

SPRING  
DA  
ZERO

## Indice generale

INSTALLAZIONE E CONFIGURAZIONE DI SQL SERVER.....	3
INSTALLAZIONE SQL SERVER EXPRESS EDITION.....	3
INSTALLAZIONE SQL SERVER MANAGEMENT STUDIO.....	6
CONFIGURAZIONI PRELIMINARI SQL SERVER.....	7
INTRODUZIONE AL SPRING FRAMEWORK.....	15
INTRODUZIONE ALLO SPRING CONTEXT.....	17
PRIMO TEST SPRING CONTEXT.....	18
DISTINGUERE I BEAN CON IL NOME O L'ANNOTATION @PRIMARY.....	20
ANNOTATION @COMPONENTSCAN.....	23

# INSTALLAZIONE E CONFIGURAZIONE DI SQL SERVER

## INSTALLAZIONE SQL SERVER EXPRESS EDITION

Il primo passo è quello di connettersi nel sito ufficiale della microsoft, che ci permetterà di scaricare SQL Server Express edition, che è una versione light di SQL server. Per fare ciò, andiamo nel sito della Microsoft e cerchiamo il link di download di questo prodotto.



### Developer

SQL Server 2022 Developer è un'edizione completa e gratuita che può essere usata come database di sviluppo e test in ambienti non di produzione.

Scarica subito



### Express

SQL Server 2022 Express è un'edizione gratuita di SQL Server ideale per lo sviluppo e la produzione di applicazioni per desktop, Web e server di piccole dimensioni.

Scarica subito

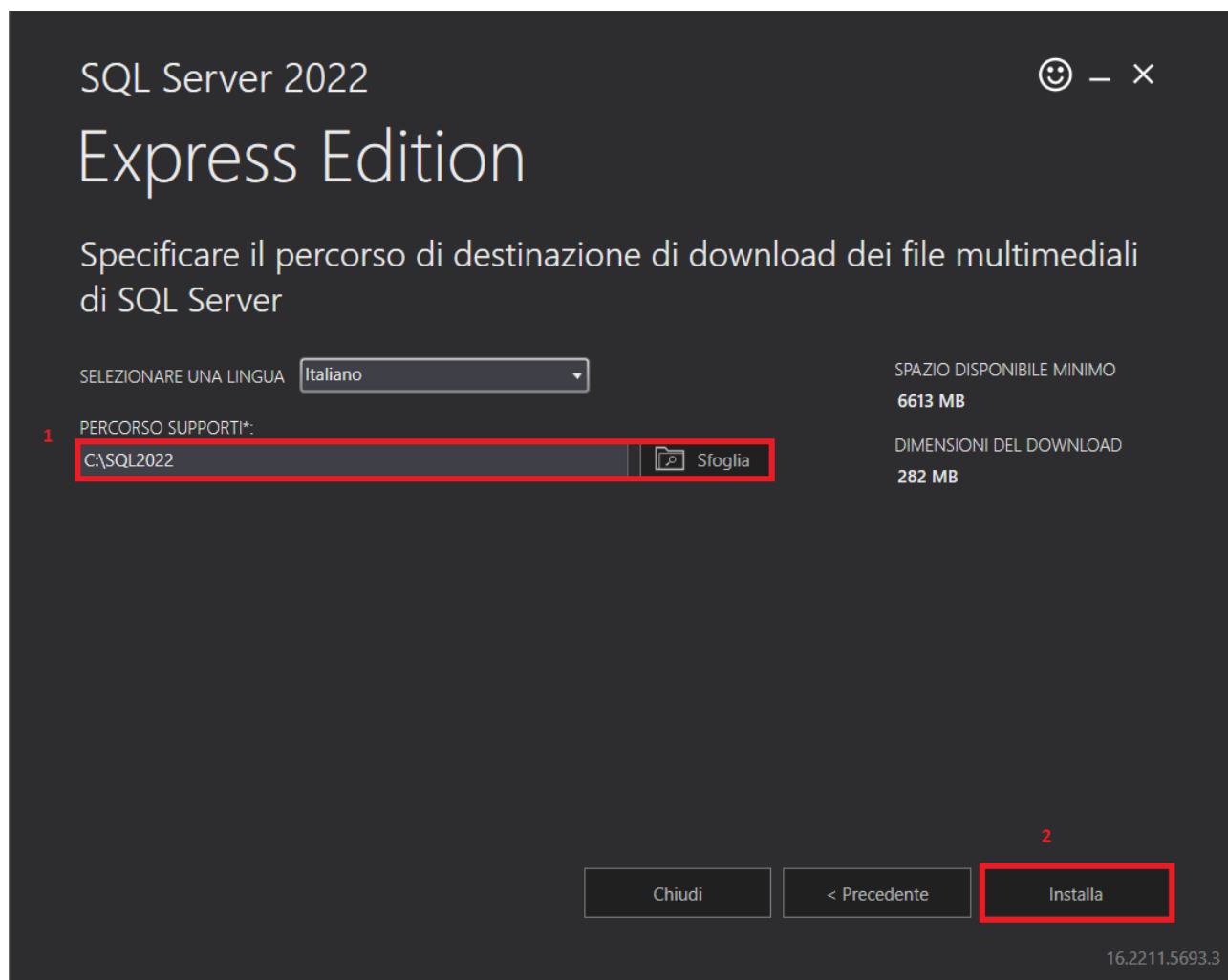
Cliccando sul bottone “Scarica subito”, si procederà con l’effettuare il download di un piccolo file eseguibile di setup.

Una volta che il download del file.exe è stato completato, possiamo cliccarci sopra due volte.

