

Impariamo Spring MVC

Con Sql Server



Introduzione a Spring MVC

Che cosa è Spring MVC



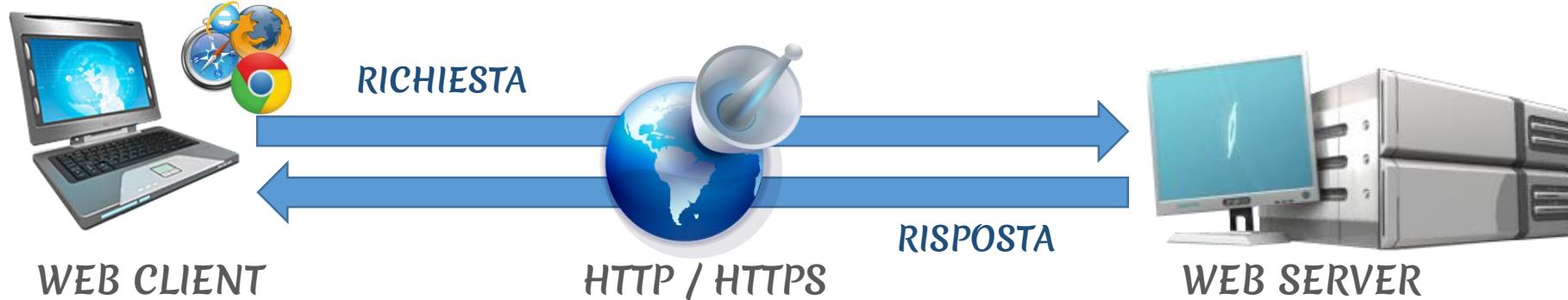
E' un framework basato su JAVA EE utilizzato per lo sviluppo di **applicazioni web**



Utilizza il paradigma MVC (Modello – Vista – Controllo)

