# Java EE vs Spring Cloud

## Introduzione

La scelta di fare una tesi che contrappone due tecnologie come Java EE e Spring Cloud nasce dall’esperienza maturata in ambito lavorativo, che giornalmente mi sottopone a scegliere la tecnologia più adatta nel rispetto dei requisiti ed i vincoli contrattuali del progetto. Le nostre scelte sono sicuramente la conseguenza di un esperienza già maturata, ma non soltanto in ambito informatico ma sotto qualsiasi aspetto.

Poniamo il caso assurdo di avere due strade, già percorse altre volte, che presentano ogni volta degli ostacoli:

1. Una strada è ritenuta più sicura perché ha una mappa certificata che ci mostra esattamente come superare gli ostacoli ma dobbiamo rispettare le istruzioni esatte della mappa, se ti perdi o non riesci a superare qualche ostacolo puoi sempre chiedere aiuto.
2. Una strada è ritenuta meno sicura, perché la mappa non è certificata ma ci mostra comunque come superare tutti gli ostacoli, inoltre ci da la possibilità di scegliere come superarli e non ci lega soltanto ad una soluzione. Il problema è che se ti perdi non puoi chiedere aiuto a nessuno, devi cavartela da solo.

Se fossi il capo di un gruppo di viaggiatori e non avessi mai percorso nessuna delle due strade sicuramente sceglierei la prima e sicuramente riuscirei ad arrivare a destinazione sano e salvo già dalla prima volta. Piano piano che percorro questa strada mi accorgo del tempo impiegato a superare gli ostacoli, quindi di tutto il tempo impiegato nel tragitto. Vorrei ottimizzare, velocizzare certi passi perché chi finanzia la nostra spedizione vuole raggiungere nuovi obiettivi, la strada da percorrere si complica, gli stessi ostacoli aumentano ma i metodi per superali rimangono uguali, la nostra mappa non cambia.

Decido di intraprendere la seconda strada, magari prima di affrontarla tutta insieme ne percorro soltanto un tratto, poi un altro tratto ancora, fino ad arrivare a destinazione. Ho scoperto che ci sono molti modi di superare quegli ostacoli e che posso scegliere quello che più mi conviene. Mi accorgo che la mappa anche non essendo certificata mi porta più velocemente a destinazione e che posso utilizzarla tranquillamente per pianificare i miei prossimi viaggi.

Io come sviluppatore mi trovo nei panni del viaggiatore, che conosce la mappa certificata che è lo standard Java EE e quella non certificata che è la documentazione di Spring. La strada da percorre è il progetto, basato su Java EE nel primo caso e basato su Spring nel secondo. Gli ostacoli sono le tecnologie messe a disposizione da un Application Server come può essere Wildfly o Weblogic o Websphere. La strada è quanto più lunga quanti sono i requisiti del progetto e la difficoltà d’implementazione di certe funzionalità.

In particolare la scelta di mettere in discussione lo standard Java Enterprise nasce da esigenze che cambiano ogni giorno, ogni ora, che richiedono un implementazione veloce in ambiente di produzione e che insistono particolarmente sulla metodologia **DevOps** (parola che nasce dalla contrazione inglese di **development**, “sviluppo” e **operations**, ad intendere la “messa in produzione” o “deployment”).

*…*

*DevOps è una metodologia di sviluppo del software che punta alla comunicazione, collaborazione e integrazione tra sviluppatori e addetti alle operations dell'information technology. DevOps vuole rispondere all'interdipendenza tra sviluppo software e IT operations, puntando ad aiutare un'organizzazione a sviluppare in modo più rapido ed efficiente prodotti e servizi software.*

*…*

*L'adozione della metodologia DevOps è guidata da diversi fattori, come:*

* *Utilizzo della metodologia agile e altre metodologie di sviluppo del software*
* *Necessità di incrementare la frequenza dei rilasci in produzione*
* *Ampia disponibilità di un'infrastruttura virtualizzata[15] e in cloud*
* *Incremento nell'uso di data center automatizzati[16] e strumenti di configuration management e Infrastructure as Code*

*(Cit. https://it.wikipedia.org/wiki/DevOps)*

## Il progetto

Il progetto in esame consiste nello sviluppo di una piattaforma per il gioco del poker.

La gestione del progetto è affidata a Maven che si divide in 3 moduli:

* **wildfly**: basato sull’omonimo Application Server di casa Red Hat utilizza lo standard Java EE 7.
* **spring-cloud**: sviluppato su Spring Cloud l’attuale prodotto di Pivotal che non utilizza standard Java ma è bsato sulle librerie di Netflix OSS.
* **poker-client**: una semplice web app per il gioco del poker che permette di compiere azioni prefissate, non permette di fare una partita completa ma serve a popolare la piattaforma con le informazioni necessarie al funzionamento. Il client è utilizzabile sia per il modulo wildfly che per quello spring-cloud.

