multer DT

1.4.5-lts.1 • Public• Publié il y a un an

Lisez-moi

Code Bêta

7 Dépendances

4 005 personnes à charge

46 Versions disponibles

Multer

Multer est un middleware node.js pour la gestion , qui est principalement utilisé pour le téléchargement de fichiers. Il est écrit sur le dessus de **Busboy** pour une efficacité maximale. `multipart/form-data`

REMARQUE : Multer ne traitera aucun formulaire qui n'est pas en plusieurs parties (). `multipart/form-data`

Traductions

Ce fichier README est également disponible dans d'autres langues :

- Español (Espagnol)
- 简体中文 (chinois)
- 한국어 (coréen)
- Русский язык (russe)

- Việt Nam (Vietnam)
- Português (Brésil portugais)

Installation

```
$ npm install --save multer
```

Usage

Multer ajoute un objet et un objet ou à l'objet. L'objet contient les valeurs des champs de texte du formulaire, l'objet ou contient les fichiers téléchargés via le formulaire. `body file files request body file files`

Exemple d'utilisation de base :

N'oubliez pas le dans votre formulaire. `enctype="multipart/form-data"`

```
<form action="/profile" method="post" enctype="multipart/form-
  <input type="file" name="avatar" />
</form>
```

```
})

const cpUpload = upload.fields([{ name: 'avatar', maxCount: 1
app.post('/cool-profile', cpUpload, function (req, res, next)
  // req.files is an object (String -> Array) where fieldname
  //
  // e.g.
  //  req.files['avatar'][0] -> File
  //  req.files['gallery'] -> Array
  //
  // req.body will contain the text fields, if there were any
})
```

In case you need to handle a text-only multipart form, you should use the method: `.none()`

```
const express = require('express')
const app = express()
const multer  = require('multer')
const upload = multer()

app.post('/profile', upload.none(), function (req, res, next)
  // req.body contains the text fields
})
```

Voici un exemple sur la façon dont multer est utilisé dans un formulaire HTML. Portez une attention particulière aux champs et : `enctype="multipart/form-data" name="uploaded_file"`

```
  </div>
</form>
```

Ensuite, dans votre fichier javascript, vous ajouterez ces lignes pour accéder à la fois au fichier et au corps. Il est important que vous utilisiez la valeur de champ du formulaire dans votre fonction de téléchargement. Cela indique à multer dans quel champ de la requête il doit rechercher les fichiers. Si ces champs ne sont pas les mêmes dans le formulaire HTML et sur votre serveur, votre téléchargement échouera : `name`

```
const multer  = require('multer')
const upload = multer({ dest: './public/data/uploads/' })
app.post('/stats', upload.single('uploaded_file'), function (r
   // req.file is the name of your file in the form above, her
   // req.body will hold the text fields, if there were any
   console.log(req.file, req.body)
});
```

API

Informations sur le fichier

Chaque fichier contient les informations suivantes :

Clé	Description	Note
`fieldname`	Nom du champ spécifié dans le formulaire	
`originalname`	Nom du fichier sur l'ordinateur de l'utilisateur	
`encoding`	Type d'encodage du fichier	
`mimetype`	Type MIME du fichier	
`size`	Taille du fichier en octets	

Clé	Description	Note
`destination`	Le dossier dans lequel le fichier a été enregistré	`DiskStorage`
`filename`	Le nom du fichier dans le répertoire `destination`	`DiskStorage`
`path`	Le chemin d'accès complet au fichier téléchargé	`DiskStorage`
`buffer`	A de l'ensemble du fichier `Buffer`	`MemoryStorage`

`multer(opts)`

Multer accepte un objet options, dont le plus basique est la propriété, qui indique à Multer où télécharger les fichiers. Dans le cas où vous omettez l'attribut options, les fichiers seront conservés en mémoire et ne seront jamais écrits sur le disque. `dest`

Par défaut, Multer renommera les fichiers afin d'éviter les conflits de noms. Le La fonction de renommage peut être personnalisée en fonction de vos besoins.

Voici les options qui peuvent être transmises à Multer.