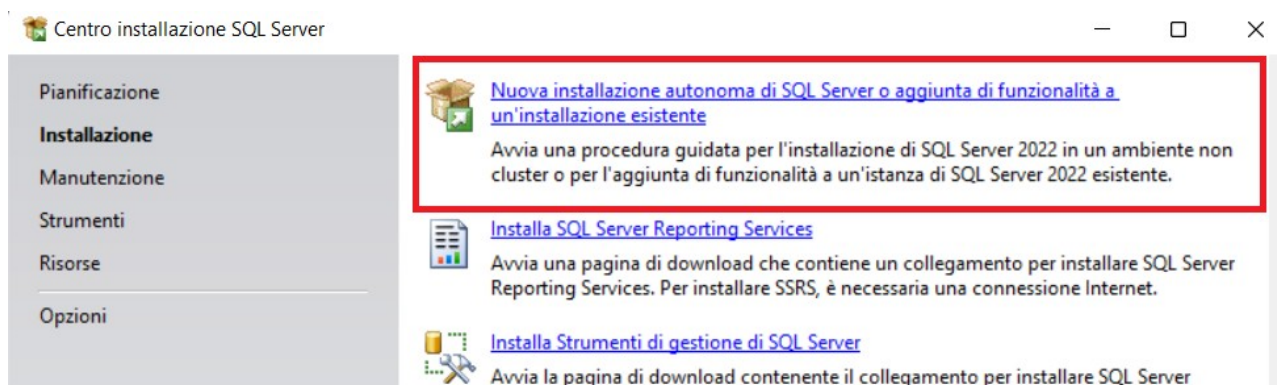
Apparirà la schermata di installazione, in cui abbiamo a disposizione 3 opzioni, ma è consigliata la scelta dell’opzione “Personalizzata”.



Il passo successivo sarà quello di decidere dove andremo ad installare SQL Server, ma possiamo anche decidere di lasciare il percorso di default. Una volta presa la decisione del percorso di installazione, possiamo procedere con il cliccare sul tasto “Installa”.



Arrivati a questo punto, uscirà il popup di installazione in cui andremo a cliccare sulla voce “Nuova installazione autonoma di SQL server ...” per permettere all’eseguibile di iniziare il processo di installazione.



Clicchiamo sempre sul tasto “Avanti” durante le varie fasi di installazione, fino ad arrivare nella fase di “Configurazione del motore di database”.

Scegliamo la modalità mista di autenticazione e inseriamo una password per l’utente “sa”

The screenshot shows the 'Configurazione del motore di database' (Database Engine Configuration) window of the SQL Server 2022 installation wizard. The window title is 'Installazione di SQL Server 2022'. The main heading is 'Configurazione del motore di database'. Below the heading, a description states: 'Specificare la modalità di sicurezza dell'autenticazione del motore di database, gli amministratori, le directory dati, TempDB, il massimo grado di parallelismo, i limiti di memoria e le impostazioni Filestream.' The left sidebar lists various installation steps, with 'Configurazione del motore di database' highlighted. The main area has several tabs: 'Configurazione server', 'Directory dati', 'TempDB', 'Memoria', 'Istanze utente', and 'FILESTREAM'. The 'Configurazione server' tab is active, showing options for authentication mode. The 'Modalità di autenticazione' section has two radio buttons: 'Modalità di autenticazione di Windows' (unselected) and 'Modalità mista (autenticazione di SQL Server e autenticazione di Windows)' (selected and highlighted with a red line). Below this, there are fields for 'Password' and 'Conferma password', both containing four dots. A section for 'Specifica amministratori di SQL Server' shows a list with 'MS\39346 (39346)' selected. To the right of this list, a note states: 'Gli amministratori di SQL Server hanno accesso illimitato al motore di database.' At the bottom of the window, there are three buttons: '< Indietro', 'Avanti >' (highlighted with a red box), and 'Annulla'.

L’utente “sa” è l’utente System Administrator, per cui è un utente che può eseguire qualsiasi operazione all’interno del nostro DBMS. Per questo motivo, questo utente deve avere una password di un certo livello di complessità.

A questo punto partirà l’operazione di installazione vera e propria, per cui non ci resterà altro che attendere il completamento.

# INSTALLAZIONE SQL SERVER MANAGEMENT STUDIO

Anche se non essenziale, si consiglia di effettuare il download e l'installazione di Microsoft SQL Server Management Studio.

Il SQL Server Management Studio ci permette di accedere al database, di visualizzare la struttura dello stesso e di poterlo modificare. Quindi questo strumento è fondamentale per poter eseguire delle operazioni di manutenzione del nostro database.

Per poter ottenere il SQL Server Management Studio bisogna connettersi al sito ufficiale della Microsoft e poi cliccare sul link "Scarica SQL Server Management Studio".

## Scaricare SSMS

Per scaricare SSMS 19 Preview 4, visitare [Download SSMS 19](#).

[Download gratuito per SQL Server Management Studio \(SSMS\) 18.12.1](#)

SSMS 18.12.1 è la versione in disponibilità generale più recente. Se è installata una versione di SSMS 18 con disponibilità generale precedente, l'installazione di SSMS 18.12.1 esegue l'aggiornamento alla versione 18.12.1.

- Numero di versione: 18.12.1
- Numero di build: 15.0.18424.0
- Data di rilascio: 21 giugno 2022

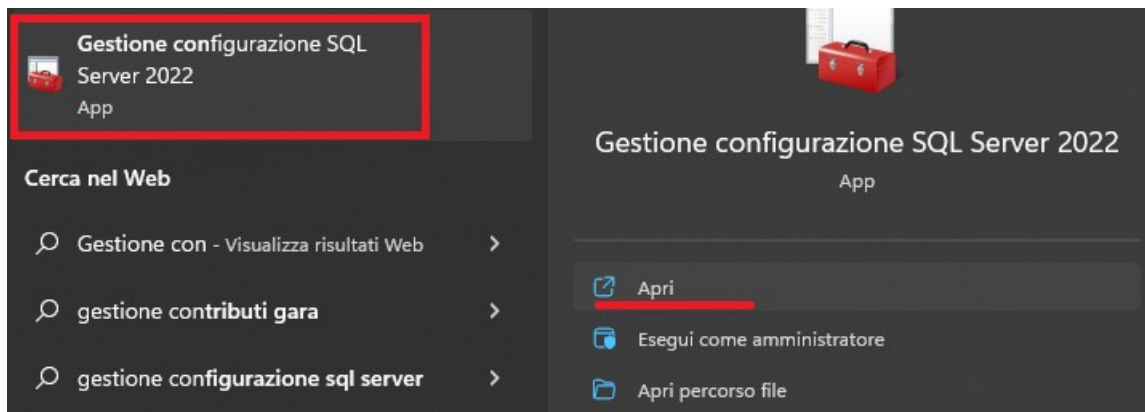
Attendere a questo punto che il browser scarichi il file che ci permetterà di installare questo software.

