ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Il sito di default su Gocker + Node + Mongo

Fabio Vitali + Vincenzo Rubano

Corso di laurea in Informatica
Alma Mater – Università di Bologna

Usare node sulle macchine del DISI

- Il progetto di TW richiede l'uso di node come motore dell'applicazione server-side.
- Inoltre le regole del corso impongono l'uso di una macchina DISI per ogni servizio server-side del progetto.
  - Ogni struttura dati e ogni servizio sw creato da studenti UniBo deve essere fornito da una macchina *.cs.unibo.it
- Il DISI mette a disposizione DUE modi per attivare un servizio node:
  - attraverso *gocker*, un servizio docker di virtualizzazione di una macchina online (*container*)
  - Via linea di comando su una porta alta (>1024)
- Entrambi richiedono qualche competenza di linea di comando (poca roba)
- Entrambi funzionano sullo spazio dati e con i permessi dello studente, e non dell'amministratore o di root.

Node da linea di comando

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

http.createServer(
   function (request, response) { ... }
).listen(8000);
```

- Il modo più semplice per attivare node è da linea di comando:
  - `fabio@eva:~$ cd /home/web/siteXXXX/html/`
  - `fabio@eva:/home/web/siteXXXX/html/$ node index.js`
- A questo punto node risponde all'indirizzo:
  - `http://eva.cs.unibo.it:8000/`

PATH e shell

- Può succedere che otteniate errori tipo

