

Pagina 1 di 11 	ITIS Antonio Meucci Firenze	Rev 1.0 Data, 20/11/2020

Specifica dei Requisiti progetto WebServer	<b>Application Note</b>

Data 20/11/20  
Versione 01  
Classificazione Public Release  
Nome del File SpecificaDeiRequisiti.doc

ITIS Antonio Meucci  
Via del Filarete 17  
50143, Firenze  
Italy

## Descrizione Documento

Cliente	Scialpi, Benvenuti
Progetto	WebServer
Titolo	Specifica dei Requisiti progetto WebServer
Tipologia Documento	Application Note
Numero Documento	1
Versione	1
Data	20.11.20
Classificazione	Public Release
Autore(i)	Mathilde Patrissi
Approvato da	Mathilde Patrissi

## Approvazione Documento

Data	Nome	Titolo	Firma
20/11/20	Mathilde Patrissi		

## Indice dei Contenuti

1 Introduzione.....	4
<b>1.1 Descrizione del documento.....</b>	<b>4</b>
<b>1.2 Descrizione del contesto.....</b>	<b>4</b>
2 Requisiti del progetto.....	4
<b>2.1 Requisiti funzionali del progetto.....</b>	<b>5</b>
<b>2.2 Requisiti non funzionali del progetto.....</b>	<b>11</b>
3 Ulteriori requisiti funzionali del prodotto.....	11

# 1 Introduzione

## 1.1 Descrizione del documento

Il documento è strutturato in tre capitoli il cui contenuto può essere riassunto come segue.

Nel primo capitolo (il presente) vengono riportate le informazioni di carattere generale che permettono di identificare il progetto a cui questo documento si applica.

Nel secondo capitolo vengono riportati i requisiti funzionali del progetto in questione.

Nel terzo capitolo vengono riportati ulteriori requisiti funzionali del progetto da implementare in una successiva fase di sviluppo.

## 1.2 Descrizione del contesto

Utilizzando il linguaggio di programmazione java è stato realizzato un web server partendo dal codice fornito al seguente sito: <https://ssaurel.medium.com/create-a-simple-http-web-server-in-java-3fc12b29d5fd>.

Dopo il suo avvio, l'applicazione si mette in ascolto su una porta specificata in un file di configurazione e potrà così fornire alcune risorse all'utente. Queste ultime si trovano a loro volta in una directory specificata nel medesimo file di configurazione

# 2 Requisiti del progetto

Lo sviluppo del progetto è stato affrontato prendendo come riferimento le informazioni e le richieste fornite dai professori, di seguito riportate:

- 1) Realizzare un semplice web server in grado di accettare richieste da un client
- 2) Alla richiesta di un URL che non ha estensione e che non termina con /, se il file richiesto non esiste, invece che restituire uno status-code 404, deve essere restituito uno status-code 301 verso una location uguale all'URL richiesto con / aggiunto alla fine.  
Esempio: <http://localhost:8080/pippo> -> <http://localhost:8080/pippo/>
- 3) Alla richiesta dell'URL `"/PuntiVendita.xml"` deve essere restituito l'apposito XML.
- 4) Aggiungere un path speciale al web server (es.: `/db`). Alla richiesta del path `/db`:
  - effettuare una query di select ad una tabella mysql di esempio (nome, cognome)
  - leggere il contenuto delle righe all'interno di una lista di oggetti java con i campi nome e cognome

- serializzare la lista in xml e json esempi:

a) /db/xml restituisce i dati serializzati in xml

b) /db/json restituisce i dati serializzati in json

5 ) lanciare in esecuzione il web server sulla macchina virtuale codeanywhere.

## 2.1 Requisiti funzionali del progetto

L'applicativo in questione è stato realizzato in java e utilizzando il meccanismo dei Thread.

All'avvio l'applicazione si mette in ascolto su una porta specificata in un file di configurazione (JavaHTTPServer.json) e potrà così fornire risorse all'utente. Le risorse vengono richieste dall'utente tramite protocollo http, quindi con un semplice browser l'utente potrà effettuare le richieste.

