

# Programmation Web – Avancé

JavaScript & Node.js

Partie 18 : Les sessions côté serveur et MVC via Express

Version 2020



Attribution –  
Partage dans les  
Mêmes Conditions  
4.0 International  
(CC BY-SA 4.0)

*Presentation template  
by [SlidesCarnival](#)*



# Gestion de session côté serveur

Comment se souvenir de données client côté serveur?



## **Introduction à la gestion de session côté serveur**

---

- Session
  - identification d'un client au travers de plusieurs requêtes à un même serveur
  - enregistrement d'information au sujet de cet utilisateur
- Utilisation d'un cookie pour gérer cette session



## Introduction à la gestion de session côté serveur

---

- Cas typiques de fin de session :
  - Redémarrage de l'application web
  - Effacement des cookies côté client
  - Fermeture complète du browser



## Gestion de session côté serveur avec Express

- Session middleware : **express-session** [\[68.\]](#)
- ID de session dans un cookie (pas d'autres données)
- Enregistrement de données de session côté serveur : mémoire, une BD ou un cache mémoire (Redis...)



## Gestion de session côté serveur avec Express

- Installation : **npm install express-session**
- Utilisation du middleware :

```
var session = require("express-session");  
// NB : the middleware shall be prior to the router  
app.use(  
  session({  
    secret: "689HiHoveyDi79*",  
    resave: false, // default value is true, but using the default has been  
                  deprecated...  
    saveUninitialized: false, // ditto  
  })  
);
```



## Gestion de session côté serveur avec Express

---

### ● Options

- Signer le « session ID cookie » : secret
- Détails [\[68.\]](#)



## Gestion de session côté serveur avec Express

- Données de sessions attachées aux requêtes :
  - Session id :  
**req.sessionID** (alias de **req.session.id**)
  - Lecture & modification de données :  
**req.session**

```
// initialize the session data
app.use(function (req, res, next) {
  if (!req.session.isAuthenticated) {
    req.session.isAuthenticated = false;
    req.session.user = "";
  }
  next();
});
```

```
router.post("/register", function (req, res, next){
  // ...
  req.session.isAuthenticated = true;
  req.session.user = req.body.email;
  // ...
});
router.get("/list", function (req, res, next){
  if (req.session.isAuthenticated)
    // render the list
  else
    // unauthorized access
});
```





## Gestion de session côté serveur avec Express

- Données de sessions attachées aux requêtes :
  - Effacer une session : `req.session.destroy()`

```
router.get("/logout", function (req, res, next) {  
  if (req.session.isAuthenticated) {  
    req.session.destroy(function (err) {  
      // cannot access session here  
      if (err) return console.error("Error in session destroy:", err);  
      return res.redirect("/login");  
    });  
  }  
});
```



# MVC appliqué à une Express MPA

Bien structurer son Express MPA à l'aide du pattern MVC...

