Redmine - Feature #35536

Use webpack to improve javascripts and stylesheets management

2021-07-06 12:00 - Takashi Kato

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
Resolution:	Wont fix		

Description

I propose the use of webpack as a library management tool for javascript. I also suggest the use of simpacker as a tool for linking rails and webpack.

The docker environment can be found here.

https://github.com/tohosaku/redmine-docker-wsl2

Motivation

- To deal with the problem that #6662 "js files are not compressed"
- To be a foundation for future npm package usage.

Related issues:

Related to Redmine - Feature #36320: Migrate to Rails 7.2	**Closed**
Related to Redmine - Feature #42510: Add Stimulus as a Javascript framework	**Closed**

History

#1 - 2021-07-06 12:03 - Takashi Kato

I'm going to post a few suggestions, since it seems that posts can't be too long.

Benefits

For users.

- webpack automatically minifies javascript and CSS.
- reduces the number of HTTP accesses by bundling files.
  - However, I do not know if there will be any visible performance improvement at this time.

For developers.

- Managing javascript as a module makes it easier to get help from editors and lint tools.
- webpack makes it easier to keep track of versions of javascript packages since we don't have to commit them directly to the repository.
- webpack-dev-server will support automatic reload for javascript modification during development.
- Image files in CSS have hash query string(for more accurate cache control).

#2 - 2021-07-06 12:06 - Takashi Kato

Questions that may be asked

- The default for Rails6 is webpacker. Why use simpacker instead of webpacker?
  - webpacker may help develop new rails apps, but it doesn't seem suitable for cases like Redmine where there is a lot of existing javascript that needs to migrate. I think it is more appropriate to use simpacker.
    - I believe it is appropriate to use simpacker because it is a thin wrapper and does not make administration overly complicated.
    - webpack was released ver5 last year, and it includes commonly used plugins and loaders, and there is also extensive documentation on the official website. it is faster to use the simpacker to control the webpack more directly.
  - This is a tool developed by an engineer of cookpad , the most famous rails user company in Japan. I hope this will be a good opportunity for you to get interested in it.

- Do all Redmine users have to install nodejs and yarn in addition to ruby?
  - No, only developers must install nodejs and yarn, and we expect to include precompiled javascript and CSS for users who install Redmine with zip, tar.gz, or other archives.
  - The process of compiling javascript must be interrupted at release time.

- Doesn't having javascript and CSS in one package degrade performance? Will it degrade performance to have javascript and CSS in one place, or will it make users download files they may not use?
  - webpack can split the output into multiple files instead of a single file. This patch follows the current javascript loading order. Instead of having a single file for all files, it generates a file for each location where the helper is currently loading javascript.
  - webpack is a tool to organize js into chunks, but language files of the jstoolbar, datepicker can be separated into different files instead of combined into one so that extra files are not loaded.

- If you use the npm module, the jquery will not be visible on the plugin side?
  - "expose-loader" exposes jquery, tributejs, etc. globally. It seems that there is no effect on plugins that expects Redmine to load jquery.
  - "rails-ujs" also expects that jquery is published globally (if not, jquery doesn't add CSRF protection functionality), but it works fine after patching.

- How does it affect the plugin and theme?
  - For compatibility with existing plugins and themes, the output path of compiled javascript and CSS is almost the same as the previous path.
  - simpacker has its helper (javascript_pack_tag, etc.), but we will use existing helpers because Redmine extends helpers for plugin and theme.
  - The following plugins have been installed and tested. I changed some of the file names used in javascript_include_tag, so we will need to modify that part.
    - UI Extension
      - The filename of "chartjs" has been changed, so it needs some modifications.
    - Issues Panel
    - Issue Templates

- Is it reasonable to use webpack as a bundle management tool?
  - rails use webpack.
  - Others such as "Parcel" , "Vite" , "Rome" , etc.
    - "vite" has a generous rails integration (https://github.com/ElMassimo/vite_ruby) feature, but it doesn't support jquery. (jquery needs a patch).

Roadmap

- Passing through loader to convert CSS and Javascript that doesn't work in some browsers, and to write SCSS (using loader such as babel).
- Use of unit testing tools such as mocha and npm packages such as eslint and prettier
- Separate public and private functions by modularizing javascript.
  - Currently, all functions in application.js are public.
  - Initially, all functions will be made public as before. After some refactoring, we may remove codes considered to be unnecessary to publish from export.

- I would like to introduce a stimulus . Currently, Redmine code has many javascript code fragments. But with stimulusjs, it will be easier to extract them.
  - Modify the ".js.erb" to pure javascript. (The amount is huge and pending this time)
    - If we can extract the javascript codes in the ".erb" files, we can get help from editors and lint tools (migrate from old notation, remove browser support that is no longer needed).
  - DOM traversal will be less frequent, so dependency on jquery will naturally decrease (but I don't want to de-jquery).
  - Helper methods such as link_to_function deprecated in rails, have some fragments of javascript code. I want to extract them into a javascript module.