L'applicativo preleva dal file di configurazione JavaHTTPServer.json anche altri parametri necessari al funzionamento:

```
{  
  "serverInfo": "Server: Java HTTP Server from Mathy : 1.0",  
  "srvPORT": "3000",  
  "webroot": "/var/www1",  
  "redirectTO": "http://localhost:3000/newPath",  
  "mysqluser": "mathy",  
  "mysqlpassw": "mathy",  
  "mysqlhost": "localhost",  
  "mysqlport": "3306",  
  "mysqldb": "tipsit"  
}
```

e di seguito illustrati:

```
"serverInfo": "Server: Java HTTP Server from Mathy : 1.0",  
"srvPORT": porta di ascolto dell'applicativo,  
"webroot": document root (directory in cui sono presenti le risorse),  
"redirectTO": url per il redirect,
```

"mysqluser": username per l'accesso a Mysql,

"mysqlpasswd": password per l'accesso a Mysql,

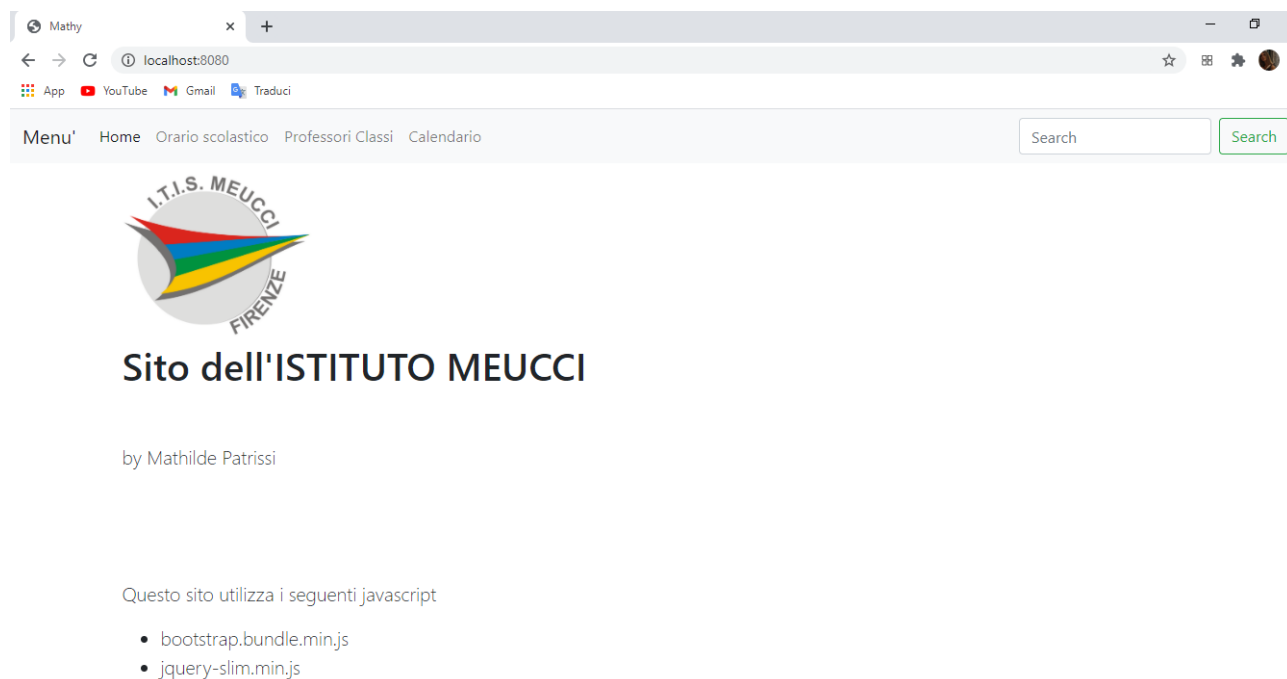
"mysqlhost": nome host del server su cui si trova Mysql (localhost se è sullo stesso server dell'applicativo),