```
fabio.vitali@eva:~$ node index.js
-bash: node: command not found
fabio.vitali@eva:~$ npm install express
-bash: npm: command not found
```

- Questo significa che non avete la directory di node e npm nel PATH (che sono le directory cercate automaticamente dalla shell).
  - E' necessario o specificare il path assoluto di node e npm...
    - `/usr/local/node/bin/node index.js`
    - `/usr/local/node/bin/npm install express`
  - oppure inserire la directory di node nel PATH:
    - `cd $home`
    - `nano .bash_profile`
    - Aggiungete la seguente riga: `export PATH=/usr/local/node/bin/:$PATH`
    - Riavviate la shell, lanciando il comando: `exit`

Limiti e problemi di node da linea di comando

- Funziona solo finché la shell è aperta.
  - Quindi non possiamo scollegarci finché vogliamo che funzioni
  - Non è vero, c'è un modo, ma non è il momento per parlarne.
- Funziona solo dalla macchina da cui abbiamo lanciato l'eseguibile.
  - Quindi l'URI deve specificare esattamente la macchina scelta.
- Funziona solo su porte alte > 1024, e la porta deve essere libera per quella macchina.
  - quindi l'URL da usare DEVE specificare la porta
  - Suggerimento: poiché Gocker funziona necessariamente sulla porta 8000, usate la porta 8000.
- Ci può essere solo UN servizio attivo su una data porta e un dato server
  - Quindi se usate tutti la porta 8000 dovete tutti usare server diversi.
- Utile nelle fasi di programmazione e debug.
- Una volta che il server funzioni dovrete cercare qualcosa di più stabile.

Docker (1/3)

- E' un fornitore di container software, ovvero macchine virtuali minimali in grado di eseguire una sola applicazione senza richiedere tutta l'infrastruttura di un sistema operativo.
- Il DISI utilizza docker per fornire macchine virtuali con un software preinstallato:
  - node
  - python
  - php
  - mongoDB
- **Sebbene in teoria sia possibile, questo laboratorio non accetta immagini con sw preinstallato diverso da questi.**
- I container docker accedono al file system condiviso dei server del dipartimento, quindi possono essere attivati su qualunque directory del file system. Noi tradizionalmente usiamo /home/web/site2122XX/html/

Docker (2/3)

- La nostra implementazione di docker risponde via ssh all'indirizzo gocker.cs.unibo.it. E' possibile accedere via ssh a gocker solo da una macchina di laboratorio. Si deve dunque accedere via ssh prima ad una macchina di laboratorio (es. eva.cs.unibo.it) e da quella macchina accedere sia ssh a gocker.cs.unibo.it.
  - I siti web studenti per ragioni storiche utilizzavano una specifica macchina virtuale chiamata Golem. ***Gocker = Golem + Docker***.
  - Oggi Golem non esiste più o esiste per altri scopi, ma il nome è rimasto.
- Alla partenza Gocker fornisce una shell MOLTO minimale, senza navigazione sul file system, con pochi comandi:
  - create: crea un container con un solo servizio (node o php o python)
  - list: fornisce un elenco di tutte le directory dell'utente su cui è possibile o è stato attivato un container
  - remove: rimuovi un container
  - restart: ferma e riavvia un container
  - exit: esci dalla shell di docker
  - help: mostra un semplice aiuto sui comandi

Ottenere un docker

- Collegarsi su https://ssl.cs.unibo.it/csservices/
- Chiedere un nuovo servizio, specificando tutti i membri del team come corresponsabili, e me come docente
- Attendere una mail di conferma dell'attivazione del servizio.

La shell di docker

Il comando più interessante è create:

- `create <image> <site> [<script>]`
- ad es.: `create node-15 site2122XX index.js`

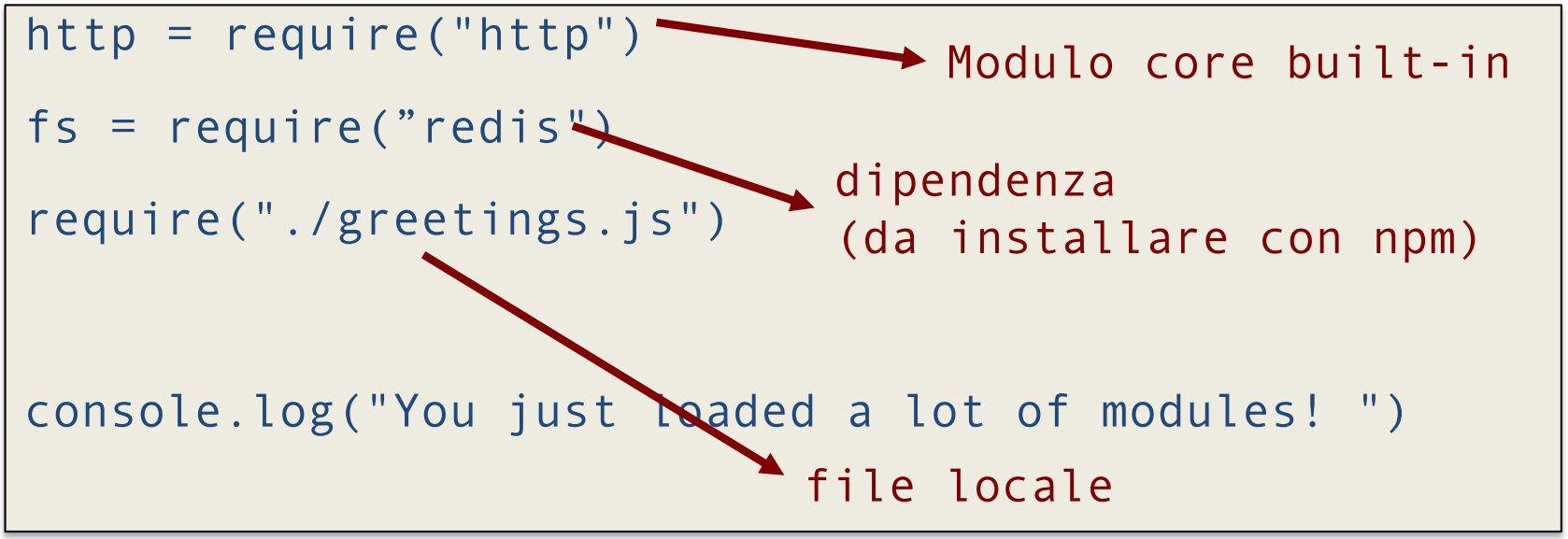
- image è uno di vari come:
  - `static (no server-side scripting)`
  - `node-15 (raccomandato)`
  - `node-14`