Funzionamento delle applicazioni web

<http://www.miosito.com/index>



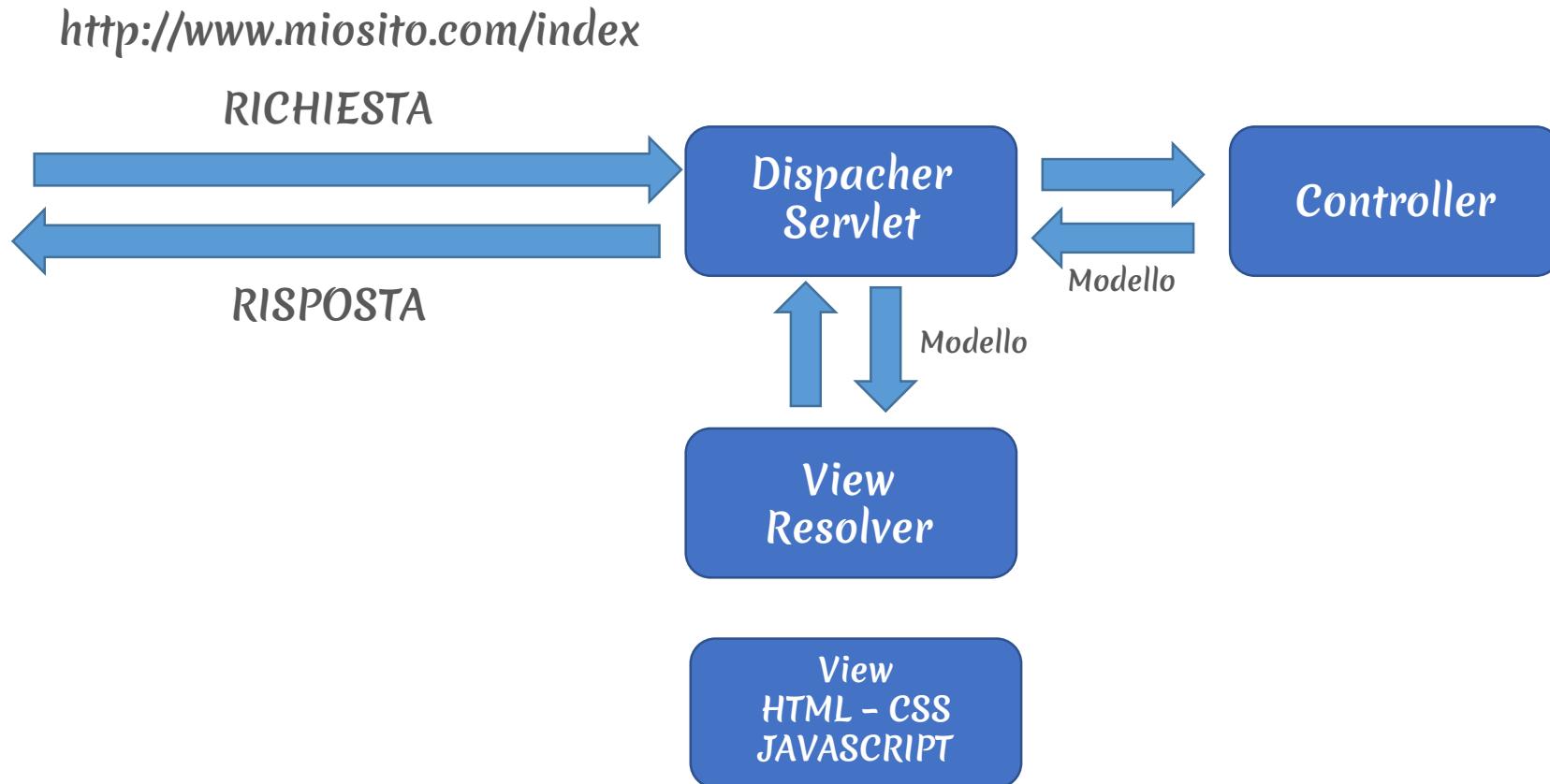
i HTTP = Hyper Text Transfer Protocol

i HTTP è Stateless

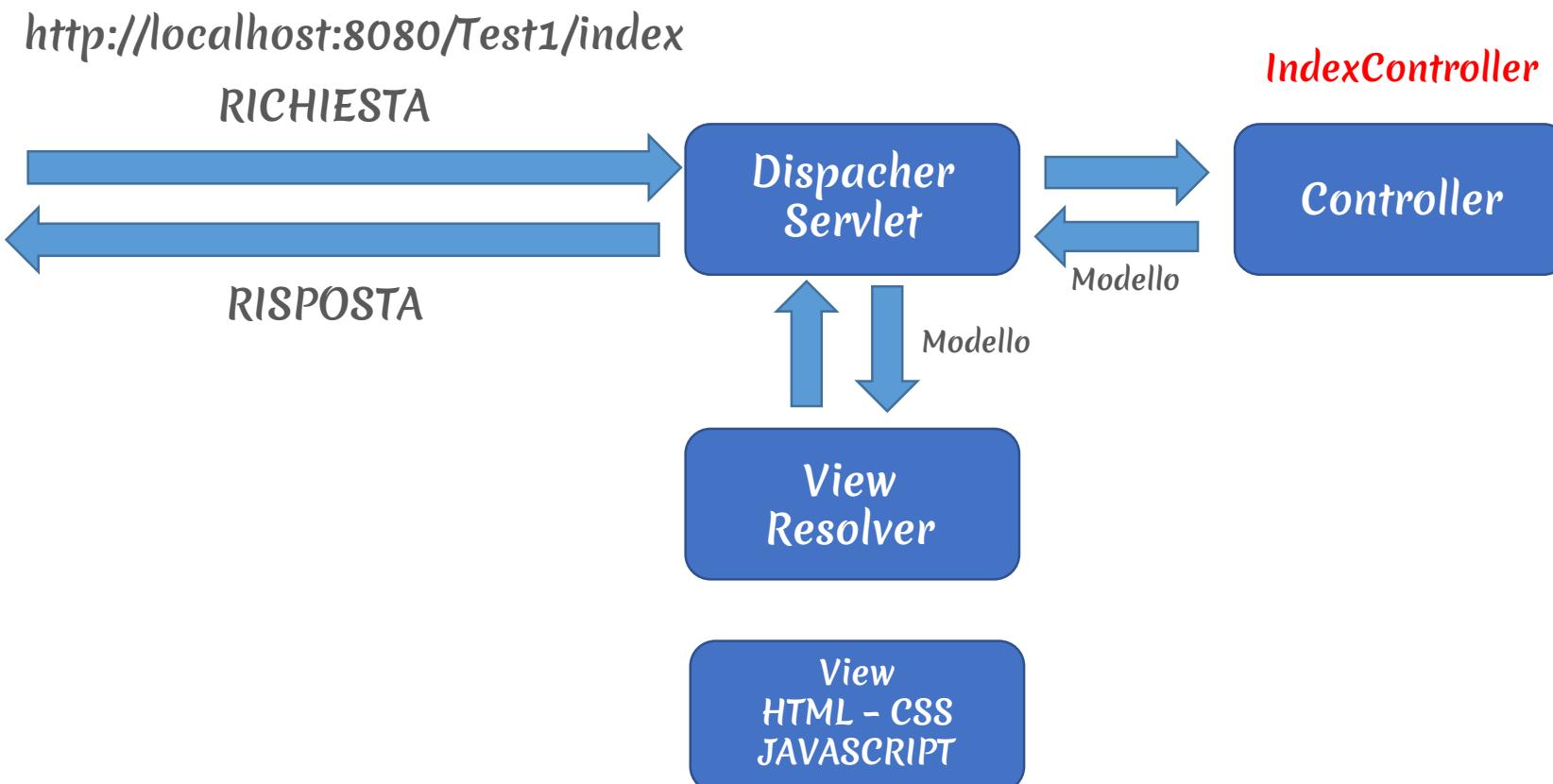
i Esistono diversi metodi con i quali si possono eseguire le richieste al server (GET – POST – DELETE - PUT)