"mysqlport": porta su cui è in ascolto Mysql,

"mysqlpdb": nome database su cui si trova la tabella utilizzata

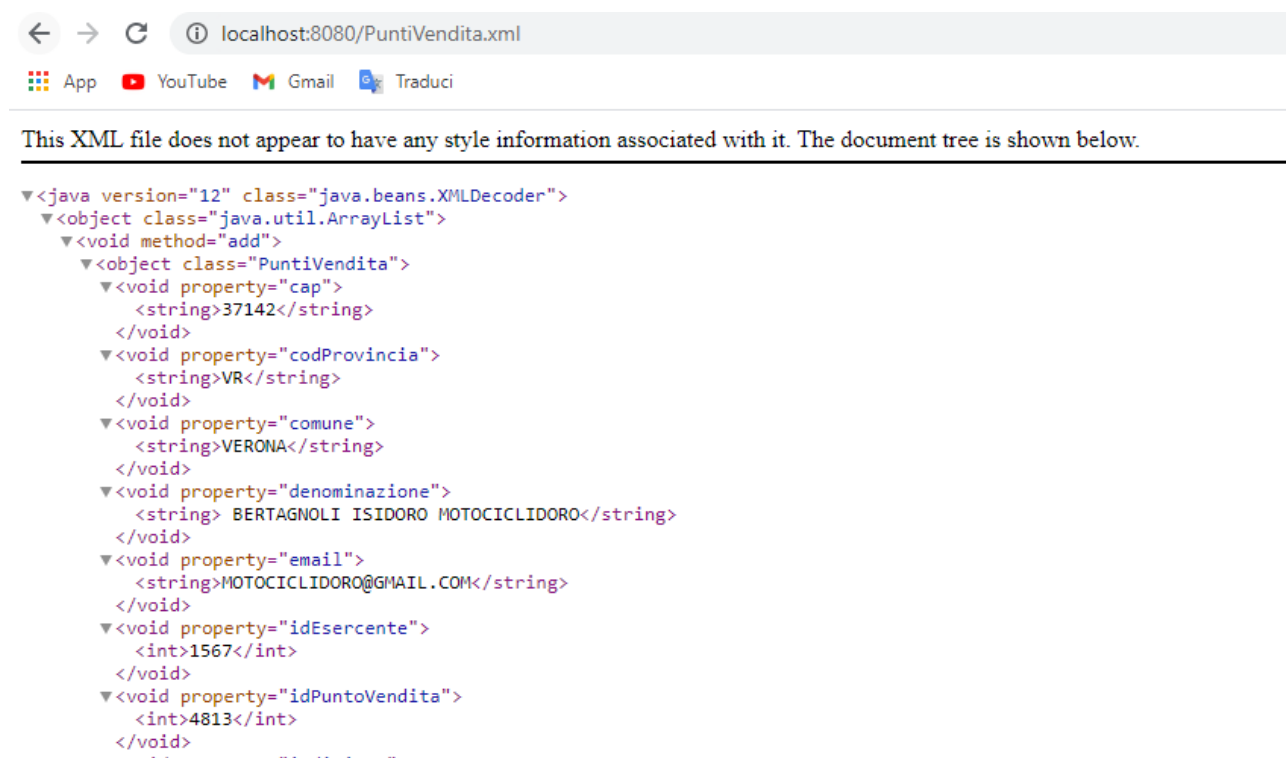
## Richiesta risorse html e txt

E' possibile richiedere all'applicativo risorse html o txt. Queste si trovano in una directory specificata nel medesimo file di configurazione. Nell'immagine seguente è mostrato il risultato della richiesta del file index.html



## Richiesta risorse XML

E' possibile richiedere come risorsa un file dei punti vendita in formato XML: alla richiesta dell'URL `"/PuntiVendita.xml"` viene restituito l'apposito XML partendo da un file json. Nell'immagine seguente è mostrato il risultato di tale richiesta.