Note

- When webpack compiles CSS by itself (called setting it to entry point) instead of accompanying javascript, it outputs unwanted javascript files. There is a plugin that removes unneeded javascripts. But it is not considered to be harmful, so I will not take any action.
- Simpacker refers to the manifest.json generated by webpack to get the information of various assets. During development, if manifest.json is missing even after starting Redmine, an exception will occur.
- All developers need to install nodejs.

#3 - 2022-01-05 07:57 - Marius BĂLTEANU

- Related to Feature #36320: Migrate to Rails 7.2 added

#4 - 2022-01-05 08:00 - Marius BĂLTEANU

I'm interested in improving the way in which we manage now the assets, but from what I have read, Rails 7 offers an alternative to webpack, let's discuss this when we do the update.

#5 - 2023-09-22 13:55 - Dmitry Lisichkin

Marius BALTEANU wrote in #note-4:

> I'm interested in improving the way in which we manage now the assets, but from what I have read, Rails 7 offers an alternative to webpack, let's discuss this when we do the update.

Alternative approach can be used in Rails 6 too (mostly) and can use webpack or other builders.
Main changes of last decisions is an obsoletion of webpacker gem that provide javascript_pack_tag.

For redmine main problem is plugins and themes that **should** be in same pipeline.

I'm working with redmine from 2013 and we use own monkey patch solution based on sprockets
https://github.com/Tab10id/redmine_plugin_asset_pipeline
We use it many years but this solution has too many limitations. So we decide to replace sprockets to actual tools while update our plugins to newer redmine version.

Some information:

- actual compile tools should compile assets to app/assets/build directory and than should be served (or final compiled) by another tools
  - compilation to app/assets/build can be done by many solutions like webpack, esbuild, rollup, bun, etc with jsbundling-rails (Rails 6+)
  - final compilation or serving can be done by sprockets or propshaft (Rails 7+)

After that all we need is to select solution for compile all redmine assets and plugin assets to app/assets/build.

Problems

- Webpack, esbuild, rollup, bun and others is not supposed to be used for compilation of ambiguous numbers of assets (in plugins)
- Yarn, npm also not suppose to work in such context.

Solution

We can compile every plugins individualy and than move assets to individual directory in app/assets/build
(app/assets/build/plugin_assets/plugin_name/....)
But in this case plugins can't share asset libraries so if several plugins will use select2 than we load select2 for every plugin that need it.

Tricky solution

For yarn and npm we can use workspaces https://classic.yarnpkg.com/lang/en/docs/workspaces/

```
// package.json
"workspaces": [
  "plugins/*"
]
```

This allow us to share libraries between redmine and plugins and to have single package-lock.json for all js libraries.

Also we need to compile at runtime single config for selected tool.
something like this:

```
// webpack.config.js (or any other tool config)

const pluginConfigFiles = globSync("plugins/*/path/to/config.js")
const pluginConfigs = pluginConfigFiles.map(function (item) {
    let pluginConfig = require(`./${item}`)
    // ... optional modifications ...
    return pluginConfig
})

configs = [globalConfig, ...pluginConfigs]

module.exports = merge(configs);
```

In our case I use webpack as it already have library for config merge https://www.npmjs.com/package/webpack-merge but it not hard to use same approach for other tools.

#6 - 2023-09-22 16:19 - Dmitry Lisichkin

Fun fact: jsbundling-rails is simple tool for install another tools (webpack, esbuild, etc) and only provide rake tasks as wrappers for that tools.
So we can use this tools even on Rails 3+.

#7 - 2023-09-24 18:11 - Takashi Kato

Hello Dmitry!

I have written patches for the Asset Pipeline with theme and plugin support, assuming you upgrade to Rails7. #39111
I'd be grateful for feedback.

#8 - 2025-04-21 07:56 - Marius BĂLTEANU

- Related to Feature #42510: Add Stimulus as a Javascript framework added

#9 - 2025-04-21 08:00 - Marius BĂLTEANU

- Tracker changed from Patch to Feature

- Status changed from New to Closed

- Resolution set to Wont fix

I'm closing this with "Won't fix" because we have just added importmap and Stimulus to the core in addition to Propshaft and we will stick with this simpler approach.

Anyway, thanks everyone for posting these improvements!

Files

01_Remove_jquery.patch	397 KB	2021-07-06	Takashi Kato
02_Move_Assets.patch	425 KB	2021-07-06	Takashi Kato
03_Add_Simpacker.patch	251 KB	2021-07-06	Takashi Kato