* Authentication認證、Cryptography密碼學

App.js

*// 以下是最基本最基本的login system，但是這個系統有重大缺陷!!*

*// 千萬不要工作時實作!!*

*// 最終結論!!千萬不要把真正的password存在DB裡面!!!*

*// 哪天被駭了，這組帳密就被到處拿來用，直接死亡!*

*// 所以我們要把密碼加密後再放入DB!!*

*// hash function是一個很好的密碼加密方式:*

1. *不可逆(極難)*
2. *不管你的密碼長度多少，FUNC出來的結果都是一樣的長度*
3. *一點點小小改變就造成很大的RESULT不同*

*// 近代資料庫都是透過這個hash function將資料做加密存進DB*

*//我們直接用業界發展成熟的HASH FUNCTION使用就可以了!*

app.post("/login",async(req,res,next)=>{

  let{username,password}=req.body;

*try*{

  let foundUser = *await* User.findOne({username});

*//記得這邊用findOne，用find回傳整個array麻煩*

*// if(!foundUser){  // !abc 等價 abc!==null*

*//   res.send("Username not found!");*

*// } else if(password ==foundUser.password){*

*//     res.render("secret.ejs");*

*//   } else{*

*//     res.send("Password not correct!")*

*// }*

*if*(foundUser!==null && password == foundUser.password){

      res.render("secret.ejs");

    } *else* {

      res.send("Password or Username not correct!")

    }

} *catch* (err){

    next(err);

  }

})

app.get("/signup",(req,res)=>{

    res.render("signup");

});

app.post("/signup",(req,res,next)=>{ *//catch的err要透過next送出去! 記得塞next parameter*

  console.log(req.body);

  let{username,password}=req.body;

  let newUser = new User({username:username,password:password});

*try*{

  newUser.save().then((data)=>{

    res.send("Data has saved into DB!");

    console.log(data);

  }).catch((dberr)=>{

    res.send("Error! Data has not saved!");

*// console.log(dberr);*

  });

} *catch*(err){

  next(err);*//別忘記把try catch這邊的錯誤交給error handler*

}

});

* 實做加密【使用npm module bcrypt】相對上一頁安全許多(user密碼經過加密才存在電腦!!)

包含了salting以及salting之後的hash function

我們要透過 npm module : node.bcrypt.js進行。

Bcrypt本身就是一個salting+hash function之後的一個加密用演算法

<https://www.npmjs.com/package/bcrypt>

cmd> npm install bcrypt

app.js 註冊部分:

針對最重要的save()前置作業要先通過bcrypt加密!!

const bcrypt = require("bcrypt");//先做require

const saltRounds = 10; *//2^10=1024次去做hash function(設定hash function做幾次)*

app.post("/signup", (req, res, next) => {

*//catch的err要透過next送出去! 記得塞next parameter*

  console.log(req.body);

 let { username, password } = req.body;

*//排除同樣名稱被重複進入資料庫的狀況*

*try* {*//先檢查DB內有沒有一樣的NAME(只檢查NAME的原因在於，資料庫有同樣的密碼其實沒關係，但同樣名稱就不行了)*

    let foundUser = *await* User.findOne({ username });

*if* (foundUser) {

      res.send("Username has been saved.");

    } *else* {

      bcrypt.genSalt(saltRounds, (err, salt) => {*//進行加密*

*if* (err) {next(err);}

        console.log(salt); *//$2b$10$6mjxrjl0FNdlwUU7JAfR3e*

        bcrypt.hash(password, salt, (err, hash) => {

*if* (err) { next(err);}

          console.log(hash); *//$2b$10$6mjxrjl0FNdlwUU7JAfR3eYQ7v8gCHeuOPZpnL68YTqjSdCC/kJTi*

          let newUser = new User({ username: username, password: hash });

*try* {newUser.save()

              .then((data) => {

                res.send("Data has saved into DB!");

                console.log(data);})

              .catch((dberr) => {

                res.send("Error! Data has not saved!");});

          } *catch* (err) {next(err) };

        }); *//別忘記把try catch這邊的錯誤交給error handler*

      });

    }

  } *catch* (e) {next(e);}

});

app.js 登入部分:

*//用niddle確認是否已經登入!*

const requireLogin = (req,res,next)=>{

*if*(!req.session.isVarified==true){ *//如果沒登入*

    res.redirect("login"); *//重導向login頁面*

  } *else* {

    next();

  }

};

app.post("/login", async (req, res, next) => {

  let { username, password } = req.body;

*try* {

    let foundUser = *await* User.findOne({ username }); *//記得這邊用findOne，用find回傳整個array麻煩*

*if* (foundUser) {

      bcrypt.compare(password, foundUser.password, (err, result) => {

*if* (result === true) { *//找到資料庫有這個人，密碼比對也相同*

          req.session.isVarified=true;*//確認登入後就在session存入isVarified property = true;*

          res.redirect("secret");*//這邊的redirect代表導向另一個route!*

        } *else* {

          res.send("Password or Username not correct!");

        };

*if* (err) {

          next(err);

        };

      });

    } *else* {

      res.send("Password or Username not correct!");

    }

  } *catch* (err) {

    next(err);

  }

});