## Richiesta risorse XML e JSON da database

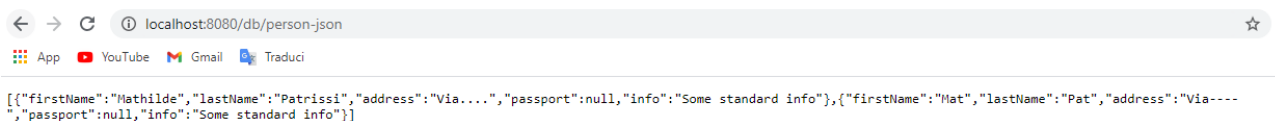
E' possibile richiedere come risorsa un file con informazioni di persone contenuti in un database in formato XML e JSON: alla richiesta dell'URL `"/db/person-xml"` o `"/db/person-json"` viene restituito l'apposito XML, o l'apposito JSON, prelevandolo da un opportuno database.

Per effettuare l'accesso al database l'applicativo preleva i dati necessari (username,password, nome database, nome server e porta di ascolto) dal file di configurazione. Come database è stato utilizzato l' RDBMS Mysql.

La tabella da cui l'applicativo preleva le informazioni ha la seguente struttura:

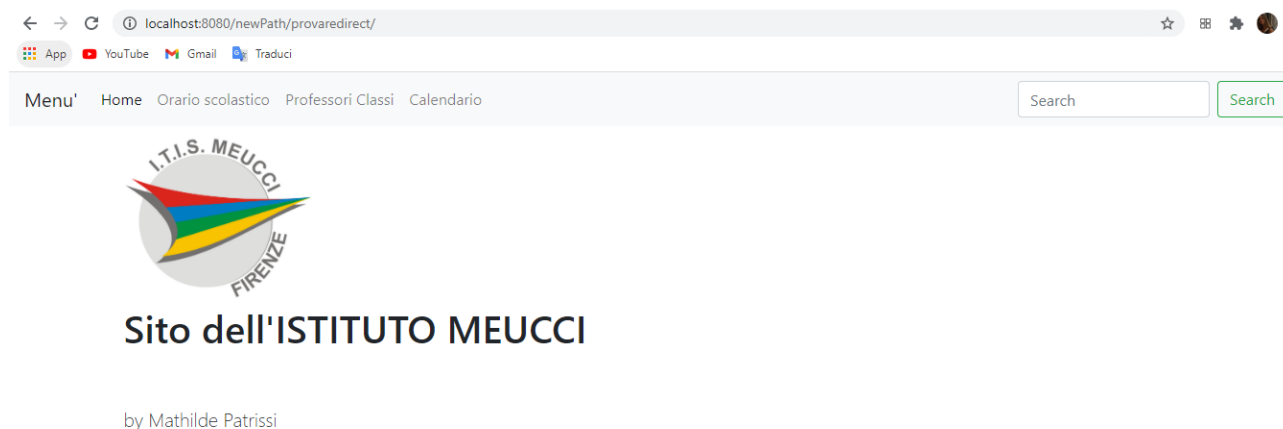
#	Nome	Tipo di dati	Lunghezza/set	Senza segno	Permetti NULL	Riempi con zero	Predefinito
1	id	INT	11	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT
2	firstName	VARCHAR	50	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Nessun valore predefi...
3	lastName	VARCHAR	50	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Nessun valore predefi...
4	address	TEXT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
5	passport	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

Nell'immagine seguente è mostrato il risultato della richiesta dell'URL "/db/person-json"



## Gesione redirect

Alla richiesta di un URL che non ha estensione e che non termina con /, se il file richiesto non esiste, invece che restituire uno status-code 404, viene restituito uno status-code 301 verso una location uguale all'URL richiesto con / aggiunto alla fine. La location di redirect è prelevata dal file di configurazione dell'applicativo. Nell'immagine seguente è possibile vedere un esempio:



Questo sito utilizza i seguenti javascript

- bootstrap.bundle.min.js
- jquery-slim.min.js



E' possibile gestire la lingua desiderata dei messaggi. Quest'ultima funzionalità è stata realizzata tramite l'utilizzo della classe `java.util.ResourceBundle` che ci permette di predisporre il supporto alle principali lingue.