# CONFIGURAZIONI PRELIMINARI SQL SERVER

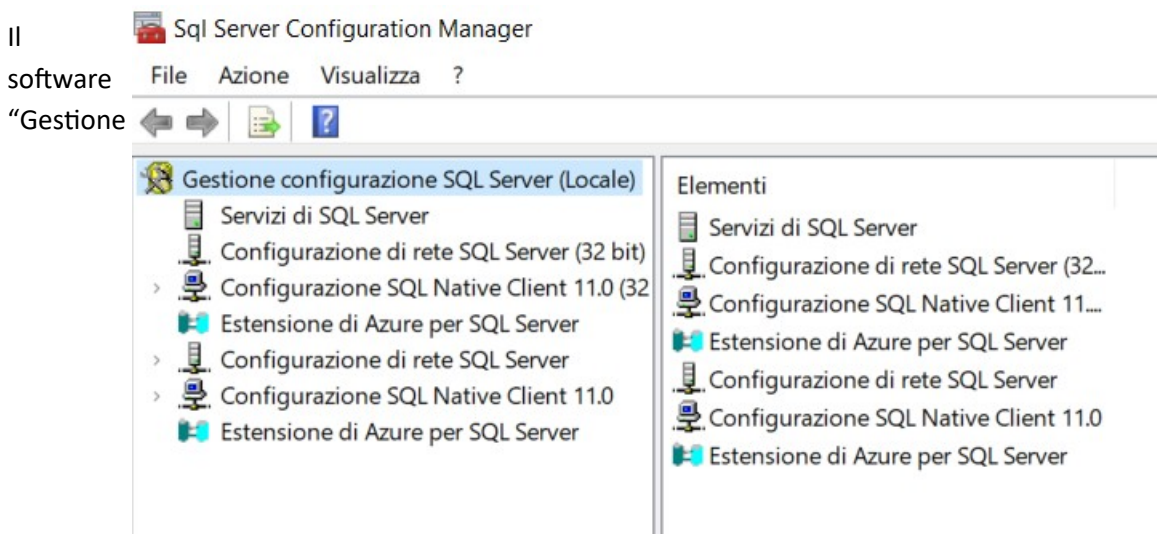
In questo paragrafo vedremo alcune operazioni che è necessario eseguire una volta che è stato installato SQL Server.

Questo DBMS infatti può presentare dei problemi per quanto riguarda le configurazioni quando lo si deve utilizzare con Spring MVC, oppure con Spring Boot.

Per prima cosa, una volta che abbiamo installato SQL Server, nella cartella d'installazione sarà presente il software "Gestione configurazione SQL Server".



Quando apriamo questo software, ci apparirà la seguente schermata:



configurazione SQL Server" ci permette di visualizzare alcuni elementi fondamentali dell'SQL Server, tra cui i servizi che si attivano quando installiamo SQL Server.

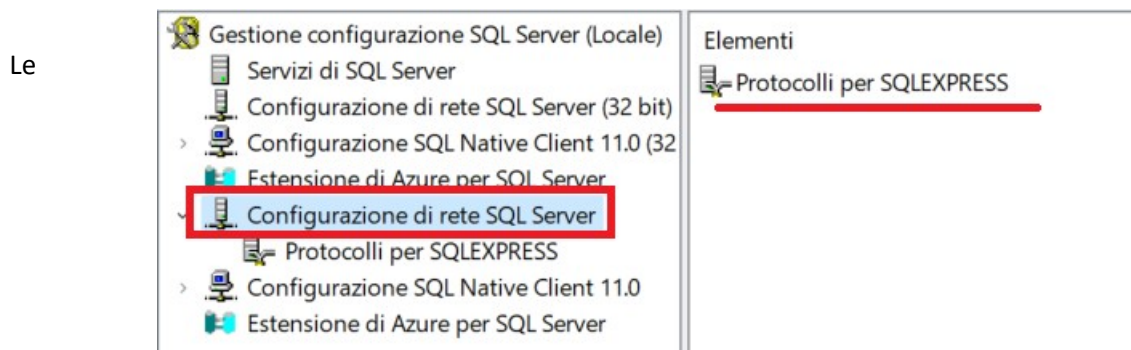
Gestione configurazione SQL Server (Locale)						
Servizi di SQL Server						
Nome	Stato	Modalità di avvio	Accedi come	ID processo	Tipo di servizio	
SQL Server Browser	Arrestato	Altro (Avvio, Sistema...	NT AUTHORITY\LOC...	0		
SQL Server (SQLE...	In esecuzione	Automatico	NT Service\MSSQLS...	8324	SQL Server	
SQL Server Agent ...	Arrestato	Altro (Avvio, Sistema...	NT AUTHORITY\NET...	0	SQL Agent	
SQL Server Launc...	In esecuzione	Automatico	NT Service\MSSQL...	16696		
SQL Full-text Filte...	In esecuzione	Manuale	NT Service\MSSQLF...	2756		

Tra i servizi più importanti di SQL Server abbiamo quello omonimo SQL Server(NOME\_ISTANZA), che rappresenta la versione del DBMS installato sulla macchina. Se sulla macchina abbiamo due o più