i Browser web conoscono HTML – JAVASCRIPT - CSS

Gestione delle richieste in Spring MVC



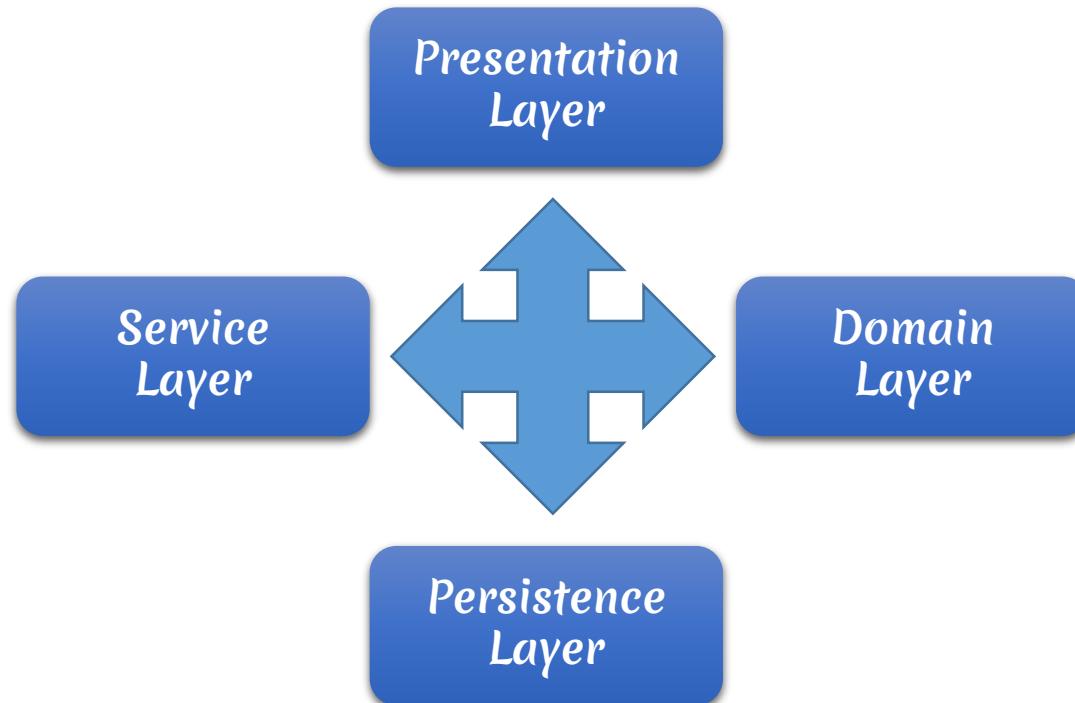
Gestione delle richieste in Spring MVC



L'architettura delle WEB Application



Il codice deve essere organizzato in **strati (layer)**



Il Domain Layer



Costituisce il **domain model** del progetto



Il domain model è la rappresentazione delle varie entità, degli attributi, dei vincoli e delle relazioni che intercorrono fra di essi