E' stata pertanto realizzata la classe `MessagesBundle` che, alla scelta della lingua desiderata, seleziona l'opportuno bundle di risorse per la gestione di tale lingua. Un bundle di risorse è un file `.properties` Java che contiene dati specifici della lingua. È un modo di internazionalizzare l'applicazione Java rendendo il codice indipendente dalla lingua locale.

## Gestione dei Log

Per mettere a punto un sistema di logging per tenere sotto controllo l'applicativo sia in fase di test che di normale funzionamento è stata utilizzata la libreria java `Log4j` : in questo modo è possibile scrivere messaggi sulla console o su un file di testo.

Per specificare se scrivere i messaggi a console, su un file di testo e il livello di logging è necessario configurare un opportuno file di configurazione della libreria `Log4j`

La seguente tabella definisce i livelli dei log e i messaggi in `log4j` in ordine decrescente di severità. La colonna di sinistra indica il livello di log designato e alla destra c'è una breve descrizione.

Livello	Descrizione
<b>OFF</b>	Il livello più alto possibile, viene usato per disattivare i log.
<b>FATAL</b>	Errore importante che causa un prematuro termine dell'esecuzione. Ci si aspetta che questo sia visibile immediatamente all'operatore.
<b>ERROR</b>	Un errore di esecuzione o una condizione imprevista. Anche questo deve essere immediatamente segnalato.
<b>WARN</b>	Usato per ogni condizione inaspettata o anomalia di esecuzione, che però non necessariamente ha comportato un errore.
<b>INFO</b>	Usato per segnalare eventi di esecuzione (esempio: startup/shutdown). Deve essere segnalato ma poi non mantenuto per tanto tempo.
<b>DEBUG</b>	Usato nella fase di debug del programma. Viene riportato nel file di log.
<b>TRACE</b>	Alcune informazioni dettagliate. Ci si aspetta che venga scritto esclusivamente nei file di log. È stato aggiunto nella versione 1.2.12.

## Simulazione ambiente

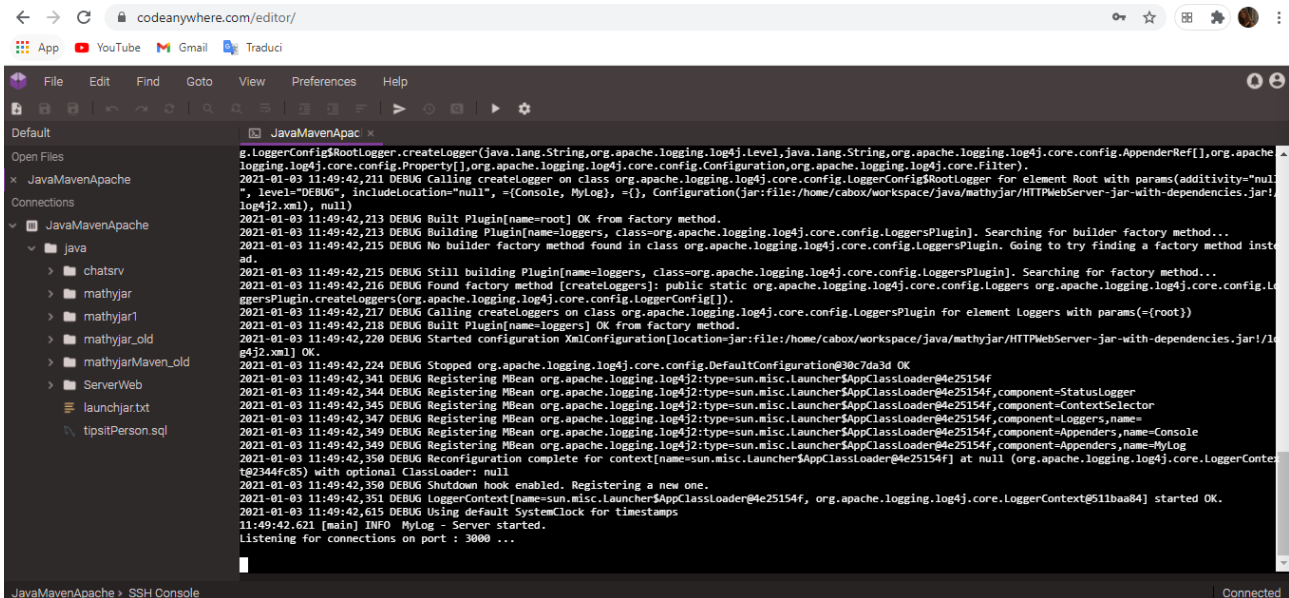
Per simulare un ambiente reale è stato utilizzato `Codeanywhere`.

`Codeanywhere` è un IDE (integrated development environment) cross-platform basato sul cloud, creato da `Codeanywhere, Inc.`, che permette agli utenti di scrivere codice, modificarlo e lanciare

progetti di sviluppo web in modalità collaborativa, il tutto su di un browser web e ogni dispositivo mobile.

Abbiamo creato un container Ubuntu con Java e Maven, sul quale abbiamo installato Mysql.

Tramite git abbiamo scaricato il repository dell'applicativo in questione e compilato tramite maven.

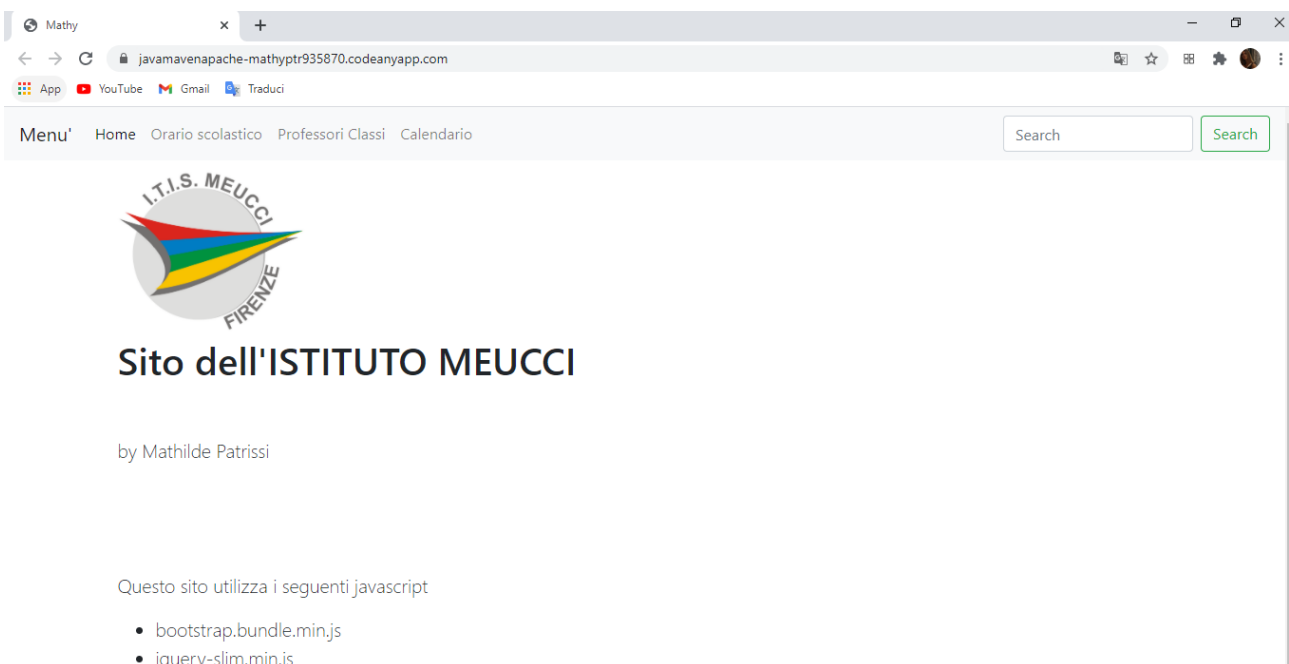


```
2021-01-03 11:49:42,211 DEBUG Calling createLogger on class org.apache.logging.log4j.core.config.LoggerConfig$RootLogger for element Root with params(additivity="null", level="DEBUG", includeLocation="null", =(Console, MyLog), =((), Configuration(jar:file:/home/cabox/workspace/java/mathjar/HTTPWebServer-jar-with-dependencies.jar!/log4j2.xml), null))
2021-01-03 11:49:42,213 DEBUG Built Plugin[name=root] OK from factory method.
2021-01-03 11:49:42,213 DEBUG Building Plugin[name=loggers, class=org.apache.logging.log4j.core.config.LoggersPlugin]. Searching for builder factory method...
2021-01-03 11:49:42,215 DEBUG No builder factory method found in class org.apache.logging.log4j.core.config.LoggersPlugin. Going to try finding a factory method instead.
2021-01-03 11:49:42,215 DEBUG Still building Plugin[name=loggers, class=org.apache.logging.log4j.core.config.LoggersPlugin]. Searching for factory method...
2021-01-03 11:49:42,217 DEBUG Calling createLoggers on class org.apache.logging.log4j.core.config.LoggersPlugin for element Loggers with params(=(root))
2021-01-03 11:49:42,218 DEBUG Built Plugin[name=loggers] OK from factory method.
2021-01-03 11:49:42,220 DEBUG Started configuration XmlConfiguration[location=jar:file:/home/cabox/workspace/java/mathjar/HTTPWebServer-jar-with-dependencies.jar!/log4j2.xml] OK.
2021-01-03 11:49:42,224 DEBUG Stopped org.apache.logging.log4j.core.config.DefaultConfiguration@30c7da3d OK
2021-01-03 11:49:42,341 DEBUG Registering MBean org.apache.logging.log4j2:type=sun.misc.Launcher$AppClassLoader@4e25154f