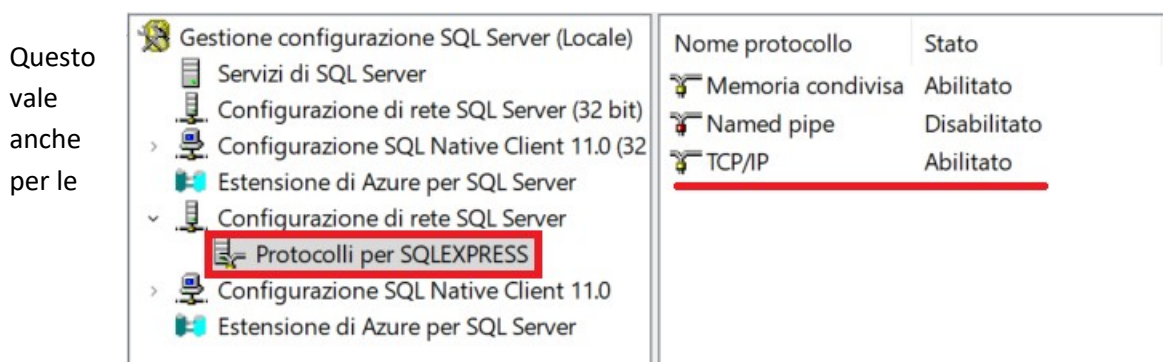


installazioni dell'SQL Server, allora avremo più istanze di tale servizio nella schermata. Se questo servizio non è presente nella schermata, oppure non è in esecuzione, allora c'è stato qualche problema in fase di installazione e dovremo ripetere tale procedura dall'inizio.

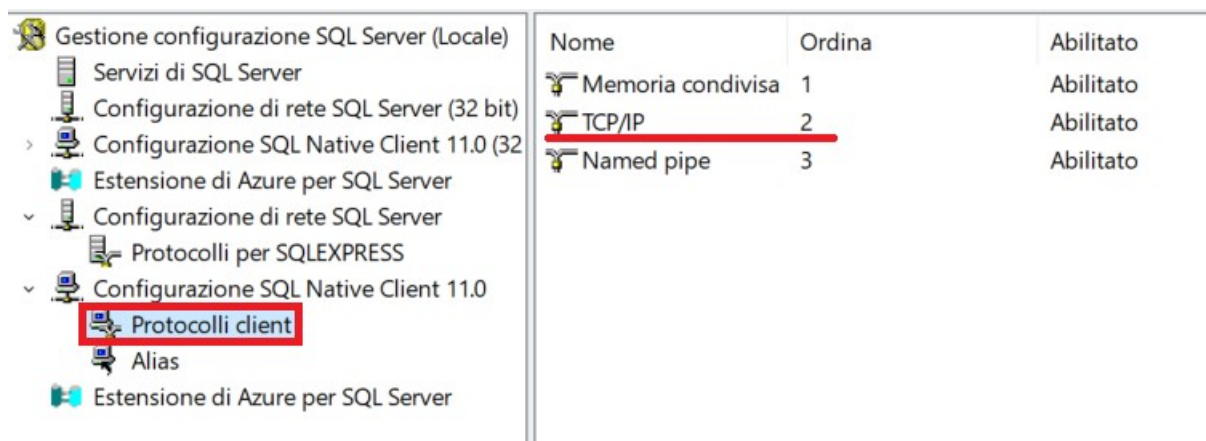
L'altro elemento fondamentale sono le configurazioni di rete di SQL Server.



configurazioni di rete di SQL Server devono avere il TCP/IP abilitato.



configurazioni SQL native.

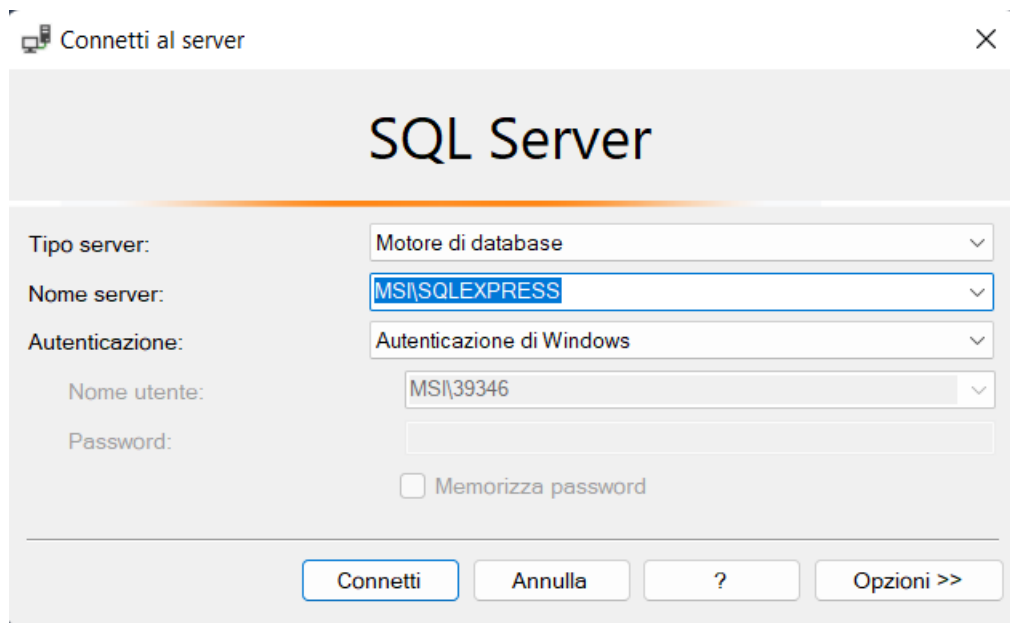


Se non fosse così, non sarebbe possibile connettersi all'SQL Server dall'esterno, cioè da dei software come nel caso di Spring.

