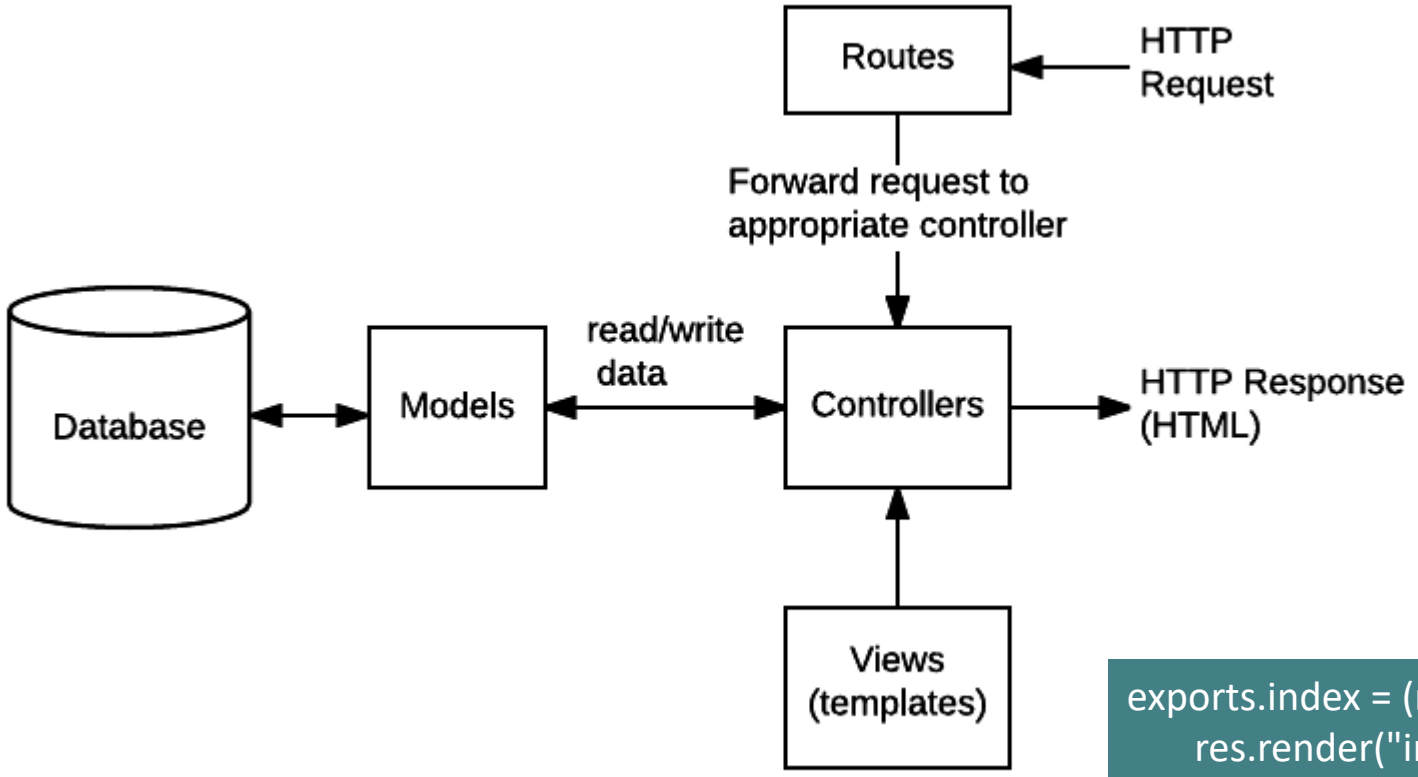
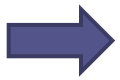
MVC (developer.mozilla.org/)




```
app.get("/", function (req, res) {
  res.render("index", { title: "About dogs",
message: "Dogs rock!" });
});
```



```
exports.index = (req, res) => {
      res.render("index", { title: "About dogs", message: "Dogs rock!" });
};
```

controller

```
const controller=require("../controller/mycontroller");
app.get("/", controller.index);
```

route

Accès aux paramètres

Access aux paramètres de la requête

- url : localhost:3000/users/userslist?page=2&limit=3

```
var express = require('express');
var router = express.Router();
var userModel = require('../model/user.js’)

/* GET users listing. */
router.get('/userslist', function (req, res, next) {

  result=userModel.readall(function(result){
    let page = req.query.page;
    let limit = req.query.limit;



    ……

  });
});
module.exports = router;
```

Accès aux paramètres de l'url

- url: localhost:3000/users/userslist/admin

```
var express = require('express');
var router = express.Router();
var userModel = require('../model/user.js')

/* GET users listing. */
router.get('/userslist/:type', function (req, res, next) {

  result=userModel.readall(function(result){
     let type = req.params.type;


     …


  });
});
module.exports = router;
```