Articoli

Listini

Clienti

Ordini

Il Persistence Layer



Gestisce la modifica e l'interscambio dei dati con la base dati (database) della web application



E' composto da una o più classi responsabile delle operazioni CRUD (Create, Read, Update, Delete)



Le classi sono identificate dalla notazione **@Repository**

Il Service Layer



E' composto dalle classi che gestiscono le **business logic** della Web Application

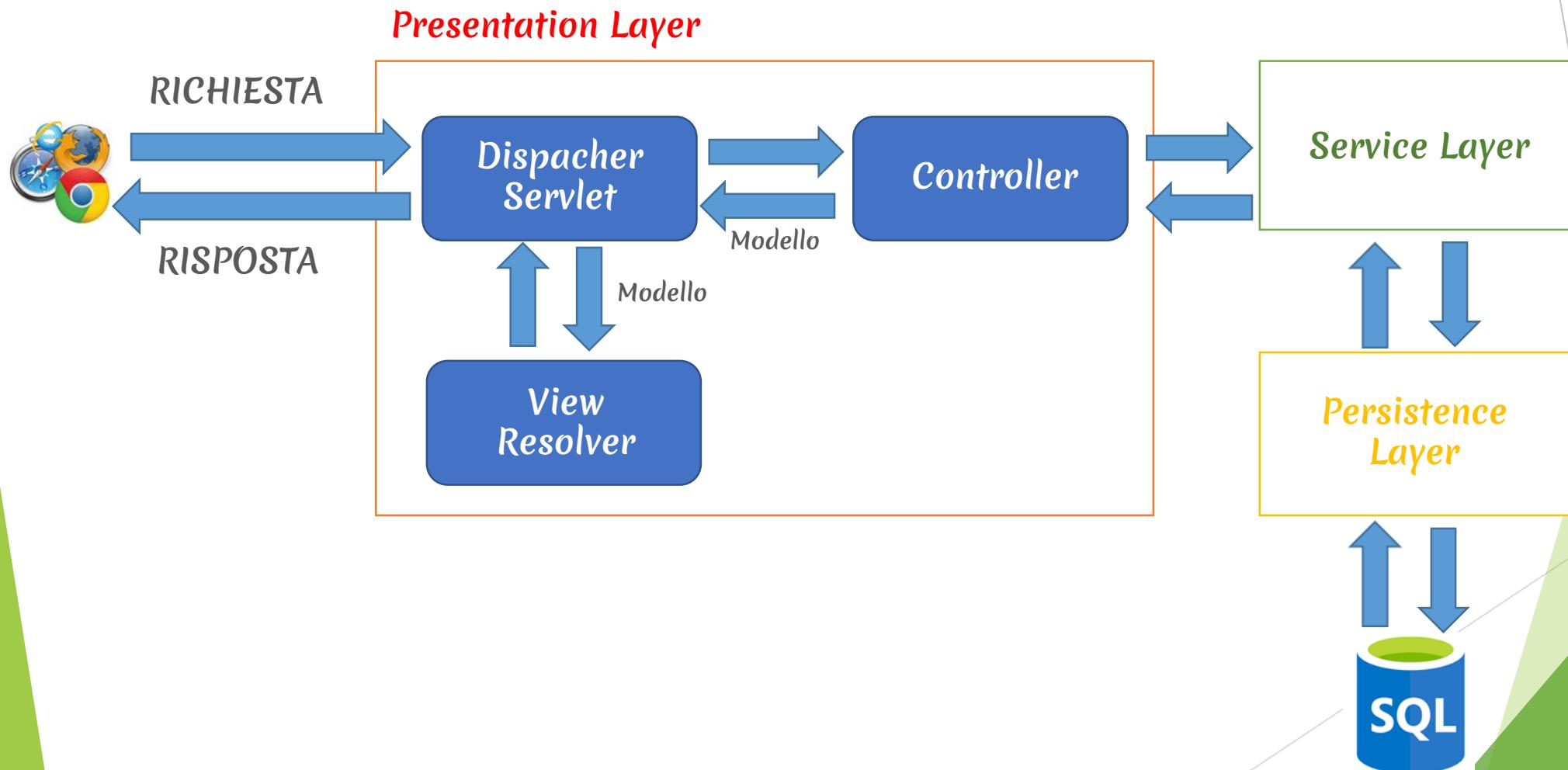


Le business logic sono le **regole sulle quali si basa il funzionamento dell'applicazione**