## Wildfly (Java EE)

Wildfly viene scelto come Apllication Server atto a rappresentante lo standard e quindi la tecnologia Java EE.

Il progetto Wildfly è organizzato secondo i seguenti moduli:

* **poker-game**
  + **poker-console**: web app amministrativa con accesso riservato per la gestione dei tavoli da gioco.
  + **poker-manager**: è la piattaforma di gioco al quale si connette il **poker-client** e la **poker-console**, contiene la business logic di gioco e tiene in memoria le informazioni di gioco più recenti. Si connette al back-office per reperire informazioni sui player.
* **back-office**: si occupa delle transazioni di gioco dei player, contiene il database con tutte le informazioni di account, conto gioco e pagamenti. Questa parte non conosce il tipo di gioco in questione e può essere riutilizzata per qualsiasi gioco effettui transazioni.

### Clustering

La procedura di clusterizzazione in Wildfly 10 viene effettuata grazie ai servizi messi a disposizione da **JGroups** che sfrutta le funzionalità di **Infinispan** per quanto riguarda la gestione della cache ed utilizza **ActiveMQ** per la gestione di JMS.

La clusterizzazione su cache è possibile:

* tramite la **Http Session** in ambito web.
* tramite **EJB3** nel contesto ejb.
* tramite cache di secondo livello **Hibernate** (implementazione di default JPA) su query ed entity.

La clusterizzazione basata su messaggi:

* tramite **ActiveMQ** che fa da broker JMS

### 

### Poker Console (poker-console)

### Poker Manager (poker-manager)

### Back Office (back-office)

### Disegno Architetturale

Di seguito il disegno architetturale che mostra la divisione Client/Server e I nodi in cluster organizzati su WildFly:

* Poker Client: installato su Apache funziona interamente sul browser del player, si connette al server tramite web service REST Json.
* Poker Server: è costituito da 3 cluster multinodo che comunicano fra loro tramite web service REST Json
  + Poker Console: 2 nodi che condividono la stessa Cache su Http Session
  + Poker Manager: 3 nodi in cluster tramite JMS e EJB3 Cache
  + Back Office: 3 nodi in cluster tramite JMS e JPA Cache
* Database: accessibile dal Back Office tramite il datasource di Wildfly basato su protocollo JDBC



## Spring Cloud

## Riferimenti al progetto

Il progetto scaricabile direttamente da GitHub al seguente indirizzo: <https://github.com/marcoromagnolo/javaee-vs-springcloud>

Una volta fatto il download o il clone del progetto, per effettuare la build bisogna utilizzare Maven versione >= 3 e lanciare il seguente comando dalla radice del progetto:

*maven clean install*

### Hardware

Il progetto è attualmente deployato su un VPS con le seguenti caratteristiche:

* 1 CPU
* RAM 1GB
* HD 20 GB

Le credenziali per accedere alla macchina Linux (CentOS 6.x) sono:

* ssh://212.237.5.69
* username: unimarconi
* password: Uni#2018

La directory in cui sono deployati i progetti si trova nel path /unimarconi (che è anche la HOME per l’user unimarconi)

### Database

Il database utilizzato per il salvataggio delle informazione è PostgreSQL 10

Per accedere al database ci sono 2 modi:

* collegandosi in ssh con utente unimarconi ed effettuando l’accesso da line adi comando:
  + *psql unimarconi*
* *collegandosi a phpPgAdmin all’indirizzo* [*http://212.237.5.69/phpPgAdmin*](http://212.237.5.69/phpPgAdmin)
  + *username:* unimarconi
  + *password: Uni#2018*

### Poker Client

Il client è installato su server Apache 2 come semplice applicazione HTML, CSS e Javascript.

Questi i link permetteranno di utilizzare il client di poker per la connessione al progetto Wildfly piuttosto che a quello Spring Cloud.

* Wildfly: <http://212.237.5.69/poker/wildfly>
* Spring Cloud: <http://212.237.5.69/poker/spring-cloud>

### Wildfly

Di seguito tutti i link e le credenziali per accedere agli endpoint del progetto Wildfly.

Wildfly Console

* <http://212.237.5.69:9990/console>
* username: unimarconi
* password: Uni#2018

Rest API:

* Poker Manager: <http://212.237.5.69:8080/poker-manager/api>
* Back Office: <http://212.237.5.69:8080/back-office/api>

Console di Amministrazione del Poker:

* Poker Console: <http://212.237.5.69:8080/poker-console>

### Spring Cloud

Di seguito tutti i link e le credenziali per accedere agli endpoint del progetto Spring Cloud.

Spring Cloud Console

Rest API

* Poker Manager
  + Api PM
  + Cash PM
  + Lobby PM
  + Round PM
  + Table PM
  + Tournament PM
* Back Office
  + Api BO
  + Account BO
  + Core BO
  + Wallet BO

Console di Amministrazione del Poker:

* Poker Console: