to better use?

That's the attraction of Jaxcent. Because Jaxcent is a very thin minimalist AJAX wrapper, you have practically nothing to learn.

Not totally – if you don't know HTML and don't want to learn it, Jaxcent may not be for you. But if you do know HTML, Jaxcent will not require much additional learning, because its classes that work with HTML, well, they simply correspond to the HTML tags! There is a class for `` and a class for `` and a class for `<P>` and a class for `<TR>` and so on... To work with a tag in your visitor's page in their browser, you simply grab the right class, instantiate an object and identify it on the page, and then change its contents, attributes and styles as much or as little as you need.

But if you don't know HTML and need to learn it, there is something you can bank on. HTML is here to stay.

Flexibility and Freedom

The Jaxcent approach of letting you play with HTML is in marked contrast to the “widget” or “special tags” based AJAX frameworks, where the framework tells you what your page should look like, and how you should code it.

It certainly can be convenient when somebody gives you a few well-made beautiful looking page design choices, and tells you what and how to program.

But there is a problem with that kind of approach – there are many different ways to do things in the web world. If what you want to do doesn't fit into your AJAX widget set, or your “special tag” set, you are out of luck. Then you will be spending large amounts of time just trying to figure out how to work around your widgets.

An open framework like Jaxcent, on the other hand, gives you raw access to HTML. The price is that you have to design the HTML page yourself. Or given that designing

beautiful pages is not always Java programmers' best strength, you have to get the HTML designer of your project involved when it is time to make the page look beautiful.

But the advantage is that you are not ever going to be stuck because of the framework. HTML is the basic technology of the web, and Jaxcent gives you access at that level.

Open Framework

The Jaxcent approach also protects you from “future shock.” As the web grows, undoubtedly hot new user interfaces will be created. With the widget-set based Ajax frameworks, you will always be depending upon the vendor for the upcoming new rev. Some frameworks pride themselves on having “hundreds of widgets.” But unless a time machine is involved, those “hundreds of widgets” are simply variations of UI designs invented so far! Of course, as new UI designs come along, the vendor can be expected to be adding those in. But still, the framework vendor will be the single source for new features. If the framework is open-source, you may be able to modify the framework source yourself – but this is not usually an easy or recommended way to integrate new features, and leads to breaking away from the mainline.

Because in Jaxcent the framework design itself is open, new and third-party user interfaces can be directly integrated with Jaxcent by the Jaxcent programmer, without any need to modify the framework.

Check out the Jaxcent samples pages to see how easy it is to take an “AutoSuggest” example right off a web tutorial, and hook it into Jaxcent. Or to integrate calendar or menu widgets built by third-party JavaScript houses, into Jaxcent.

No JavaScript / JavaScript Independence

Programming in Jaxcent requires no JavaScript knowledge or programming. There is only a single static JavaScript file involved, and all that's required is to add an “include” in the HTML content.

This is a huge benefit. JavaScript is a scripting language. It is excellent for short and cool program fragments. But JavaScript cannot be organized or maintained or debugged well, for much serious programming. For professional programmers who do not specialize in JavaScript, avoiding JavaScript whenever you can, is good advice.

Of course, sometimes delving into JavaScript becomes necessary, much as in an earlier generation, serious programmers had to occasionally delve into assembler languages.

Jaxcent's independence from JavaScript means that if you do want to use some cool JavaScript you have gotten from a third-party JavaScript house (or written yourself,) Jaxcent will not get in your way. You can add JavaScript to your page entirely independently of Jaxcent.

In addition, Jaxcent has JavaScript hooks, so if you do want to connect your cool new JavaScript to Jaxcent, it is very easy to do so, and the framework is very open to integration.

Incremental Nature

Jaxcent can be added incrementally. You can take a fully working web application, pick a page out of it, and Jaxcent-enable that single page without affecting anything else in the whole application. That single page can still continue to participate in the old application, except that with Jaxcent it can be more dynamic. You can stop with the single page and a little dynamic action, or you can then slowly evolve your entire application to be an AJAX application.

Same incremental approach can be used with issues like JavaScript integration. You can do all data verification from Jaxcent. You can do all data verification from JavaScript. Or you can move single pieces back and forth as best suited.

Ease of Use

Programming in Jaxcent requires no complicated XML descriptions. Building a single page in some AJAX frameworks can be a monumental task. In Jaxcent, you have to do three basic tasks (a) add a single JavaScript include statement in the HTML, (b) tell the framework the name of the Java class that will be handling that HTML page, and (c) write the Java class.

Server-Side Framework

Finally, perhaps the biggest strength of Jaxcent, from the point of view of Java programmers, is that it runs on the server side.

When it comes to **Java-only** AJAX frameworks, there is an important design trade-off. AJAX Java frameworks can be compiled to JavaScript, and run on the browser. This is client-side. Or they can run on the server, and communicate with the browser. This is server-side.

You may think that running the AJAX on the server side means you are putting extra load on the server. But remember, AJAX is not JavaScript. If you are doing anything useful in AJAX, it must involve server computations by definition! Little performance enhancement is to be gained by having your main loop on the client instead of on the server. Server side does mean that you will be tempted to do things on the server that actually did not require any server computation. For instance, making sure fields are not empty, or integer fields contain integers, and so on. But such small temptations are very convenient during initial development, and if the performance starts to be a problem, these things can easily be moved to JavaScript.

Server-side AJAX does have a very significant advantage as far as programming is concerned. It means you can use all those tricks and techniques of Java in your arsenal, that you have come to learn and rely upon all those years. You can start Java threads that update the web page as and when they need to. You can access local databases and files

using the same coding you always have. You can access your local intranet from your AJAX program if you so desire.

You do not have to worry about security, because all the database and file access is happening locally, without any involvement at all from the browser side. There is absolutely no possibility that you will inadvertently put some SQL or a table name or a file name or the program logic on the client, because your code never gets compiled into JavaScript or otherwise gets placed on the client! It always stays secure on the server. You need invest no extra precaution on security beyond the standard ones.

The entire Java environment, including any libraries that you may be using, continues to be available to you in Jaxcent.

All this is not likely to be the case in client-side Java AJAX. A simple task such as updating the display from a thread, is not so simple, never mind access to any specialized JDBC library you might have... Client-side Java AJAX is still Java technically. But realistically, it is a new programming environment to learn.

But the server-side AJAX of Jaxcent means you have to learn no new programming styles. You can use the same kind of programming as you have always been doing. To listen to events in the browser, you simply override methods. To modify page contents, you simply call the appropriate method. Everything else stays exactly the way you are used to. The browser simply becomes an extra API, rather than a whole new way of programming.

As far as project teams are concerned, staying within programmers' expertise and comfort zone, and not requiring a lot of new learning and heavy frameworks, directly translates into:

- Faster Time to Delivery
- Better Designs
- More Reliable Code and Fewer Bugs

In today's programming environments, new "standards"-of-the-day keep coming out every few months. Most of these "standards" are forgotten the very next year, some stay on to become very useful. It can be hard to know which standards to invest time and effort on. In this demanding environment, if you can access a new and useful technology like AJAX without investing a lot of new learning, why pick some other framework and spend a lot of time and effort learning how to do the same things or less, but in a whole new and hard way? That's why Jaxcent is the sensible choice for Java AJAX programming.