Le classi sono identificate dalla notazione **@Service**

Lo Schema della Web Application Spring MVC



Riepilogo Notazioni Bean Validation

Notazione	Tipo di elemento
@AssertFalse	booleano di tipo false
@AssertTrue	booleano di tipo true
@DecimalMax	numero
@DecimalMin	
@Digits	Numero con un determinato valore di decimali
@Future	data nel futuro
@Max	Numero che NON deve superare il valore inserito
@Min	Numero che deve avere un valore minimo
@NotNull	NON può essere NULL
@Null	Deve essere NULL
@Past	Data nel passato
@Pattern	Deve rispettare una determinata espressione regolare

Impariamo Spring MVC



Introduzione a
Hibernate e JPA

Che cosa è Hibernate e la JPA



Java Persistence API (JPA) è un framework che gestisce la persistenza in Java. E' leggero ed è basato su POJO

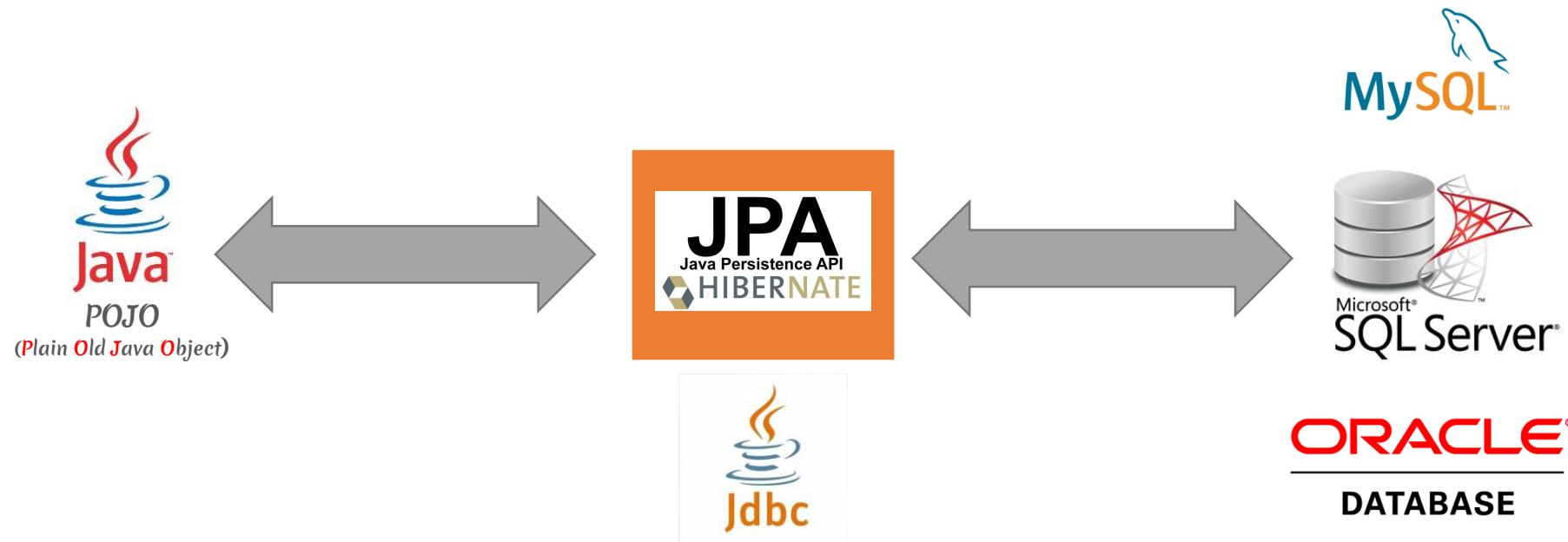


Hibernate è un framework Object Relational Mapping open source ed è il provider di JPA



Ha lo scopo di trasformare in maniera efficiente e trasparente i dati da una classe Java (Plain Old Java Object) a una tabella di un database relazionale

Schema di funzionamento di Hibernate



Creiamo le Query con JPA

Tecnologie di creazione delle query con la JPA



Java Persistence Query Language (JPQL)



