What are the fundamental differences between Struts and JSF

Will Hartung's thoughts from a similar question posted on TheServerSide (*posted here in its entirety with permission*):

> *Routing around the hyperbole and actually trying to answer the question...*
>
> *The key differences between the two are the base paradigms that underly each platform.*
>
> *Specifically, JSF is a "component" framework whereas Struts is an "action" framework.*
>
> *What does that mean?*
>
> *In a component framework, artifacts that are rendered on the page are initially developed as individual components, much like in modern GUI "fat client" libraries. You have components, they have events, and your code is written to work with those events against the components.*
>
> *Most of the time, in mainstream development, your code is pretty much ignorant of the HTTP request cycle and processing.*
>
> *Struts (both 1 and 2) are action frameworks. In essence they give you the ability to map URLs to activities and code on the back end. Here, the layout and workflow tends to be more page oriented. As a developer you tend to interact with the HTTP request cycle directly, though Struts 2 helps isolate at least the binding of the request data to the action implementation classes.*
>
> *Action frameworks tend to be much "thinner" in how they stand between your code and the raw HTTP request compared to component frameworks.*
>
> *For those people just cutting their teeth on web development, the component frameworks (especially with good tooling) can be much more approachable, as the tools tend to hide the heavy weight nature of the component frameworks. But, on the downside, their size and their "square peg in to round hole" nature of turning HTTP requests in to the rough equivalent of mouse clicks mean there's more complexity to try to understand if and when the framework starts misbehaving or getting in the way.*
>
> *But, to be fair, frameworks like ASP.NET and JSF are popular because novices can get quick success with them with the modern tools.*
>
> *Old School HTTP wranglers may simply be more comfortable with action frameworks, simply because they've been through the crucible of understanding how HTTP requests are structured. Action frameworks tend to work better with "pretty" urls (though component frameworks are getting better at his). Action framework coders can have more control of the structure and presentation of URLs, since their systems are more intimately tied to them compared to a component framework.*
>
> *As a basic guideline, I find the action frameworks are better for "web sites", site like this one, sites that focus on delivering content to the user. Where it's mostly a "read only" experience for the end user who is likely to want to bookmark things, come back to arbitrarily deep pages, etc.*
>
> *But the component frameworks are better for "web apps". CRUD screens, back office applications, lots of forms and controls, etc. Here, the workflow is more controlled. You tend to not get to a "detail" screen with going through the "list" screen or "header" screen first, for example.*
>
> *It's nice to be able to just drag and drop a grid component on to a form, add a couple of buttons and point it as a DB table to get "instant" results. But these apps don't bookmark well, HATE the "refresh button", may behave poorly with the back button, etc. Overall, they can not be very good citizens when working with the web browser.*
>
> *So, if it were me, I'd write a blog in a action framework like Struts 2 (or even better, Stripes), but I'd write an accounting package in JSF.*