Quo Vadis Maven Site

This page is about to draft some ideas about Maven Site capabilities, possibly to spin up some conversation about it's future.

Is Using Maven Site Simple?

Is not, it is full of "hidden gems" that are hard to figure out. The road from initial "site" goal to the "site that I want to end up with" is long, incomprehensibly complicated, full of pitfalls and possible breakages you have no idea why happen.

Is It Competitive?

No, and never was. Given it still uses 10+ years old technology (more precisely, uses unmaintained code that is 10+ years old, so uses even older tech), Maven Site capabilities will never be (and never was) competitive, when you compare it to other "static site" build tools (JBake, Hugo, and so on).

Should It Be Competitive?

No, it should not. HTML and all the related "web" technologies are evolving in lightning speed, just think about "responsive" mobile device optimized sites for example. Maven will never be able to keep up, nor is it's "thing" to do so.

More even, some reports like XRef lost their reason to exist, as 99.9% of source repositories provide "source browsing", even with syntax highlighting and cross linking.

Finally, having site release-cycle tied to source release-cycle is just plain wrong: You do a new release of unchanged sources for a site typo fix? Or new release just for new logo?

So Where Are We?

We should accept the facts: Instead to maintain a sh*tload of code base that is neither trivial, competitive, nor modern, nor is able to fulfill today's requirement, we should embrace today's standards and solutions. For start, we have to acknowledge there are superior dedicated tools doing static site generation. Using them (may) be simple and intuitive, while using Maven Site is neither of those two.

So What Should Be Done?

From here, it is my personal opinion, meant for discussion.

Maven should stop producing "site", but it should keep producing "reports", and depending on report (as for example Javadoc is rendered by javadoc tool into HTML), render them into some "intermediate" format, like AsciiDoc or Markdown (w/extensions) is. All the legacy (now "exotic") formats (used nowadays by Maven only, for example APT, xdoc) should be dropped.

Most, if not all, of the site related things are completely neglected for 10+ years. Just accept that we have no manpower to maintain it. Bigger problem is wasted engineering hours spent on it (currently only 1 person works on all the site related stuff), as in 10 years perspective, all we can do is adopt to latest dependencies/libraries, remove usage of deprecated libraries, but even with these changes we will not get anything "more modern" or responsive site generation, on final output. IMHO, we should instead:

- "Radically purify" (meaning: drop) related stuff, leaving Doxia Sink, that should be make able to produce AsciiDoc and Markdown (w/ extension, like tables) output for start. Am unsure is Doxia Sink API already able to do this, if not, we need to improve it. Having Sink API as "indirection" layer should serve the purpose that our reports are "future proof" in a way, if tomorrow new wildly popular format comes in play, all we'd need is just to add new Sink implementation for that format.
- We could make Maven Build somehow "collect" all the produced (intermediate) formatted reports. Javadoc naturally remains as is, is already a separate artifact in form of today "-javadoc.jar" (same would apply to some other tool that "renders on it's own") and is HTML not "intermediate". We could collect them into "-reports.jar" and have it deployed along javadoc, sources, etc.
- The "usual" pattern would be to have separate "site project" (this is today the case with almost all projects out there, except those "minimal" one person OSS ones maybe, but they can easily switch to JBake Maven Plugin to remain "one pass builds and produces site"), that uses whatever tool author decides (JBake, Hugo, gatsby) that simply fetches deployed "-reports.jar" and renders them.
- Maven should NOT render anything (again, naturally exclusions here are things like Javadoc tool, or any other possible tool that "renders on it's own")
- Drop some reports that are not useful anymore, drop exotic formats that are not wanted anymore.
- An independent (Maven or not) site project, that is able to consume maven project reports as resources (as is, for example the Javadoc rendered HTML) or as input (Markdown) and render final site? This is already doable today with jbake Maven Plugin: just download the deployed "-reports.jar" artifact, and render.

TL;DR

- Maven Site should be gone
- Maven should not render anything
- Reports should output (configurable, for start AsciiDoc or Markdown w/extensions) "intermediate" format only
- The intermediate formatted report outputs should be collected and deployed along other artifacts (for example "-reports.jar")
- User using their tool of choice should just pull the deployed reports, and render them along with other (in site) intermediate formatted pages to final site and deploy it