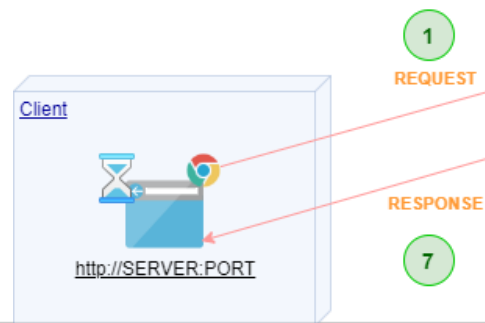


## MVC

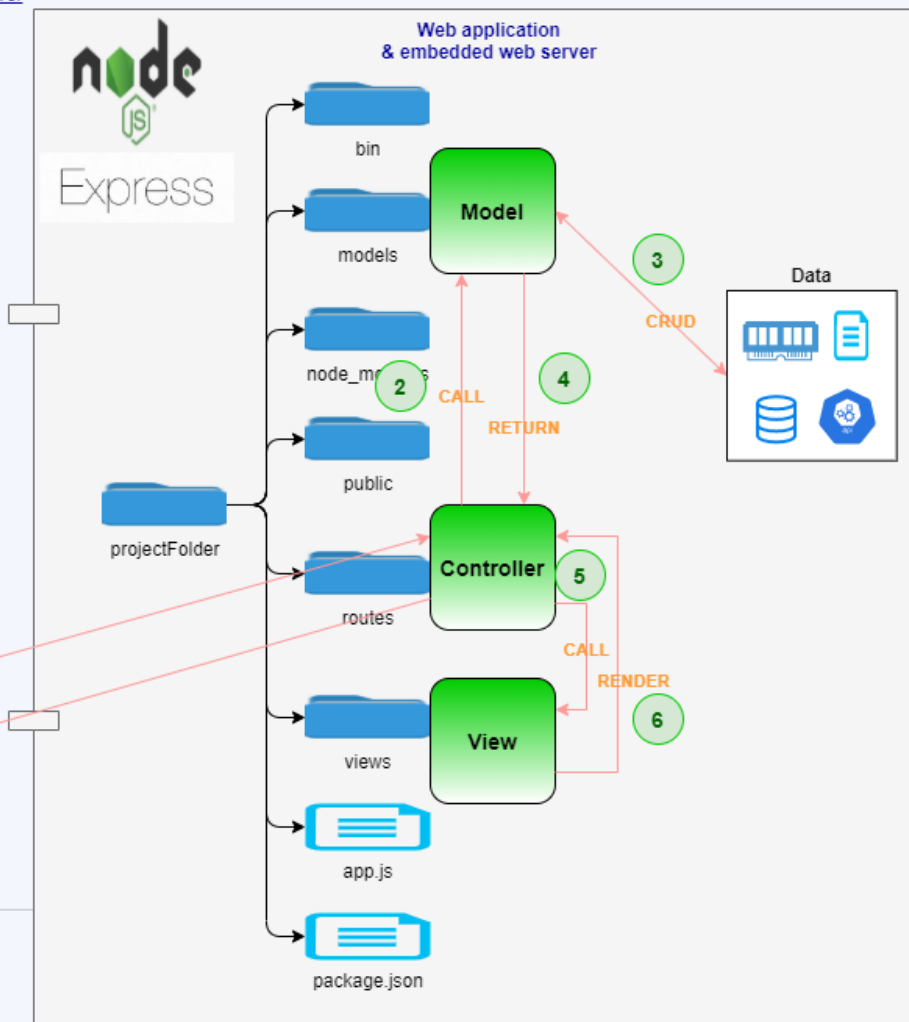
---

- Pattern Model View Controller facilement applicable à une MPA sous Express
- Model : gestion des données
- View : gestion du Server Side Rendering via des templates (hbs, pug...)
- Controller : fourniture des views en fonction des requêtes

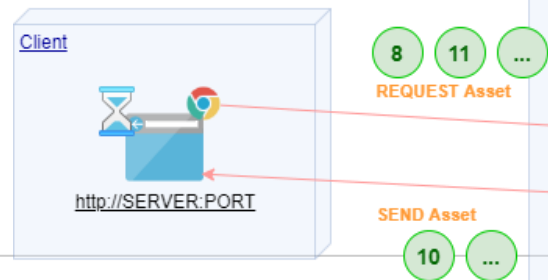
## Pattern MVC appliqué à une MPA sous Express



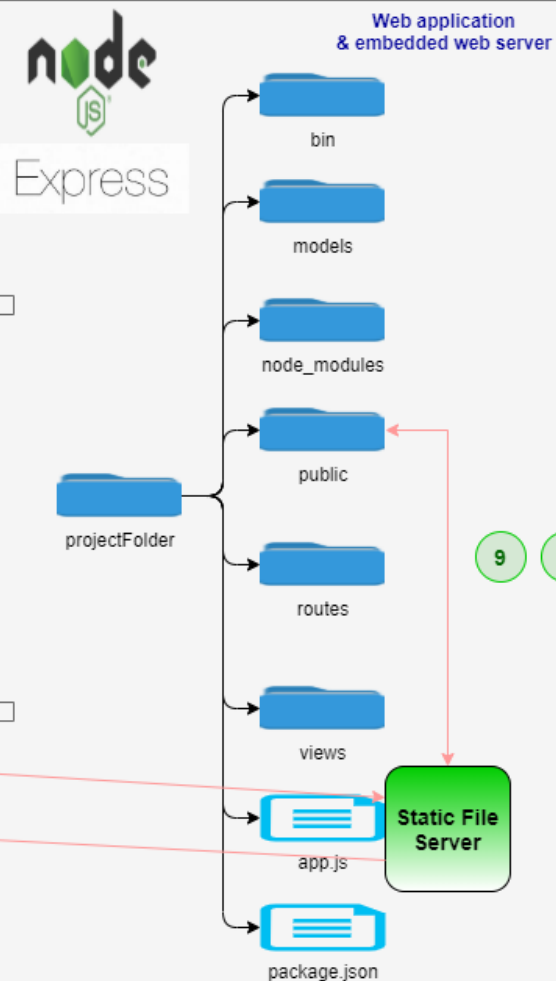
Server



## Pattern MVC appliqué à une MPA sous Express



Server

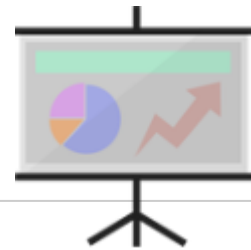




## Introduction à la gestion de session côté serveur

### 🕒 DEMO : Gestion de session via express-session & MVC (hbs)

- Gestion de l'authentification d'utilisateurs sur base de l'email et du password donné à l'enregistrement.
- Utilisateur non authentifié : login ou register (liste des utilisateurs non autorisée)
- Utilisateur authentifié : liste des utilisateurs, logout  
=> pas de login ou de register
- Données de session : user (email affiché dans le menu), isAuthenticated



## Introduction à la gestion de session côté serveur

- NB : DEMO : Gestion de session via express-session & MVC (pug)