2021-01-03 11:49:42,344 DEBUG Registering MBean org.apache.logging.log4j2:type=sun.misc.Launcher$AppClassLoader@4e25154f, component=StatusLogger
2021-01-03 11:49:42,345 DEBUG Registering MBean org.apache.logging.log4j2:type=sun.misc.Launcher$AppClassLoader@4e25154f, component=ContextSelector
2021-01-03 11:49:42,347 DEBUG Registering MBean org.apache.logging.log4j2:type=sun.misc.Launcher$AppClassLoader@4e25154f, component=Loggers, name=
2021-01-03 11:49:42,349 DEBUG Registering MBean org.apache.logging.log4j2:type=sun.misc.Launcher$AppClassLoader@4e25154f, component=Appenders, name=Console
2021-01-03 11:49:42,350 DEBUG Reconfiguration complete for context[name=sun.misc.Launcher$AppClassLoader@4e25154f] at null (org.apache.logging.log4j.core.LoggerContext@4e2344ec85) with optional ClassLoaders: null
2021-01-03 11:49:42,350 DEBUG Shutdown hook enabled. Registering a new one.
2021-01-03 11:49:42,351 DEBUG LoggerContext[name=sun.misc.Launcher$AppClassLoader@4e25154f, org.apache.logging.log4j.core.LoggerContext@511baa84] started OK.
2021-01-03 11:49:42,615 DEBUG Using default SystemClock for timestamps
11:49:42.621 [main] INFO MyLog - Server started.
Listening for connections on port : 3000 ...
```

Abbiamo messo il web server in ascolto sulla porta 3000 in modo che fosse raggiungibile esternamente attraverso il seguente url:

<https://javamavenapache-mathyptr935870.codeanyapp.com>

Il risultato è mostrato nell'immagine seguente:



## 2.2 Requisiti non funzionali del progetto

### **Requisiti di prodotto**

Il progetto deve essere realizzato tramite il linguaggio di programmazione java e utilizzando come punto di partenza il codice presente alla pagina web <https://ssaurel.medium.com/create-a-simple-http-web-server-in-java-3fc12b29d5fd> .

### **Requisiti di consegna**

Il progetto, che deve essere terminato entro il 6 Gennaio, deve essere inserito in un repository git il cui url dovrà essere consegnato entro e non oltre la data suddetta.

## 3 Ulteriori requisiti funzionali del prodotto

In una fase successiva del progetto si potrebbe ipotizzare di implementare il protocollo https.