# Template Engines

A **template engine** enables you to use static template files in your application. At runtime, the template engine replaces variables in a template file with actual values, and transforms the template into an HTML file sent to the client. This approach makes it easier to design an HTML page.

Some popular template engines that work with Express are [Pug](https://pugjs.org/api/getting-started.html), [Mustache](https://www.npmjs.com/package/mustache), and [EJS](https://www.npmjs.com/package/ejs). The [Express application generator](https://expressjs.com/en/starter/generator.html) uses [Jade](https://www.npmjs.com/package/jade)as its default, but it also supports several others.

**Note**: Jade has been renamed to [Pug](https://www.npmjs.com/package/pug). You can continue to use Jade in your app, and it will work just fine. However if you want the latest updates to the template engine, you must replace Jade with Pug in your app.

**To render template files, set the following**[**application setting properties**](https://expressjs.com/en/4x/api.html#app.set)**, set in app.js in the default app created by the generator:**

* **views, the directory where the template files are located. Eg: app.set('views', './views'). This defaults to the viewsdirectory in the application root directory.**
* **view engine, the template engine to use. For example, to use the Pug template engine: app.set('view engine', 'pug').**

Then install the corresponding template engine npm package; for example to install Pug:

$ npm install pug --save

After the view engine is set, you don’t have to specify the engine or load the template engine module in your app; Express loads the module internally, as shown below (for the above example).

app.set('view engine', 'pug')

Create a Pug template file named index.pug in the views directory, with the following content:

html

head

title= title

body

h1= message

Then create a route to render the index.pug file. If the view engine property is not set, you must specify the extension of the view file. Otherwise, you can omit it.

app.get('/', function (req, res) {

res.render('index', { title: 'Hey', message: 'Hello there!' })

})

When you make a request to the home page, the index.pug file will be rendered as HTML.

Note: The view engine cache does not cache the contents of the template’s output, only the underlying template itself. The view is still re-rendered with every request even when the cache is on.

**Developing template engines for Express**

Use the app.engine(ext, callback) method to create your own template engine. ext refers to the file extension, and callback is the template engine function, which accepts the following items as parameters: the location of the file, the options object, and the callback function.

The following code is an example of implementing a very simple template engine for rendering .ntl files.

var fs = require('fs') // this engine requires the fs module

app.engine('ntl', function (filePath, options, callback) { // define the template engine

fs.readFile(filePath, function (err, content) {

if (err) return callback(err)

// this is an extremely simple template engine

var rendered = content.toString().replace('#title#', '<title>' + options.title + '</title>')

.replace('#message#', '<h1>' + options.message + '</h1>')

return callback(null, rendered)

})

})

app.set('views', './views') // specify the views directory

app.set('view engine', 'ntl') // register the template engine

Your app will now be able to render .ntl files. Create a file named index.ntl in the views directory with the following content.

#title#

#message#

Then, create the following route in your app.

app.get('/', function (req, res) {

res.render('index', { title: 'Hey', message: 'Hello there!' })

})

When you make a request to the home page, index.ntl will be rendered as HTML.