Clé	Description
`dest` ou `storage`	Où stocker les fichiers
`fileFilter`	Fonction permettant de contrôler quels fichiers sont acceptés
`limits`	Limites des données téléchargées
`preservePath`	Conservez le chemin d'accès complet des fichiers au lieu du nom de base

Dans une application web moyenne, seule peut être requise et configurée comme indiqué dans l'exemple suivant. `dest`

```
const upload = multer({ dest: 'uploads/' })
```

Si vous souhaitez avoir plus de contrôle sur vos téléchargements, vous pouvez utiliser l'option au lieu de . Multer est livré avec des moteurs de stockage et ; D'autres moteurs sont disponibles auprès de tiers. `storage` `dest` `DiskStorage` `MemoryStorage`

`.single(fieldname)`

Acceptez un seul fichier portant le nom . Le fichier unique sera stocké dans. `fieldname` `req.file`

`.array(fieldname[, maxCount])`

Acceptez un tableau de fichiers, tous portant le nom . Si vous le souhaitez, erreur en cas de plus que les fichiers sont téléchargés. Le tableau de fichiers sera stocké dans . `fieldname` `maxCount` `req.files`

`.fields(fields)`

Accepter un mélange de fichiers, spécifié par . Un objet avec des tableaux de fichiers seront stockés dans . `fields` `req.files`

`fields` doit être un tableau d'objets avec et éventuellement un .
Exemple: `name` `maxCount`

```
[
  { name: 'avatar', maxCount: 1 },
  { name: 'gallery', maxCount: 8 }
]
```

`.none()`

N'acceptez que les champs de texte. Si un téléchargement de fichier est effectué, erreur avec le code « LIMIT_UNEXPECTED_FILE » sera émise.

`.any()`

Accepte tous les fichiers qui arrivent sur le fil. Un tableau de fichiers sera stocké dans . `req.files`

AVERTISSEMENT: Assurez-vous de toujours gérer les fichiers qu'un utilisateur télécharge. N'ajoutez jamais multer en tant qu'intergiciel global, car un utilisateur malveillant pourrait télécharger vers un itinéraire que vous n'aviez pas prévu. N'utilisez cette fonction que sur les itinéraires où vous manipulez les fichiers téléchargés.

`storage`

`DiskStorage`

Le moteur de stockage sur disque vous donne un contrôle total sur le stockage des fichiers sur le disque.

```
const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, '/tmp/my-uploads')
  },
  filename: function (req, file, cb) {
    const uniqueSuffix = Date.now() + '-' + Math.round(Math.ra
    cb(null, file.fieldname + '-' + uniqueSuffix)
  }
})

const upload = multer({ storage: storage })
```

Deux options sont disponibles et . Ils sont tous les deux qui déterminent l'emplacement de stockage du fichier. `destination` `filename`

`destination` est utilisé pour déterminer dans quel dossier les fichiers téléchargés doivent être stockés. Cela peut également être donné sous la forme d'un (par exemple). Si no est donné, le répertoire par défaut du système d'exploitation pour les files est utilisé. `string` `'/tmp/uploads'` `destination`

Note: Vous êtes responsable de la création du répertoire lors de la fourniture en tant que fonction. Lors du passage d'une chaîne, multer s'assurera que L'annuaire est créé pour vous. `destination`

`filename` est utilisé pour déterminer le nom du fichier à l'intérieur du dossier. Si no est donné, chaque fichier recevra un nom aléatoire qui n'est pas inclure n'importe quelle

extension de fichier. `filename`

Note: Multer n'ajoutera aucune extension de fichier pour vous, votre fonction doit renvoyer un nom de fichier complet avec une extension de fichier.

