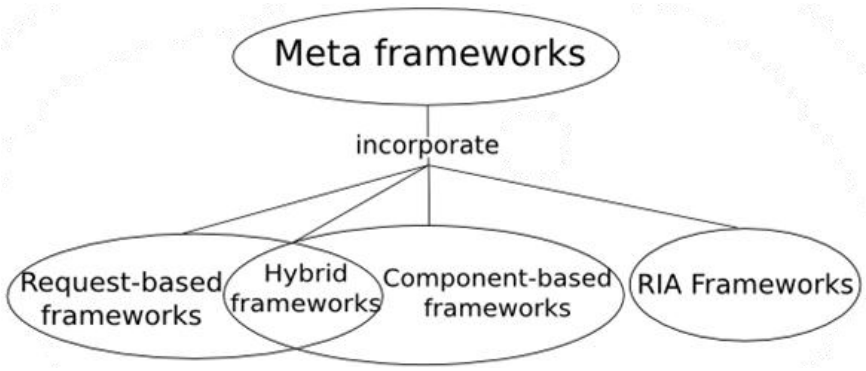
Choosing a Web Framework

for the IXP platform as a replacement of wingS

To come straight to the point: there is no best web framework, just a lot of awesome choices. The actual challenge is to choose the right tool for the right job. And this is not an easy one regarding the plethora of frameworks available. Wikipedia mentions and roughly compares a few of them here.

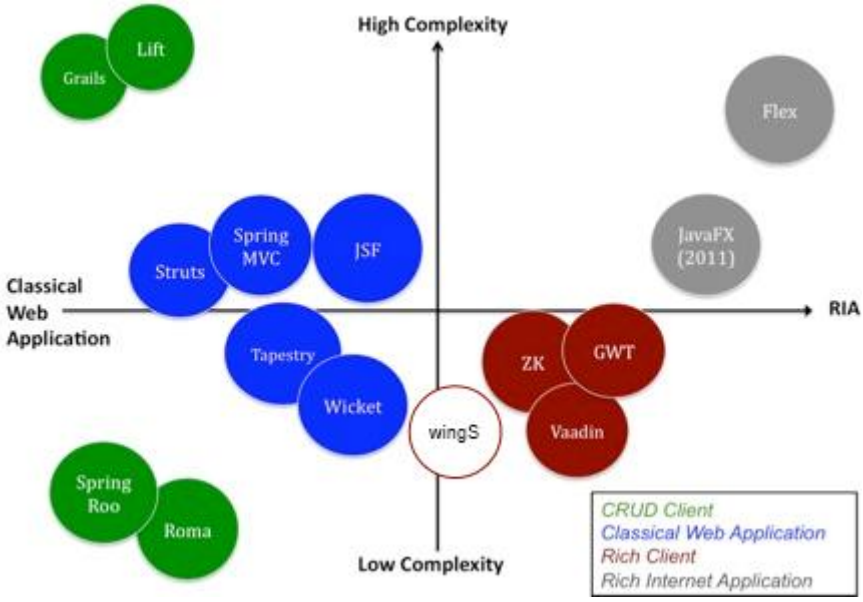
In order to narrow down the choice, let's start by categorizing web frameworks. Basically there are five different types: Request-based, Component-based, Hybrid, Meta and RIA frameworks.