  - `nodemon-15 (raccomandato)`
  - `mongo (mongodb 4.4.5, raccomandato)`
- site è il dominio di V livello associato al servizio
  - `http://site2122XX.tw.cs.unibo.it/`
- Ricordarsi che la nostra configurazione di Docker prevede che la porta aperta per node.js sia la porta 8000!
- script è utile solo per node, ed è lo script javascript staticamente associato all'applicativo node, che fornisce le funzionalità attive.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Moduli NPM preinstallati

I moduli di node

- Node è estremamente modulare. E' possibile mettere codice in file javascript ed eseguirlo secondo necessità.
- Il meccanismo di caricamento di moduli di node è semplice:
  - un modulo è un file Javascript.
  - Quando si richiede un modulo, esso viene cercato localmente o globalmente secondo un modello preciso.
- I moduli vengono caricati con `require()`.



npm

- I moduli di node vengono distribuiti ed installati con npm (*node package manager*)
- npm viene eseguito via command-line e interagisce con il registro npm.
  - meccanismo robusto per gestire dipendenze e versioni dei pacchetti (moduli)
  - semplice processo di pubblicazione di pacchetti e condividerli con altri utenti.
- In più: una piattaforma per repository pubbblici e privati, servizi enterprise (integrati con firewall aziendali), integrazione con altre piattaforme, ecc.

Usare npm

Un pacchetto npm è semplicemente un insieme di file, il più importante dei quali è *package.json*, che contiene metadati sul pacchetto (contenuto, dipendenze, ecc.)

Comandi utili di npm:

- npm init
  - Genera un file package.json globale nella directory in cui si trova. Tipicamente è la directory in cui eseguire node.
  - Viene anche creata la directory node_modules in cui verranno posizionati i package installati.
- npm install [package]
  - installa il pacchetto specificato, comprese tutte le dipendenze specificate, nella directory node_modules
- npm uninstall [package]
  - rimuove il pacchetto specificato, comprese tutte le dipendenze specificate e non usate da altri pacchetti installati
- npm update [package]
  - aggiorna esplicitamente il pacchetto specificato, comprese tutte le dipendenze specificate.

Express.js

- Express è un modello di server web per node.js
  - Open-source, licenza MIT
  - molto usato e molto supportato dalla comunità
  - molto semplice e molto espandibile con plugin
- Express si dedica al routing, alla gestione delle sessioni, all'autenticazione degli utenti, e molti altri principi fondanti delle connessioni HTTP.
- Nato nel 2010, acquistato nel 2015 da IBM, poi regalato da IBM alla Node.js Foundation.

Express/cors

Permette di definire server Express che realizzano completamente le funzionalità CORS (*Cross Origin Resource Sharing*).

- Per ragioni di sicurezza, un documento HTML può ricevere risorse (immagini, dati Ajax, script, ecc.) solo da un server posto nello stesso dominio del documento HTML di origine (*Same Origin Policy*).
- E' a discrezione del server permettere ad un'applicazione di un dominio diverso di richiedere le proprie risorse (*Cross Origin Resource Sharing*).
- Il browser, subito prima di eseguire un collegamento Ajax ad una risorsa di un dominio diverso, esegue un OPTION al server.
- Affinché sia poi possibile eseguire la richiesta, il server deve rispondere con alcune intestazioni HTTP specifiche, altrimenti il browser non effettuerà la richiesta vera e propria.
- Tipicamente: `Access-Control-Allow-Origin: *`

Il package express/cors inserisce automaticamente la risposta cors più appropriata e gestisce la creazione di servizi server-side aperti a tutti i richiedenti.

Handlebar.js

- Handlebar.js è una semplice estensione di Mustache.js