Chaque fonction reçoit à la fois la requête () et des informations sur le fichier () pour aider à la décision. `req file`

Notez qu'il n'a peut-être pas encore été entièrement rempli. Cela dépend de l' afin que le client transmette les champs et les fichiers au serveur. `req.body`

Pour comprendre la convention d'appel utilisée dans le callback (besoin de passer null comme premier param), reportez-vous à **la section Gestion des erreurs Node.js**

`MemoryStorage`

Le moteur de stockage en mémoire stocke les fichiers en mémoire sous forme d'objets. Il n'a pas d'options. `Buffer`

```
const storage = multer.memoryStorage()
const upload = multer({ storage: storage })
```

Lors de l'utilisation du stockage en mémoire, les informations sur le fichier contiennent un champ appelé qui contient le fichier entier. `buffer`

AVERTISSEMENT : Téléchargement de fichiers très volumineux ou de fichiers relativement petits nombres très rapidement, peut entraîner un manque de mémoire de votre application lorsque Le stockage en mémoire est utilisé.

`limits`

Objet spécifiant les limites de taille des propriétés facultatives suivantes. Multer passe cet objet directement dans busboy, et les détails des propriétés peuvent être trouvés sur la **page de busboy**.

Les valeurs entières suivantes sont disponibles :

Clé	Description	Faire défaut
`fieldNameSize`	Taille maximale du nom de champ	100 octets
`fieldSize`	Taille maximale de la valeur du champ (en octets)	1 Mo
`fields`	Nombre maximal de champs non liés à un fichier	Infini
`fileSize`	Pour les formulaires en plusieurs parties, la taille maximale du fichier (en octets)	Infini
`files`	Pour les formulaires en plusieurs parties, le nombre maximal de champs de fichier	Infini
`parts`	Pour les formulaires en plusieurs parties, le nombre maximal de parties (champs + fichiers)	Infini
`headerPairs`	Pour les formulaires en plusieurs parties, le nombre maximal de paires clé=>valeur d'en-tête à analyser	2000

La spécification des limites peut aider à protéger votre site contre les attaques par déni de service (DoS).

`fileFilter`

Définissez cette fonction pour contrôler quels fichiers doivent être téléchargés et lesquels doit être ignorée. La fonction doit ressembler à ceci :

```
function fileFilter (req, file, cb) {

  // The function should call `cb` with a boolean
  // to indicate if the file should be accepted

  // To reject this file pass `false`, like so:
```

```
  cb(null, false)

  // To accept the file pass `true`, like so:
  cb(null, true)

  // You can always pass an error if something goes wrong:
  cb(new Error('I don\'t have a clue!'))

}
```

Gestion des erreurs

En cas d'erreur, Multer délègue l'erreur à Express. Vous pouvez Affichez une belle page d'erreur en utilisant **la voie express standard**.

Si vous souhaitez intercepter les erreurs spécifiquement à partir de Multer, vous pouvez appeler la méthode l'intergiciel fonctionne par vous-même. De plus, si vous souhaitez intercepter uniquement **les erreurs Multer**, vous pouvez utiliser la classe attachée à l'objet lui-même (par exemple,). `MulterError multer err instanceof multer.MulterError`

```
const multer = require('multer')
const upload = multer().single('avatar')

app.post('/profile', function (req, res) {
  upload(req, res, function (err) {
    if (err instanceof multer.MulterError) {
      // A Multer error occurred when uploading.
    } else if (err) {
      // An unknown error occurred when uploading.
    }

    // Everything went fine.
  })
})
```

Moteur de stockage personnalisé

Pour plus d'informations sur la création de votre propre moteur de stockage, consultez Moteur de stockage Multer.

Licence

MIT

Mots-clés

forme Publier Multipart données-formulaire formdata exprimer Intergiciel

Installer

```
npm i multer
```

Dépôt

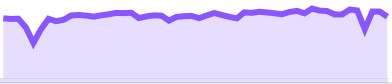
github.com/expressjs/multer

Page d'accueil

github.com/expressjs/multer#readme

Téléchargements hebdomadaires

4 015 081



Version	Licence
1.4.5-lts.1	MIT
Taille déballée	Nombre total de fichiers
27.6 kB	11
Questions	Demandes de tirage

177 59

Dernière publication

il y a un an

Collaborateurs

>_Essayer RunKit

Signaler un logiciel malveillant

Soutien

Aide

Avis

Statut

Contacter npm

Compagnie

Environ

Blog

Presser

Conditions d’utilisation et politiques

Manifeste

Conditions d’utilisation

Code de conduite

Vie privée