Accès aux données body de post

- url: localhost:3000/users/userslist (post => ajouter un utilisateur via un formulaire)

```
var express = require('express');
var router = express.Router();
var userModel = require('../model/user.js’)
app.use(express.urlencoded({ extended: true }));
app.use(express.json());

router.post('/users', function (req, res, next) {

    const user_fname = req.body.fname;
    const user_lname = req.body.lname;
    …

  });
});
module.exports = router;
```

Gestion des cookies, sessions

Cookies http

- Donnée de petite taille (<=4ko)
  - Envoyée par le serveur au client (navigateur)
- Plusieurs utilités :
  - Gestion des sessions
  - Personnalisation
  - Pistage

Gestion des cookies

- npm install cookie-parser

```
var express = require('express')
var cookieParser = require('cookie-parser')
var app = express()
app.use(cookieParser())
app.get('/', function (req, res) {
  // Cookies that have not been signed
  console.log('Cookies: ', req.cookies)

  // Cookies that have been signed
  console.log('Signed Cookies: ', req.signedCookies);

  //Delete the saved cookies
  res.clearCookie();

  //Set a cookie
  res.cookie(`attribute name`,`value`);

})
```

Gestion des sessions

- http est stateless (sans état) => le serveur oublie que le client a déjà fait une requête juste avant.
- On a besoin d'un mécanisme pour sauvegarder l'état, par exemple après une authentification pour éviter de s'authentifier à chaque opération
  - le mécanisme ***session*** répond à ce besoin

Gestion des sessions : principe

1. connexion au serveur, le serveur crée une session et la stocke côté serveur.
2. Lorsque le serveur répond au client, il envoie un cookie. Ce cookie contiendra l'identifiant unique de la session stocké sur le serveur, qui sera désormais stocké sur le client.
3. Ce cookie sera envoyé à chaque requête au serveur.

Gestion des sessions : middlewares

- Npm install express-session cookie-parser
- Ou ajouter les dépendances dans package.json et lancer npm install dans le dossier racine de votre projet

Gestion des sessions : initialisation

- **Dans app.js :**

```
const cookieParser = require("cookie-parser");
const sessions = require('express-session');
```

```
const deuxHeures = 1000 * 60 * 60 * 2;
app.use(sessions({
    secret: "votre secret ici hdhhdhhshshshshsh",
    saveUninitialized:true,
    cookie: { maxAge: deuxHeures },
    resave: false
}));

// cookie parser middleware
app.use(cookieParser());
```

Option	Description
secret	une chaine unique aléatoire utilisée pour authentifier une session, elle ne doit pas être exposé au public.
saveUninitialized	permet de retourner une session non initialisée
cookie	durée de cookie de sessiosn
resave	prend une valeur booléenne. Il permet de stocker la session dans le stockage de sessions, même si la session n'a jamais été modifiée lors de la demande. Cela peut entraîner une situation de concurrence dans le cas où un client fait deux requêtes parallèles au serveur.

Gestion des sessions : création de la sessions


```
app.post('/user',(req,res) => {
    if(req.body.email == myemail && req.body.password == mypassword){
        var session=req.session;
        session.userid = req.body.email;

        console.log(req.session)
        res.send(`Hey there, welcome <a href=\'/logout'>click to logout</a>`);
    }
    else{
        res.send('Invalid email or password');
    }
})
```

Gestion des sessions : utiliser la session


```
app.get('/',(req,res) => {
    session=req.session;
    if(session.userid){
        res.send("Welcome User <a href=\'/logout'>click to logout</a>");
    }else
    res.redirect('/login.html');
});
```

Gestion des sessions : destruction de la sessions


```
app.get('/logout',(req,res) => {
    req.session.destroy();
    res.redirect('/');
});
```

Gestion des sessions : utiliser le middleware « **passeport** »

- Configurer les fonctions de sérialisation et désérialisation (user vers session et session vers user)

```
var passport = require('passport');

passport.serializeUser(function(user, cb) {
  process.nextTick(function() {
    return cb(null, {
      id: user.id,
      username: user.username,
      picture: user.picture
    });
  });
});

passport.deserializeUser(function(user, cb) {
  process.nextTick(function() {
    return cb(null, user);
  });
});
```

Gestion des sessions : utiliser le middleware « passeport »

- Authentifiez toutes les routes en utilisant passeport.authenticate() comme middleware au niveau de l'application.

```
app.use(passport.authenticate('session'));
```

- Vous pouvez également authentifier des routes spécifiques en utilisant passeport.authenticate() sur des routes montées sur un chemin.

```
app.get('/pages',
  passport.authenticate('session'),
  function(req, res, next) {
    /* ... */
  });
```

- Documentation
  - https://www.passportjs.org/docs/