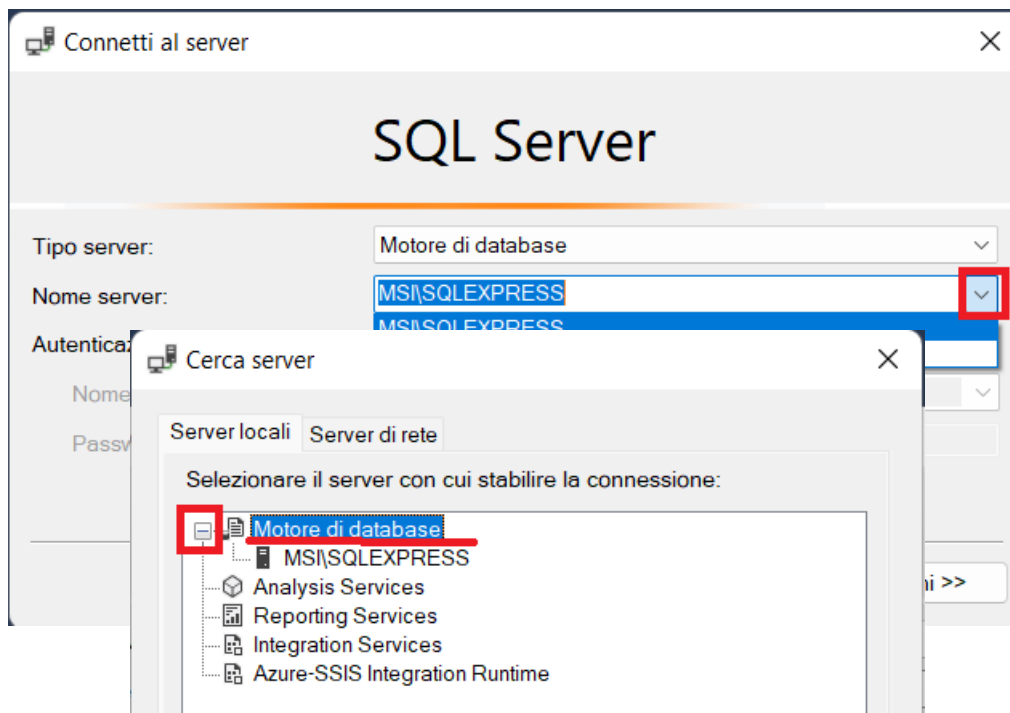
Una volta che abbiamo effettuato queste semplici verifiche, il passo successivo è quello di abilitare l'autenticazione mista di SQL Server, quindi procediamo con l'aprire il software SQL Server Management Studio.



Nella schermata di



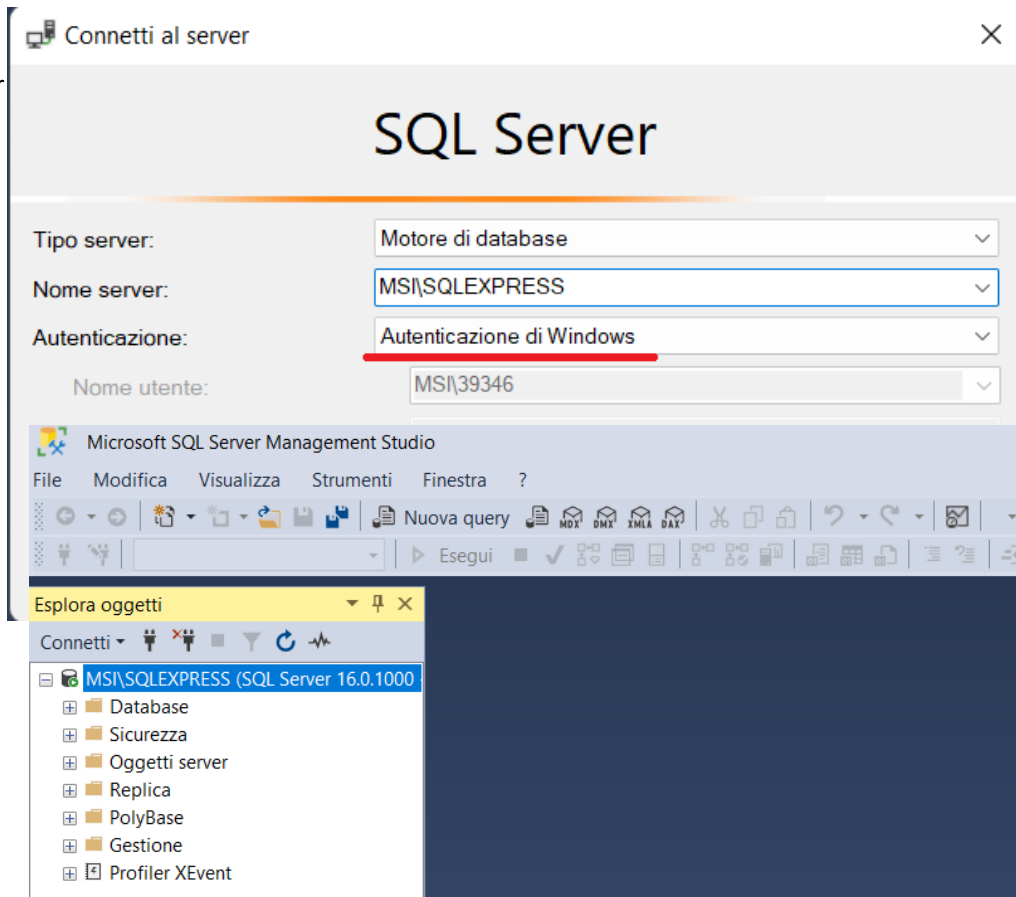
connessione all'SQL Server, il primo passo è quello di specificare il nome del server. Se questo non fosse presente, o non siamo sicuri che sia corretto, per poterlo trovare il procedimento è alquanto semplice:



Quando apriamo “Motore di database” abbiamo la possibilità di vedere l’elenco di tutte le istanze dell’SQL Server che sono presenti sulla nostra macchina. Quindi, selezioniamo il “Motore di database” che ci occorre.