Criteria API

Utilizziamo Criteria API

Selezioniamo gli elementi di una Entity (Clienti.java)

```
CriteriaBuilder cb = entityManager.getCriteriaBuilder(); ← Interfaccia «Criteria Builder»
```

```
CriteriaQuery<Clienti> queryDefinition = ← Creazione della CriteriaQuery  
cb.createQuery(Clienti.class);
```

```
Root<Clienti> recordset = queryDefinition.from(Clienti.class);
```

```
queryDefinition.select(recordset).  
where(cb.equal(recordset.get("codFidelity"), CodFidelity));
```

```
retVal = entityManager.createQuery(queryDefinition).getSingleResult();
```

Utilizziamo il JPQL

Selezioniamo gli elementi di una Entity (Clienti.java)

```
List<Clienti> retVal; ← Valore di Ritorno
```

```
String JPQL = "SELECT a FROM Clienti a WHERE a.comune = :comune";
```

```
retVal = entityManager.createQuery(JPQL) ← Creazione della Query
```

```
.setParameter("comune", Comune) ← Impostazione Parametro
```

```
.getResultList();
```

Riepilogo sintassi JPQL

Selezioniamo gli elementi di una Entity (Clienti.java)

CLAUSE	SCOPO
SELECT	Comando di selezione
FROM	Comando di origine
WHERE	Comando di filtro
ORDER BY	Comando di ordinamento

Utilizziamo Hibernate

Mappare una classe Java con le annotazioni JPA

```
@Entity  
@Table(name = "CLIENTI")  
public class Clienti implements Serializable  
{  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private long id;  
}
```

Impariamo Spring MVC



SPRING SECURITY

Introduzione a Spring Security

Elementi Chiave della Sicurezza 1



Autenticazione: Identificazione dell'utente che procede alla richiesta del servizio e/o della risorsa



Autorizzazione: Segue l'autenticazione e determina quali risorse sono accessibili all'utente

Elementi Chiave della Sicurezza 2



Credenziali di accesso al DB: Le credenziali di accesso alla fonte dati non devono essere accessibili e in chiaro



Criptare dati sensibili: Dati come i numeri di carta di credito, password o i dati clinici devono essere criptati prima di essere memorizzati nel DB



Protezione a livello di trasporto (SSL): Le informazioni che viaggiano attraverso la rete devono essere criptate e sicure

Caratteristiche dello Spring Security



E' uno dei progetti dell'ecosistema Spring. Facilmente integrabile nei progetti Spring MVC e Spring Boot



Include tutto ciò che serve per garantire la sicurezza, l'autenticazione e l'autorizzazione alle applicazioni



E' facilmente implementabile nella maggioranza dei sistemi di autenticazione enterprise

Impariamo Spring MVC



SPRING BOOT

Introduzione a
Spring Boot

Caratteristiche del Spring Boot



Configurazione Automatica: Lo Spring boot ha la capacità di generare automaticamente la configurazione delle applicazioni Spring (MVC, REST)



Dipendenze Automatiche: Spring Boot crea in automatico le dipendenze e le versioni necessarie



Actuator: Elemento che permette di controllare il comportamento e le caratteristiche della Web App in esecuzione



Deployment Semplificato: Le Web App create in Spring Boot facilmente distribuite anche in cloud