- Request-based frameworks: The framework directly handles incoming requests. Each request is essentially stateless but with server-side sessions, a certain degree of statefulness can be achieved. Examples: Django, Ruby on Rails, Struts, Grails
- Component-based frameworks: The framework abstracts the internals of request handling and encapsulates the logic into reusable components. The state is automatically handled by the framework. Together with some form of event handling, this development model is very similar to the features of desktop GUI toolkits. Examples: wingS, Vaadin, ZK
- Hybrid frameworks: The framework combines both, request-based and component-based frameworks, by taking control of the entire data and logic flow in a request-based model. Developers have full control over URLs, forms, parameters, cookies and path infos. However, instead of mapping actions and controllers directly to the request, a hybrid framework provides a component object model that behaves identically in many different situations such as individual pages, intercepted requests, portal-like pages fragments and integratable widgets. Components can be wired together and be packaged as groups that are components in their own right. They can be distributed separately and be seamlessly integrated into other projects. This combines the form of reusability in component-based frameworks with the raw controls of a request-based approach. Example: RIFE
- Meta frameworks: The framework has a set of core interfaces for common services and a highly extensible backbone for integrating components and services. A meta framework is close to a framework of frameworks. Example: Keel
- RIA frameworks: The framework for developing Rich Internet Applications (RIA). RIA refers to a web page-based application running in a browser with rich user interface features which are common in desktop application such as drag and drop. Generally, a plugin is needed to run the application. Example: Flex

Most common web frameworks are either request-based or component-based. While a request-based framework is the natural fit for simple CRUD (create, read, update, delete) clients or classical web applications, a component-based framework is the right choice when developing a so called rich client that resembles a desktop application. In contrast to RIAs, a rich client does not need a plugin to satisfy its requirements but tries to create a better user experience by using plain HTML and AJAX.

Taking into account this typology, some well-known web frameworks can be distributed according to the following graphic where complexity means difficulty to learn the framework for a Java developer.



Regarding the requirements of the IXP platform, this helps us to eliminate most of our options when it comes to „choosing the right tool“. In fact – since we need to build a rich client web user interface consisting of forms with various components and want to leverage our server side Java skills to the maximum – the only remaining frameworks are wingS, Vaadin, ZK and GWT.

wingS is currently the framework used for the IXP platform. But because it is not under active development anymore, an adequate replacement should be found in the medium term.

Vaadin is probably the closest alternative to wingS and has therefore been the framework of choice for porting existing and developing future Java web applications at the Wilken Group.

Even though we don‘t have any personal experiences with ZK at the Wilken Group, looking at the framework’s architecture and feature set, this seems like another viable replacement option.

We don’t recommend using plain GWT for the IXP platform because compared to ZK and Vaadin (which uses GWT on the client side) GWT is more low-level and therefore slightly more complex.

Another future requirement of the IXP platform is to replace the current polling mechanism with a state of the art server push infrastructure. During a diploma thesis, wingS has been extended with comet support. However, the development efforts in this area never left the experimental stage, are meanwhile outdated and as such definitely not recommended in productive environments. In their current versions Vaadin as well as ZK provide basic server push functionality out of the box. GWT, in contrast, does not have a comparable core feature but needs to be extended with a dedicated third-party library like gwteventservice. So at the end of the day each promoted wingS alternatives has

basic support for actively pushing events to the client. Depending on the actual requirements of the IXP platform and its runtime context this might be sufficient. If not, e.g. because you need to support more server environments or want to provide better fallback mechanisms, we recommend to have a look at the Atmosphere framework. During the last few years the latter established itself as kind of the de facto standard when it comes to server push in enterprise Java applications.

In order to sum it up, you should try to replace wingS as soon as possible. Even though it might do the job at the moment, you don't want to base a product on a framework which is already dead for more than three years now. As a viable alternative for wingS we suggest either Vaadin or ZK. Which one you eventually choose is probably a matter of personal taste. Their feature set is more or less comparable and in any case big enough to supersede your wingS application. When it comes to grids ZK seems to provide the advanced features we often missed in Vaadin's table component. But for other components it might be the other way round. In order to minimize undesired surprises in an advanced transition level, we therefore recommend the following: list your goals in terms of required components and features, weight the importance of each goal and evaluate how likely each of the two frameworks is to meet your goals. Our experience has shown that writing a simple prototype based on each option touching (not necessarily solving) the main goals is the best you can do in advance to ease or confirm your decision.