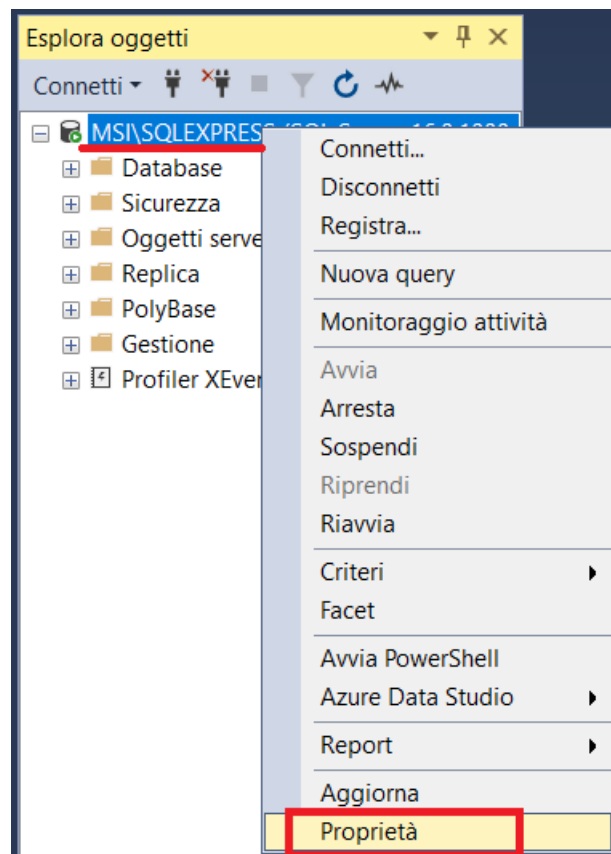
Dopo aver scelto, possiamo procedere con il premere il tasto “Connetti”

Se tutto è andato per il meglio, saremo all'interno di una schermata simile a questa:

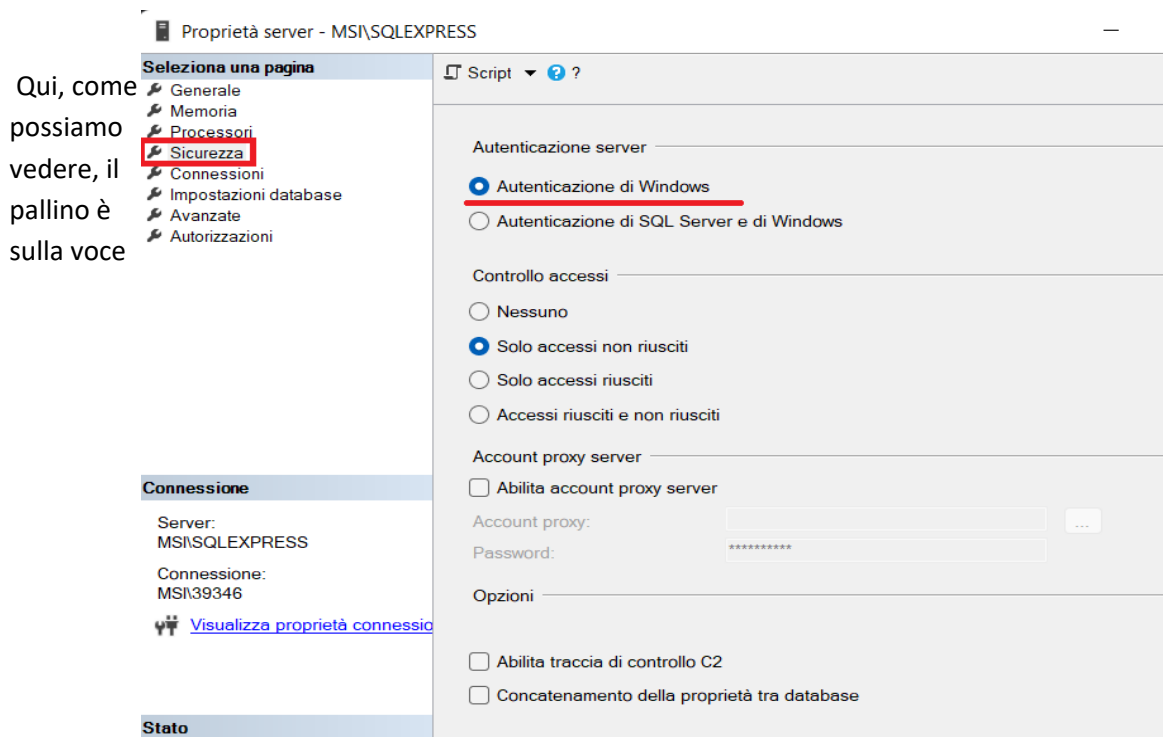


Da questa schermata possiamo abilitare l'autenticazione mista.

