



TEMPLATES EN EXPRESS

GLOSARIO


- **Template:** es un archivo que contiene HTML y, opcionalmente, instrucciones adicionales sobre cómo generar fragmentos de HTML, tales como interpolación de texto, bucles, condicionales, inclusiones, etc.
- **View engine:** También llamado "biblioteca de plantillas" o "templater", es una biblioteca que implementa la funcionalidad de vista, y potencialmente también un lenguaje personalizado para especificarlo (como lo hace Pug).



GLOSARIO

- **Text interpolation / String interpolation:** La inserción de valores variables en una cadena de algún tipo. Los ejemplos típicos incluyen cadenas de plantillas de ES6, o este ejemplo en Pug:

Hola #{user.username}!

- **Locals:** Las variables  que se pasan a una plantilla, que se utilizarán en la representación de esa plantilla. Generalmente se especifican cada vez que desea representar una plantilla.



INSTALACIÓN Y CONFIGURACIÓN

Instalación en el proyecto

```
npm install --save pug
```



Configuración en Express

```
let app = express();  
app.set("view engine", "pug");  
/* ... rest of the application goes here ... */
```



IMPLEMENTACIÓN

homepage.pug

```
html
  body
    h1 Hello World!
    p Nothing to see here.
```



app.js

```
router.get("/", (req, res) => {
  res.render("homepage");
});
```



Express agregará automáticamente una extensión al archivo. Eso significa que, el nombre de la plantilla de "página de inicio" en el ejemplo anterior apuntará a *views / homepage.pug*.



TRABAJANDO CON LOCALS

homepage.pug

```
html
  body
    h1 Hello World!
    p Hi there, #{user.username}!
```


app.js

```
router.get("/", (req, res) => {
  res.render("homepage", {
    user: req.user
  });
});
```



CONDICIONALES

homepage.pug

```
html
  body
    h1 Hello World!
    if user != null 
      p Bienvenido, #{user.username}!
    else
      p Usuario desconocido!
```



app.js

```
router.get("/", (req, res) => {
  res.render("homepage", {
    user: req.user
  });
});
```



LOOPS

homepage.pug


```
html
  body
    h1 Hello World!
    if user != null
      p Bienvenido, #{user.username}!
    else
      p Usuario desconocido!
    p La lista de vegetables:
    ul
      for vegetable in vegetables 
        li= vegetable 
```




LOOPS

app.js

```
router.get("/", (req, res) => {  
  res.render("homepage", {  
    user: req.user,  
    vegetables: [ "carrot", "potato", "beet" ]  
  });  
});
```



REQUEST WIDE LOCALS

Si se desea que una variable esté disponible en cada *res.render* para una solicitud, sin importar de qué ruta o middleware  esté siendo procesada la página. Esto se puede lograr estableciéndolo como una propiedad en el objeto *res.locals*.

Un ejemplo típico es el objeto de usuario para el usuario actual.



REQUEST LOCAL




homepage.pug

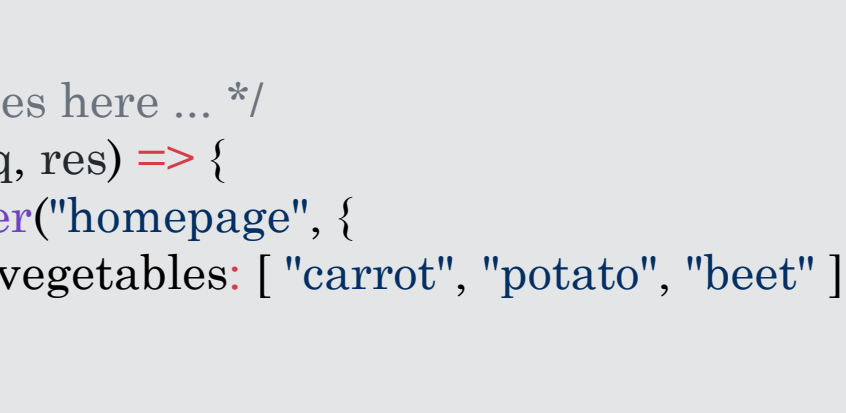
```
html
  body
    h1 Hello World!
    if user != null
      p Bienvenido, #{user.username}!
    else
      p Usuario desconocido!
    p La lista de vegetables:
    ul
      for vegetable in vegetables
        li= vegetable
```



REQUEST LOCAL

app.js

```
app.use((req, res, next) => {  
  res.locals.user = req.user;  
  next();   
});  
/* ... more code goes here ... */  
router.get("/", (req, res) => {  
  res.render("homepage", {  
    vegetables: [ "carrot", "potato", "beet" ]  
  });  
});
```



Se pasa el control a la siguiente función de middleware. Para que ejecute, o quedará colgada la solicitud.



APP WIDE-LOCALS

Algunas veces, un valor necesita ser aplicado a toda la aplicación; un ejemplo típico sería el nombre del sitio para una aplicación u otra configuración de aplicación que no cambia para cada solicitud.

Esto funciona de manera similar a *res.locals*, solo que ahora se configura en *app.locals*.



APP WIDE LOCAL



homepage.pug

```
html
  body
    h1 Hello World, this is #{siteName}!
    if user != null
      p Hi there, #{user.username}!
    else
      p Hi there, unknown person!
    p Have some vegetables:
    ul
      for vegetable in vegetables
        li= vegetable
```



APP WIDE LOCAL

app.js

```
app.locals.siteName = "Vegetable World";
/* ... more code goes here ... */
app.use((req, res, next) => {
  res.locals.user = req.user;
  next();
});
/* ... more code goes here ... */
router.get("/", (req, res) => {
  res.render("homepage", {
    vegetables: [ "carrot", "potato", "beet" ]
  });
});
```

El orden de especificidad es el siguiente: los *app.locals* se sobrescriben con *res.locals* del mismo nombre, y *res.locals* se sobrescriben con *res.render* locales del mismo nombre.