By the way, Matt Raible held a quite good presentation at Devoxx France 2013 about „Comparing JVM Web Frameworks" including strategies for choosing and research results from various related Java communities. Unfortunately his framework matrix (see here, here and here) doesn't include ZK.

Criteria	Struts 2	Spring MVC	Wicket	JSF 2	Tapestry	Stripes	GWT	Grails	Rails	Flex	Vaadin	Lift	Play
Developer Productivity	0.50	0.50	0.50	0.50	1.00	0.50	1.00	1.00	1.00	0.00	1.00	0.50	1.00
Developer Perception	0.50	1.00	1.00	0.50	0.50	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Learning Curve	1.00	1.00	0.50	0.50	0.50	1.00	1.00	1.00	1.00	1.00	1.00	0.50	1.00
Project Health	0.50	1.00	1.00	1.00	1.00	0.50	1.00	1.00	1.00	0.50	1.00	1.00	1.00
Developer Availability	0.50	1.00	0.50	1.00	1.00	0.50	1.00	0.50	1.00	1.00	0.50	0.00	0.50
Job Trends	1.00	1.00	0.50	1.00	0.50	0.00	1.00	0.50	1.00	1.00	0.00	0.00	0.50
Templating	1.00	1.00	1.00	0.50	1.00	1.00	0.50	1.00	1.00	0.50	0.50	0.50	0.50
Components	0.00	0.00	1.00	1.00	1.00	0.00	0.50	0.50	0.50	1.00	1.00	0.00	0.00
Ajax	0.50	1.00	0.50	0.50	0.50	0.50	1.00	0.50	0.50	0.50	1.00	1.00	0.50
Plugins or Add-Ons	0.50	0.00	1.00	1.00	0.50	0.00	1.00	1.00	1.00	1.00	1.00	0.50	1.00
Scalability	1.00	1.00	0.50	0.50	0.50	1.00	1.00	0.50	0.50	0.50	0.50	1.00	1.00
Testing	1.00	1.00	0.50	0.50	1.00	1.00	0.50	1.00	1.00	0.00	0.50	0.50	1.00
i18n and l10n	1.00	1.00	1.00	0.50	1.00	1.00	1.00	1.00	0.50	0.50	1.00	1.00	1.00
Validation	1.00	1.00	1.00	0.50	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.50	0.50
Multi-language Support (Groovy / Scala)	0.50	0.50	1.00	1.00	1.00	1.00	0.00	1.00	0.00	0.00	1.00	0.00	1.00
Quality of Documentation/Tutorials	0.50	1.00	0.50	0.50	0.50	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Books Published	1.00	1.00	0.50	1.00	0.50	0.50	1.00	1.00	1.00	1.00	0.50	0.50	0.00
REST Support (client and server)	0.50	1.00	0.50	0.00	0.50	0.50	0.50	1.00	1.00	0.50	0.50	0.50	0.50
Mobile / iPhone Support	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.50	1.00	1.00	1.00
Degree of Risk	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.50	0.50	0.50
Totals	14.5	17	15	14	15.5	14	17	17.5	17	13.5	15.5	11.5	14.5

For the sake of completeness: In case Java is not a limiting factor for your framework choice, you might also want to consider component-based JavaScript frameworks like ExtJS and KendoUI. However, be aware that compared to a server-side Java framework like Vaadin or ZK the transition from your current wingS application will be considerably harder and way more time expensive.

ExtJS is one of the major players in the field of JavaScript frameworks for years now. It is extremely stable and pretty comprehensive regarding its feature set. Besides wingS and Vaadin, ExtJS is the third framework on which we based a whole product line at the Wilken Group.
KendoUI is a relatively new JavaScript framework which is massively pushed by Telerik (a well-known provider of development tools for .NET) since Microsoft declared HTML5 and JavaScript as their key technology for building native Windows 8 apps. Chances are that the frontend of our new product line called Wilken S4 will be built upon KendoUI.