Obiettivo Primario di Spring Boot

Actuator

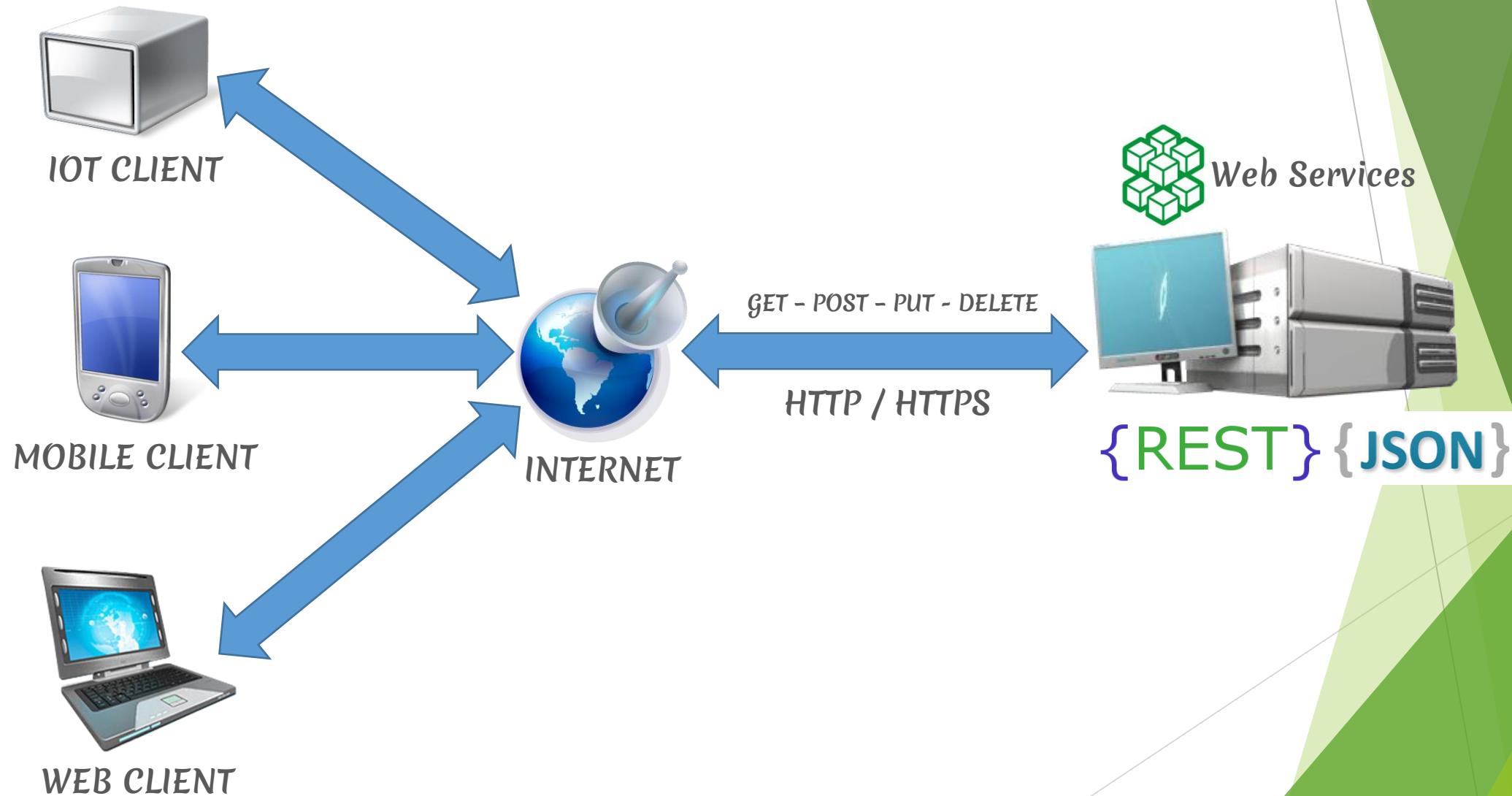
Configurazione
Automatica

Semplificare
Sviluppo

Deployment
Facilitato

Dipendenze
Automatiche

Cosa sono i Web Services



Principali Caratteristiche dei Web Services



Fruibilità: I Web Services possono essere «**consumati**» da applicazioni eterogenee



Riusabilità: Le business logic delle App possono essere facilmente soggetti a modifiche e riutilizzo delle logics



Indipendenza: I Web Services possono essere creati in ecosistema di moduli indipendenti che preservano la funzionalità primaria della Web App



Facilità nella Distribuzione: I Web Services possono essere facilmente installati e distribuiti in servizi cloud o server locali

Tipi di Web Services



SOAP: Il **Simple Object Access Protocol** è un protocollo basato sul XML, indipendente dalla piattaforma, avente lo scopo di permettere l'interscambio di dati fra client e server. Questi ultimi dialogano tra di loro attraverso il contratto **Web Service Description Language**.



REST: REpresentational State Transfer è una architettura software basata sul trasferimento dei dati in formato rappresentativo (XML, JSON o HTML), privo di intermediari e volto a soddisfare il consumatore di dati.