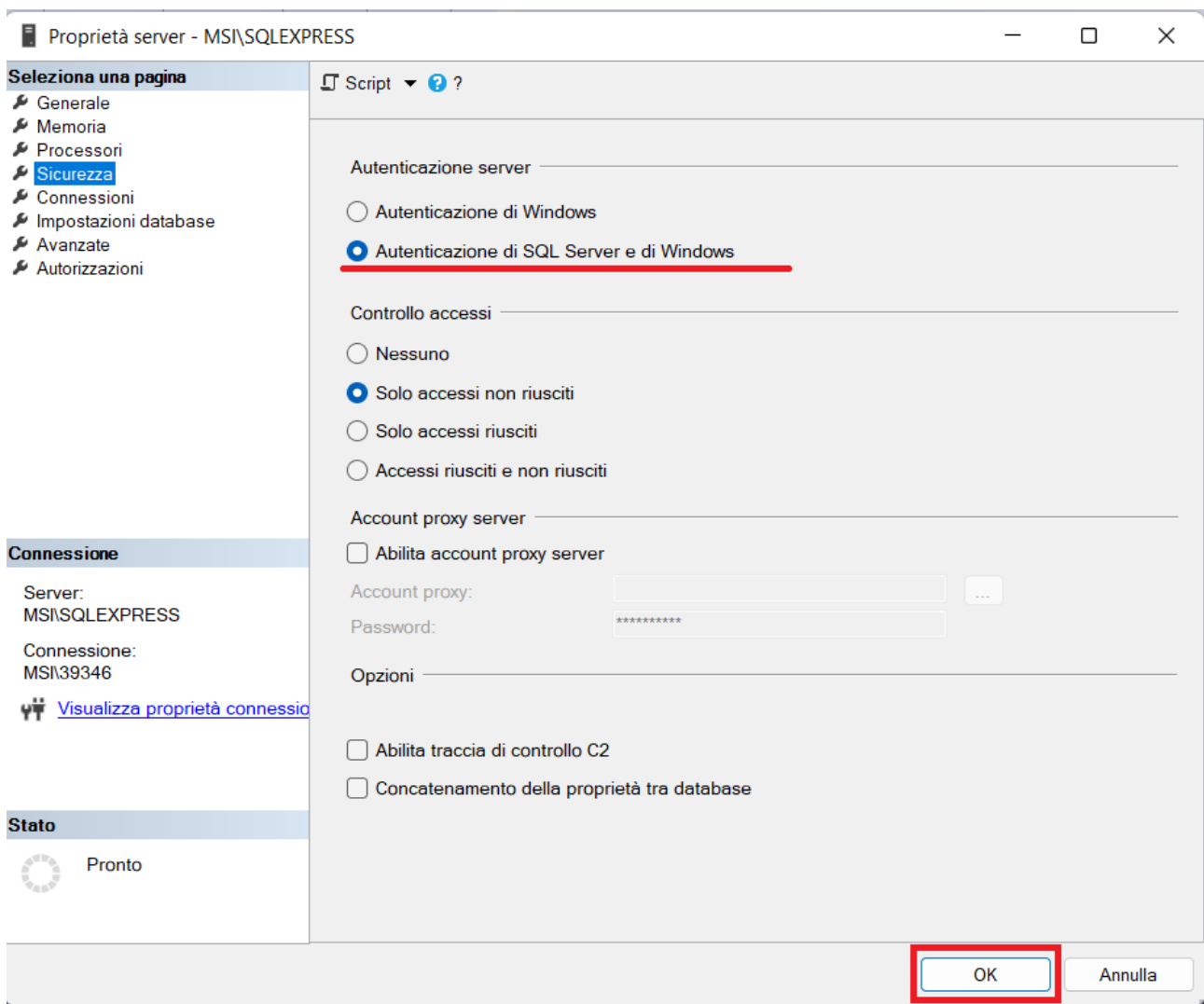
Andiamo con il tasto destro sul nome del server e clicchiamo su "Proprietà":



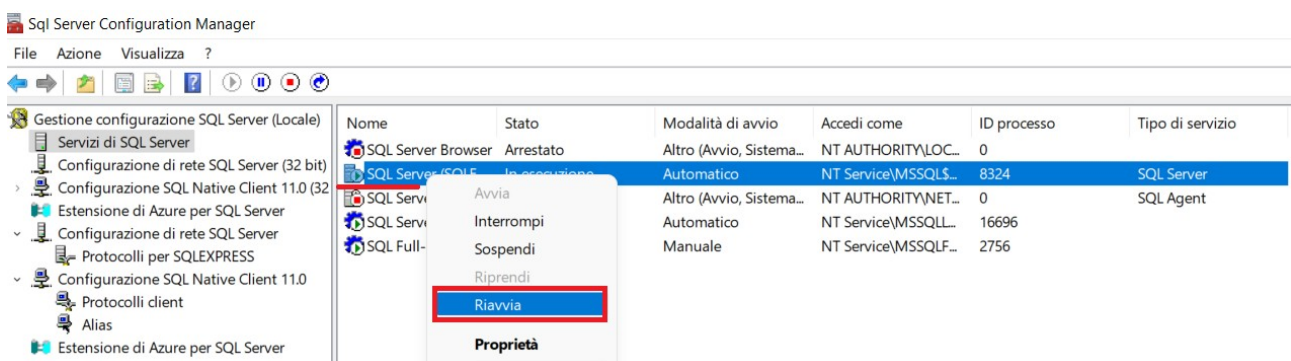
Nella nuova schermata, clicchiamo sul gruppo “Sicurezza”:



“Autenticazione di Windows”. Il nostro obiettivo invece è quello di abilitare l’autenticazione mista, quindi spostiamo il pallino sulla voce “Autenticazione di SQL Server e di Windows” e cliccare su “Ok”.

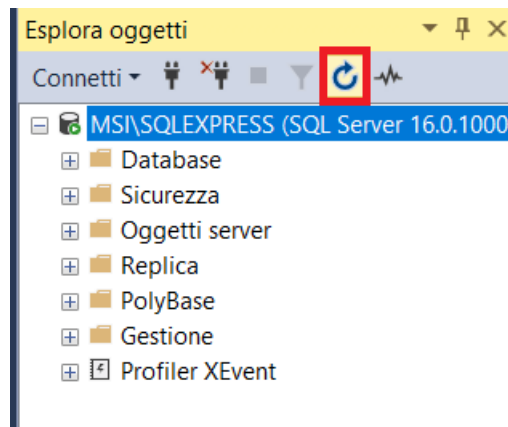


Questa modifica che abbiamo effettuato, fino a quando non riavviamo il servizio SQL Server(NOME\_ISTANZA), non verrà accettata dal DBMS. Quindi, torniamo sul software “SQL Server Configuration Manager” e, sul nome del servizio SQL Server(NOME\_ISTANZA), clicchiamo il tasto destro del mouse e sul tasto “Riavvia”.

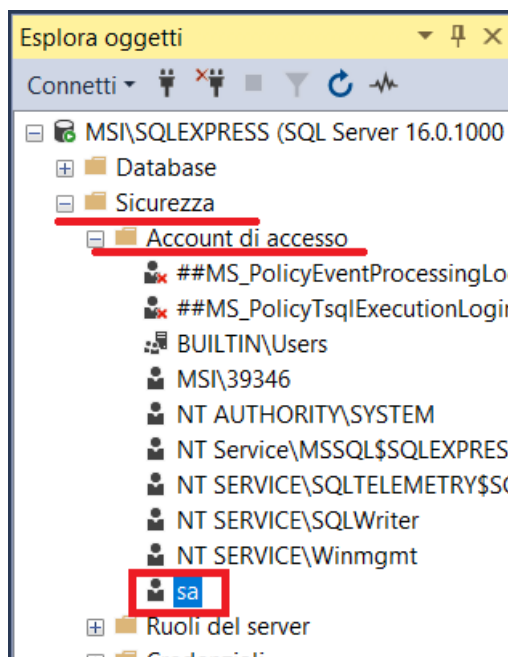


Una volta che il servizio SQL Server(NOME\_ISTANZA) è stato riavviato, possiamo procedere alla verifica della presenza dell’utente sa.