- Un linguaggio di template logicless per tenere viste e codice separati
- Permette di accedere agli oggetti annidati dentro alle variabili da interpolare e di ottenere l'effetto di cicli e istruzioni condizionali usando bocchi contestuali.
- https://handlebarsjs.com/

Passport.js (non pre-installato)

- L'autenticazione può venire realizzata da package aggiuntivo di express.
  - Ce ne sono dozzine
  - Alcune autenticazioni base sono incluse direttamente in Express
  - Ad esempio *HTTP basic digest authentication*
- Passport.js è uno dei package più adottati
  - Flessibile e modulare
  - Supporta numerosissimi modelli di autenticazione: HTTP digest, Facebook, Google, Twitter, LinkedIn, ecc., chiamati *strategie*.
  - Permette anche di aggiungere nuove strategie ad-hoc.

nodemon

- Una semplicissima, utilissima utility per node.
- Node.js fa prefetch e cache degli script da eseguire. Quindi ogni modifica sugli script non viene percepita ed usata da node.js fino al riavvio, che va fatto a mano ogni volta.
- In fase di debugging, questo significa che node va riavviato a mano ogni pochi minuti, che può essere sgradevole e pesante.
- Nodemon tiene sotto controllo tutti i file di una directory, e ogni volta che percepisce il salvataggio di una modifica riavvia node.js automaticamente.
- Attenzione!
  - Voi scrivete il file su una delle macchine del laboratorio;
  - Il file system di gocker viene avvisato della modifica dopo 2-3 secondi;
  - Nodemon si accorge della modifica dopo 1 secondo e riavvia node.js;
  - Node.js richiede 3-4 secondi per riavviarsi.
- Nel frattempo il vostro sito darà errore 500 Internal Server Error
- Ci vogliono mai meno di 5 secondi, e a volte anche 10 secondi per rivedere il vostro sito online!!!

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Mongo

MongoDB

- In generale, un'applicazione web, anche data-oriented, non ha bisogno di un DB né SQL né di altro tipo, poiché tutti i linguaggi server-side (da Perl a PHP a node.js) forniscono metodi per utilizzare nativamente il file system e la applicazione può scrivere i suoi dati persistenti su un file qualunque.
- Si usa un DB:
  - quando il file diventa troppo grande per stare tutto in memoria,
  - quando ho bisogno di metodi veloci per cercare un sottoinsieme di dati all'interno del file,
  - quando ho bisogno di gestire accessi concorrenti in scrittura.
- Usare un DB per leggere, ma non modificare, un file intero di dimensioni non enormi è come comprare una Ferrari per andare a far la spesa. Funziona, ma è complicato, ed è uno spreco di tempo e risorse.

MongoDB - Introduzione

- MongoDB è un database NoSQL orientato ai documenti.
- Esso memorizza i documenti in JSON, formato basato su JavaScript e più semplice di XML, ma comunque dotato di una buona espressività.
- I documenti sono raggruppati in collezioni che possono essere anche eterogenee. Ciò significa che non c'è uno schema fisso per i documenti. Tra le collezioni non ci sono relazioni o legami garantiti da MongoDB.
- MongoDB si adatta a molti contesti, in generale quando si manipolano grandi quantità di dati eterogenei e senza uno schema.
- MongoDB fornisce un driver per JavaScript **lato server** ossia per **Node.js**. Sulle macchine di cs è preinstallato, sulle vostre dovrete installarlo di persona.

MongoDB - CRUD

- Un database è un insieme di collezioni che corrisponde ad un insieme di file nel disco fisso.
- Un'istanza di MongoDB può gestire più database. Di seguito mostreremo brevemente come effettuare le operazioni CRUD (Create, Read, Update, Delete).
- Su MongoDB, non esiste alcun comando esplicito per creare un database.
- Anche le collezioni vengono create implicitamente al primo utilizzo.

MongoDB - CRUD

- Per inserire dati:

```
try {
   let dbname ="mySite" ;
   let coll = "myCollection" ;
   let data = {
     name: "Fabio Vitali",
     age: 28,
     url: "http://www.fabiovitali.it",
     tags: ["Development", "Design"]
   } ;

   let result = await db[dbname]
     .collection[coll]
     .insert(data);
} catch (e) {
   alert("something went wrong")
}
```

```
try {
   let dbname ="mySite" ;
   let coll = "myCollection" ;
   let data = [{
     name: "Fabio Vitali",
     age: 28,
     url: "http://www.fabiovitali.it",
     tags: ["Development", "Design"]
}, {
     name: "Angelo Di Iorio",
     age: 26,
     url: "http://www.angelodiiorio.it"
     tags: ["Programming", "Systems"]
   }] ;

   let result = await db[dbname]
     .collection[coll]
     .insertMany(data);
} catch (e) {
   alert("something went wrong")
```

- La risposta corretta del server sarà tipo:

```
{
    "acknowledged": true,
    "insertedCount": 1,
    "insertedIds": {
        "0":"62694126e084590335f12f44"
    }
}
```

che dice che è stato inserito un record nella collezione.

MongoDB - CRUD

Per interrogare tutti i documenti presenti nella collezione:

```
db[dbname].collection[coll].find(query)
```

dove query è un oggetto complesso composto da valori, nomi di campo, e operatori (introdotti da $) e il nesting si ottiene attraverso contenimento.

SQL query	Mongo query
`SELECT * FROM dbname WHERE name = "Fabio Vitali"`	`db[dbname].collection[coll].find({ name: "Fabio Vitali" })`
`SELECT * FROM dbname WHERE age > 18`	`db[dbname].collection[coll].find({ age: { $lt: 18 } })`
`SELECT * FROM dbname WHERE name LIKE "fabio%" AND age > 18`	`db[dbname].collection[coll].find({ name: /^fabio/, age: { $lt: 18 } })`
`SELECT * FROM dbname WHERE name LIKE "fabio%" OR age > 18`	`db[dbname].collection[coll].find({ $or: { name: /^fabio/, age: { $lt: 18 } } })`

MongoDB - CRUD

L'aggiornamento dei documenti avviene tramite i metodi **updateOne, updateMany*(filter,update,options), e replaceOne***:

Mongo update

```
db[dbname].collection[coll].updateOne(
    {  name: "Fabio Vitali" },
    {
        $set: { age: 35 }
    })
```

```
db[dbname].collection[coll].updateMany(
    {
        age: { $lt: 40 }
    },
    {
        $set: {
           age: 40,
           status: "presumed"
        }
     }
})
```

MongoDB - CRUD

L’eliminazione di un documento dalla collezione viene effettuata per mezzo dei metodi **deleteOne** o **deleteMany**.

Mongo delete

```
db[dbname].collection[coll].deleteOne(
    {  name: "Fabio Vitali" }
)
```

```
db[dbname].collection[coll].updateMany(
    { age: { $lt: 40 } }
})
```

MongoDB - Server

Ad esempio, per accedere ad una collection su un database di MongoDB la sintassi è la seguente:

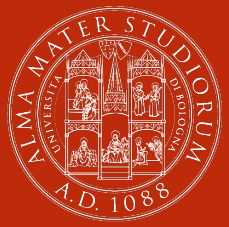
```
var client = require('mongodb').MongoClient;
client.connect("mongodb://site2122XX:[PWD]@mongosite2122XX?writeCon
cern=majority",
    async function(error, db) {
        if(!error) {
            var people = db.collection("people");
            await people.insert(
                { nome: "Fabio", cognome: "Vitali" });
            db.close();
        }
    }
);
```

MongoDB e il progetto

- Bisogna lanciare MongoDB su un docker separato.
- Usate gocker per lanciare MongoDB.

  create mongoDB site2122XX
- Il server risponde:

  Mongodb username: site2122XX - Mongodb password: YYYYYYY
  You can connect your mongodb from your site web using hostname mongo_site2122XX
- Questi tre dati vanno conservati ed utilizzati dallo script su node.js per collegarsi con successo.
- Mongo non è accessibile all'esterno, ma solo alla vostra installazione gocker.



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Fabio Vitali

Dipartimento di Informatica – Scienze e Ingegneria
Alma mater – Università di Bologna

Fabio.vitali@unibo.it