Vantaggi del REST



Veloce, Leggero e basato sull'HTTP: Il REST utilizza il protocollo HTTP e i suoi metodi (GET,POST,PUT) per trasferire o modificare i dati fra applicazioni eterogenee



Indipendente dalla piattaforma: Il REST può essere scritto eseguito in qualsiasi piattaforma



Multiformato: Il REST è compatibile con i formati XML, HTML, JSON, testo

Impariamo Spring MVC



SPRING DATA

Introduzione a
Spring Data JPA

Cosa è Spring Data JPA



E' un framework, usato nello **strato di persistenza**, che aggiunge un extra livello di astrazione al provider JPA al fine di minimizzare il **boiler plate code**

Scopo del Spring Data JPA



Minimizzare il **boiler plate code**



Facilitare la creazione dello strato di persistenza delle
Web APP



Rientra nella obiettivo dello Spring Boot di facilitare e
velocizzare al massimo la creazione di Web Services o
Web App MVC

Caratteristiche dello Spring Data JPA

Spring Data JPA

Spring Data Commons

JPA Provider

Caratteristiche dello Spring Data JPA

Spring Data Commons

[Repository<T, ID extends Serializable>](#)

[CrudRepository<T, ID extends Serializable>](#)

[PagingAndSortingRepository<T, ID extends Serializable>](#)

[QueryDslPredicateExecutor<T>](#)

Spring Data JPA

[JpaRepository<T, ID extends Serializable>](#)

[JpaSpecificationExecutor<T>](#)

Impariamo Spring MVC

JUnit

Introduzione al
Unit Test e al JUnit

Cosa è il test



Un test è la misura dell'**efficacia** e dell'**efficienza** di un determinato elemento



Lo scopo del test è quello di verificare che dal software si ottengano i risultati attesi (**Efficacia**)



Lo scopo di altri test è quello di verificare che i risultati attesi si ottengano col minor dispendio di risorse (**Efficienza**)

Benefici dei test automatici



I test automatici sono il sistema più veloce per trovare anomalie nel funzionamento del software



Trovare i difetti a monte del delivery riduce i costi di manutenzione e modifica (**prima viene trovato il difetto minori sono i costi**)



I test rappresentano una efficace **documentazione non obsoleta** del software, del suo funzionamento e delle sue logiche

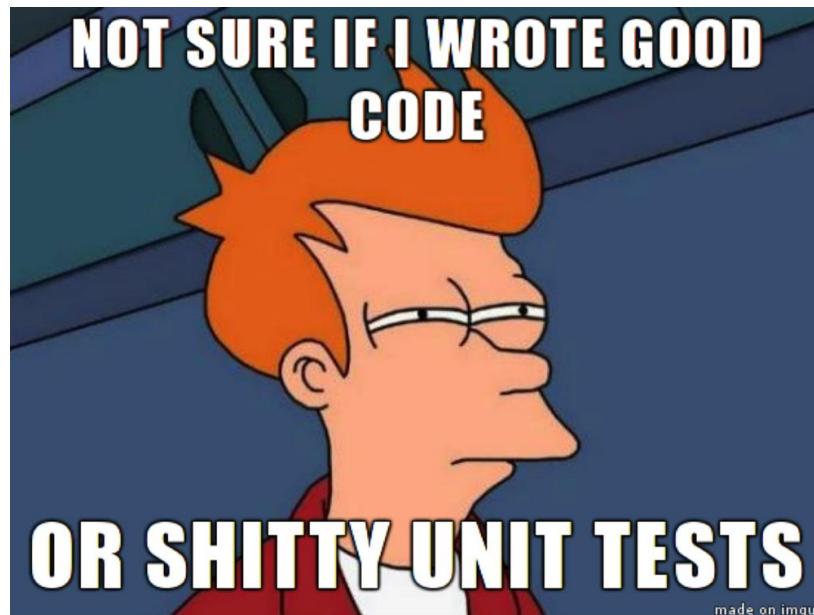
Svantaggio dei test automatici



I test automatici richiedono tempo per la loro creazione e implementazione



Incorrere nell'errore di **confondere il fine con i mezzi**



Cosa è il Unit Test



E' il test automatico che riguarda una piccola porzione di codice (unit)



La determinazione dell'unità di test spetta al programmatore (classe o parte di essa)



Una volta determinata la unit di test si creerà il mock per testarne il funzionamento

Concetto di Test-Driven Development (TDD)

