EBook Gratis

APRENDIZAJE ember-cli

Free unaffiliated eBook created from
Stack Overflow contributors.

#ember-cli

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: ember-cli

It is an unofficial and free ember-cli ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official ember-cli.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con ember-cli

Observaciones

Esta sección proporciona una descripción general de qué es ember-cli y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema grande dentro de ember-cli, y vincular a los temas relacionados. Dado que la Documentación para ember-cli es nueva, es posible que deba crear versiones iniciales de esos temas relacionados.

La sintaxis simple para crear un proyecto es:

```
ember new my-new-app
cd my-new-app
ember s
```

Por favor revise las instrucciones para configurar ember-cli en este documento

Examples

Instalación

Ember-cli primero requiere que Node y NPM estén instalados en el sistema. Siga las instrucciones de instalación en nodejs.org o use un administrador de paquetes preferido (como Homebrew en OSX). Se recomienda instalar la última versión de cada uno.

Una vez hecho esto, ejecute los siguientes comandos para asegurarse de que la instalación fue correcta:

```
node -v
npm -v
```

Dado que el administrador de paquetes Yarn se ha lanzado recientemente (octubre de 2016), es posible instalar dependencias con Yarn en lugar de NPM. Revisando la guía en el sitio web de hilo para más detalles.

A continuación, instale Ember CLI globalmente:

```
npm install -g ember-cli
```

O

```
yarn global add ember-cli
```

Esto otorgará acceso al corredor de línea de comando de brasas.

CENADOR

Instale globalmente Bower, un administrador de paquetes que mantiene las dependencias de front-end al día. (incluyendo jQuery, Ember, y QUnit)

```
npm install -g bower
```

O

```
yarn global add bower
```

Esto otorgará acceso al corredor de línea de comandos de Bower.

PhantomJS

Con Ember CLI, use un corredor de pruebas automatizado preferido. La mayoría de los servicios de prueba recomiendan o requieren PhantomJS, que se puede instalar a través de npm o el sitio web de PhantomJS. (PhantomJS es el corredor de prueba predeterminado para Testem y Karma).

Para usar PhantomJS para las pruebas de integración, se debe instalar globalmente:

```
npm install -g phantomjs-prebuilt
```

o

```
yarn global add phantomjs-prebuilt
```

Sereno

En sistemas operativos similares a OSX y UNIX, se recomienda instalar la versión 4.x de Watchman. Esto le proporciona a Ember CLI una manera más efectiva de ver los cambios de proyectos.

La observación de archivos en OSX es propensa a errores y el `NodeWatcher` incorporado de `NodeWatcher` tiene problemas para observar árboles grandes. Watchman resuelve estos problemas y se desempeña bien en árboles de archivos extremadamente masivos.

En OSX, instale Watchman usando Homebrew:

```
brew install watchman
```

Para obtener instrucciones de instalación completas, consulte los documentos en el sitio web de Watchman .

No utilice una versión de la NGP vigilante. El siguiente comando se puede usar para desinstalarlo:

```
npm uninstall -g watchman
```

¡Felicidades! Ahora puedes crear tu primer proyecto ejecutando:

```
ember new my-first-app
```

iniciar el servidor Ember ejecutando:

```
ember s
```

Vaya a `http://localhost:4200` para ver la nueva aplicación en acción.

Vaya a `http://localhost:4200/tests` para ver los resultados de la prueba en acción.

Lea Empezando con ember-cli en línea: https://riptutorial.com/es/ember-cli/topic/7441/empezando-con-ember-cli

Capítulo 2: Estructura de Pods Ember-Cli

Sintaxis

- Ember g [blueprints.eg: route] [nombre] --pod
- Ember g route foo --pod
- Ember g componente my-name --pod

Parámetros

Generar	vainas
sol	--vaina

Observaciones

Simplemente pase `--pod` a `ember generate` al generar nuevos archivos.

Si desea usar la estructura de pods como la predeterminada para su proyecto, puede configurar usePods en su archivo de configuración `.ember-cli` en verdadero (la configuración se denominó anteriormente `usePodsByDefault`). Para generar o destruir un plano en la estructura de tipos clásica mientras `usePods` es `true` , use el indicador `--classic` .

Con los usePods establecidos en true.

```
// .ember-cli
{
    "usePods": true
}
```

Lo siguiente ocurriría al generar una ruta:

```
ember generate route taco

installing
  create app/taco/route.js
  create app/taco/template.hbs
installing
  create tests/unit/taco/route-test.js

ember generate route taco --classic

installing
  create app/routes/taco.js
  create app/templates/taco.hbs
installing
  create tests/unit/routes/taco-test.js
```

Hay algunos beneficios de usar este método, sin embargo, depende completamente de usted. En primer lugar, separa su aplicación en agrupaciones más lógicas, por lo tanto, puede mantener sus archivos perfectamente organizados en recursos.

Esta estructura también facilita la vida de nuestro desarrollo. Por ejemplo, si quiero encontrar el `controller myname` en la estructura predeterminada, tengo que hacer un prefacio de lo que realmente quiero (myname) con el tipo (controladores). Sin embargo, con las cápsulas, puedo encontrar el mismo controlador de forma aproximada simplemente buscando "mi nombre".

Examples

Organizar con vainas

```
app/controllers/myname.js
app/templates/myname.hbs
app/routes/myname.js
app/models/myname.js
```

Usando vainas, el ejemplo anterior se traduciría en esto:

```
app/myname/controller.js
app/myname/template.hbs
app/myname/route.js
app/myname/model.js
```

podModulePrefix: aplicación / pods

```
ember generate route foo --pod

installing
  create app/pods/foo/route.js
  create app/pods/foo/template.hbs
installing
  create tests/unit/pods/foo/route-test.js
```

Lea Estructura de Pods Ember-Cli en línea: https://riptutorial.com/es/ember-cli/topic/7855/estructura-de-pods-ember-cli

Capítulo 3: Primeros pasos con Ember-Cli y las implementaciones

Sintaxis

- Miembro desplegar producción // desplegar entorno de producción
- Miembro desplegar almacenamiento provisional / implementar entorno de almacenamiento intermedio
- Miembro desplegar desarrollo // desplegar entorno de desarrollo que no está comprimido ni minimizado

Parámetros

parámetros	detalles
`ember help`	Mostrar todos los parámetros posibles y la guía de profundidad, así como códigos cortos

Observaciones

Ember-Cli es una herramienta poderosa que viene con muchas otras para ayudarnos a implementar más rápido y conveniente. Todo lo que necesitas para instalar Ember-Cli-Deploy y usar `ember deploy` .

> Ember CLI Deploy estructura la implementación de su aplicación utilizando una canalización de implementación, que consta de varios enlaces de canalización. Estos enlaces estándar son la base de un rico ecosistema de complementos que puede componer para crear un proceso de implementación adecuado para su aplicación.

Como Ember-Cli-Deploy es un complemento de Ember para que pueda instalarlo fácilmente con `ember install ember-cli-deploy` . Hay otros dos complementos útiles que hacen que nuestra compilación y compresión sean confiables durante la implementación.

Simplemente ejecuta los siguientes comandos:

```
# Install the Build plugin, which builds your app during deployment
ember install ember-cli-deploy-build

# Gzip our files
ember install ember-cli-deploy-gzip
```

Sin embargo, si va a maximizar sus beneficios utilizando el `ember deploy` , es muy probable que tenga diferentes entornos en su aplicación Ember e implemente la versión de producción, puesta en escena o desarrollo de su aplicación con la configuración adecuada.

Las plataformas que puedes desplegar ahora son:

- Heroku
- Azur
- AWS S3
- Base de fuego
- CouchDB cluster

Consulte la sección de ejemplo para ver cómo puede implementar.

Examples

Heroku

Debes tener instalado Heroku Toolbelt primero. Tener una cuenta en Heroku y la instalación de ember-cli-deploy son obligatorios.

Creación de una nueva instancia de Heroku desde el directorio principal de una aplicación Ember CLI:

```
$ heroku create --buildpack https://github.com/tonycoco/heroku-buildpack-ember-cli.git

$ git push heroku master
```

Azur

En primer lugar, se requiere instalar el módulo ember-cli-azure-deploy de Microsoft. Necesitas estar en el directorio raíz de tu aplicación.

```
npm install --save-dev -g ember-cli-azure-deploy
azure-deploy init
```

Si está utilizando el gestor de paquetes Yarn, simplemente puede instalarlo mediante:

```
yarn global add ember-cli-azure-deploy
azure-deploy init
```

Esto creará un `deploy.sh` en la carpeta raíz de su proyecto, lo que permitirá a Azure seguir un conjunto de instrucciones, incluida la instalación de todos los módulos de nodo necesarios, la ejecución de la `ember build` y la implementación de la carpeta `dist/` resultante en `wwwroot` su sitio web.

Base de fuego

Primero, necesitas instalar las herramientas de Firebase . Simplemente, ejecute los comandos a continuación:

Gestor de paquetes Npm

```
npm install -g firebase-tools
```

o gestor de paquetes Yarn

```
yarn add firebase-tools
```

Para configurar su aplicación para que esté lista para implementarse, debe ejecutar lo siguiente en el directorio raíz de su aplicación:

```
firebase init
```

Finalmente, ejecutando el siguiente comando puedes implementar tu aplicación.

```
firebase deploy
```

AWS S3

Para continuar con el despliegue a S3, instalaremos estos complementos:

- ember-cli-deploy-s3-index : carga el index.html a S3 con información de revisión y lo activa
- ember-cli-deploy-s3 : carga los activos (js, css y otros archivos de medios) a S3

Como son complementos de Ember, puedes instalarlos fácilmente ejecutando los siguientes comandos

```
ember install ember-cli-deploy-s3-index
ember install ember-cli-deploy-s3
```

Todo lo que necesita después de eso es configurar el archivo deploy.js que debería en la carpeta / config:

```
// config/deploy.js

module.exports = function(deployTarget) {

  var ENV = {
    build: {
      environment: deployTarget
    },
    'revision-data': {
      type: 'git-commit'
    },
    's3-index': {
      accessKeyId: process.env['S3_ACCESS_KEY'],
      secretAccessKey: process.env['S3_SECRET_ACCESS_KEY'],
      bucket: "your-app-deployment-bucket",
      region: "YOUR REGISION",
      allowOverwrite: true // if you want to overwrite index file if not change it to false
    },
    's3': {
      accessKeyId: process.env['S3_ACCESS_KEY'],
      secretAccessKey: process.env['S3_SECRET_ACCESS_KEY'],
```

```
      bucket: "your-app-deployment-bucket",
      region: "YOUR REGISION",
    }
  };

  return ENV;

};
```

Tenga en cuenta que hemos utilizado las variables de entorno en nuestra configuración. Si necesita leer la configuración de un archivo, también es posible devolver una promesa que se resuelva con el objeto `ENV` . Aquí hay un ejemplo para definir diferentes entornos en el archivo deploy.js:

```
if (deployTarget === 'development') {
    ENV.build.environment = 'development';
    // configure other plugins for development deploy target here
  }

  if (deployTarget === 'staging') {
    ENV.build.environment = 'production';
    // configure other plugins for staging deploy target here
  }

  if (deployTarget === 'production') {
    ENV.build.environment = 'production';
    // configure other plugins for production deploy target here
  }
```

Finalmente, puede ejecutar fácilmente el siguiente comando para desplegar

```
ember deploy [YOUR APP ENVIRONMENT] //e.g-> ember deploy production or ember deploy staging
```

Si te gusta ver detalles puedes correr:

```
ember deploy production --verbose --activate=true
```

Lea Primeros pasos con Ember-Cli y las implementaciones en línea: https://riptutorial.com/es/ember-cli/topic/7444/primeros-pasos-con-ember-cli-y-las-implementaciones

Creditos

S. No	Capítulos	Contributors
1	Empezando con ember-cli	4444, Community, Majid
2	Estructura de Pods Ember-Cli	Majid
3	Primeros pasos con Ember-Cli y las implementaciones	Majid