

WebJars : la main dans le pot

Webjar (/wɛbjär/) *n.*: Pot de toile

WebJar = Web Java ARchive

WebJar : JAR Java qui contient des assets statiques, généralement issus d'une librairie Web classiquement disponible via npm.

Permet au développeur Java d'utiliser des librairies Web sans avoir à passer par l'écosystème Node habituellement utilisé pour le front.

Régulièrement utilisé sans même qu'on en ait conscience (e.g. SwaggerUI).

Anatomie

Exemple pour la librairie Leaflet (`org.webjars.npm:leaflet`)

```
META-INF/  
  resources/  
    webjars/  
      leaflet/ # Nom de la lib  
        1.9.4/ # Version  
          ... # Contenu de la lib, différent pour chacune  
maven/  
  # Dossier classique métadonnées Maven
```

Explication plus détaillée.

Ambition

The mission of the WebJars project is to make it easy to use open source web libraries (JavaScript & CSS) in Java-platform projects. We do this by publishing web libraries to Maven Central, replicating the dependency information in the POM, and allowing anyone to publish NPM libraries without our help. (source <https://github.com/webjars/webjars/blob/main/MISSION.md>)

Projet indépendant, non adossé à une JCP/JSR.

Initié par James Ward ([bio](#)) en 2012 (source [WebJars Officially Launched!](#)), originellement pour les écosystèmes Play & Dropwizard.

Écosystème

<https://www.webjars.org> : Un site qui permet de trouver un Webjar et de le générer si celui-ci n'existe pas (disponible ensuite dans les heures qui suivent dans Maven Central).

Intégration disponible pour 16 frameworks Java (vous trouverez certainement le vôtre).

Conventions

- GroupId en `org.webjars` : construit à façon via un projet Maven, notamment pour pouvoir gérer la déclaration de dépendances transitives (e.g. `org.webjars:Bootstrap` qui dépend de `org.webjars:popper.js`) et d'autres subtilités avancées,
- GroupId en `org.webjars.npm` : provient directement d'un packaging automatique via <http://www.webjars.org> d'une lib issue de <https://www.npmjs.com>.

Convention de nommage plus détaillée sur la page d'accueil de <http://www.webjars.org>.

Comment ça marche ?

- S'appuie très simplement sur le mécanisme de chargement de ressources à partir du classpath,
- Intégré à certains frameworks pour permettre le chargement via des URL raccourcies (exemple Spring Boot :
`/webjars/**` est mappé sur `classpath:/META-INF/resources/webjars`),
- Possibilité d'omettre la version précise de la lib dans les URL des ressources avec `webjars-locator-core` (e.g. `href='/webjars/bootstrap/css/bootstrap.min.css'` au lieu de `href='/webjars/bootstrap/3.1.0/css/bootstrap.min.css'`),
- **Attention** : Les dépendances transitives sont récupérées automatiquement dans le projet, mais doivent être spécifiées explicitement dans la page Web (e.g. nécessité de charger les ressources Bootstrap & Popper pour que Bootstrap fonctionne).

La petite démo

Concepts avancés

- Chargement possible des ressources via RequireJS avec `RequireJS.getSetupJavaScript`, helper fourni par `webjars-locator`. Peu documenté, et les essais ne sont pas toujours couronnés de succès...
- Chargement depuis un CDN jsDelivr en préfixant l'URL avec
`//cdn.jsdelivr.net/webjars/{groupId}` (e.g. `/webjars/jquery/2.1.0/jquery.js` →
`//cdn.jsdelivr.net/webjars/org.webjars/jquery/2.1.0/jquery.js`).

Bénéfices

- Un principe vraiment très simple à appréhender pour un usage de base,
- Un écosystème lui aussi très bien pensé avec la publication automatisée de livrables npm dans Maven central,
- La possibilité de faire du front en Java sans avoir à utiliser l'écosystème npm.

Limites

- Une documentation correcte mais qui se contente de l'essentiel,
- Un concept très utilisé sans forcément le savoir, donc peu de Q&A sur le sujet,
- Des mécaniques plus avancées qui peuvent ou non fonctionner (RequireJS),
- L'impossibilité d'exploiter la richesse du monde front pour la gestion des assets statiques (bundling notamment).

Pour aller plus loin

- Introducing WebJars – Web Libraries as Managed Dependencies : l'article fondateur sur le blog de James Ward,
- Le site <https://www.webjars.org> qui offre le tooling et la documentation essentielle,
- Spring Documentation - 27. Developing Web applications - 27.1.5 Static Content : la section de la doc Spring qui traite du sujet,
- Client Side Development with Spring Boot Applications : post sur le blog de Spring très intéressant sur les différentes options pour servir du front depuis Spring,
- Le code de [webjar-locator-core](#) sous GitHub.

Merci !