Torniamo, quindi, su SQL Server Management Studio ed effettuiamo un refresh.

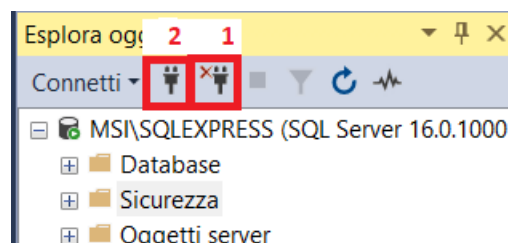


All'interno del gruppo della "Sicurezza", in "Account di accesso", scorriamo fino a trovare l'utente sa.



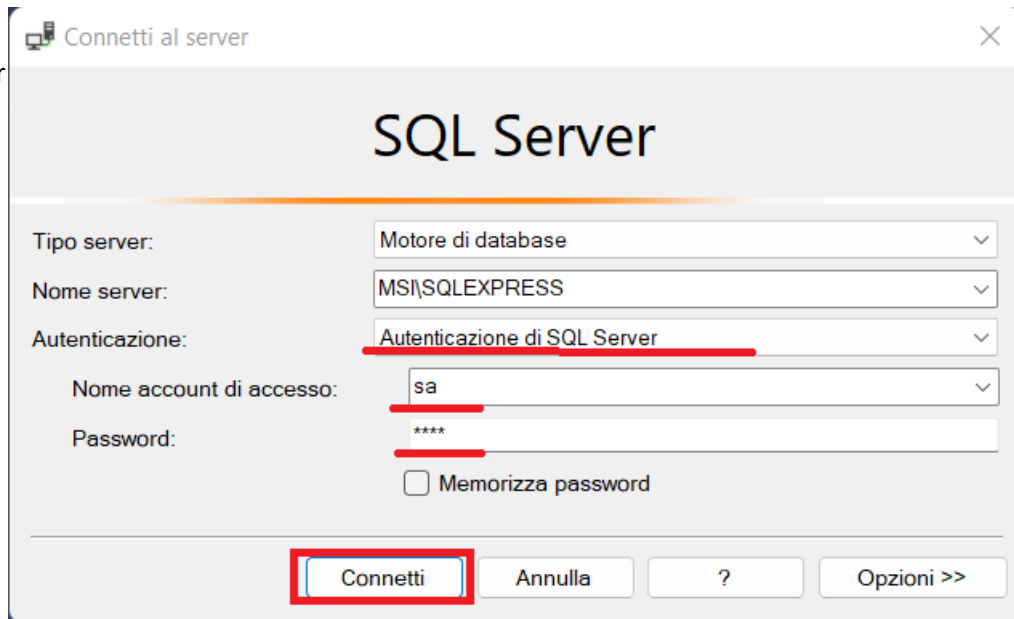
L'utente sa, come già detto, è l'utente System Administration dell'SQL Server. Ci occorrerà per eseguire le operazioni amministrative, come ad esempio il ripristino di un database.

Per entrare come utenti sa, dobbiamo prima di tutto disconnetterci dal DBMS, per poi riconnetterci subito dopo.

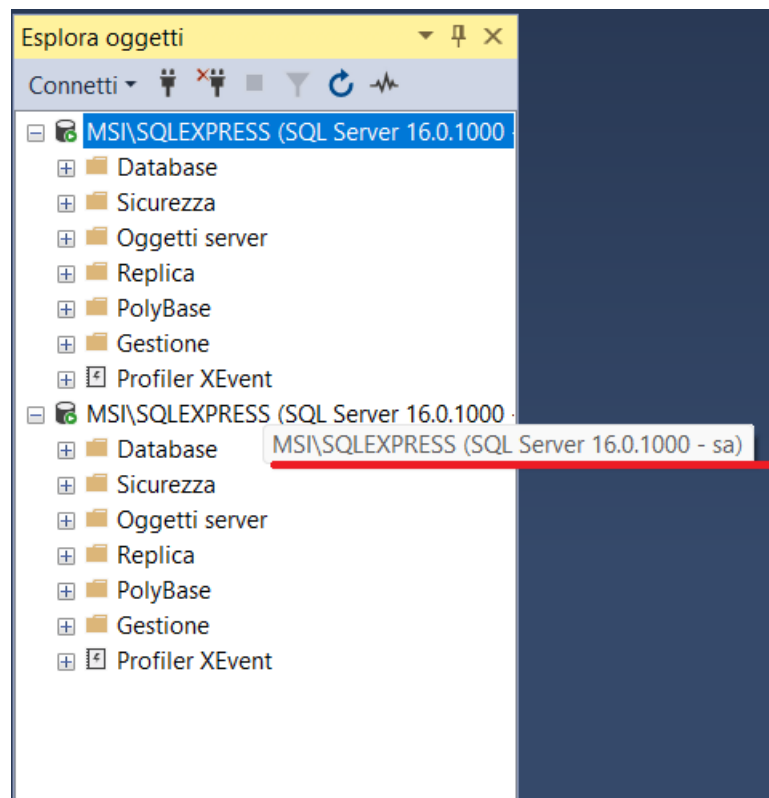


Questa volta, la schermata di connessione all'SQL Server la impostiamo nel seguente modo, inserendo la password impostata in fase di installazione dell'SQL Server:

Se tutto è andato per il meglio, dovremo



connetterci al nostro DBMS con l'utente sa.



# INTRODUZIONE AL SPRING FRAMEWORK

Lo Spring Framework è un cosiddetto application framework basato sul linguaggio JAVA.

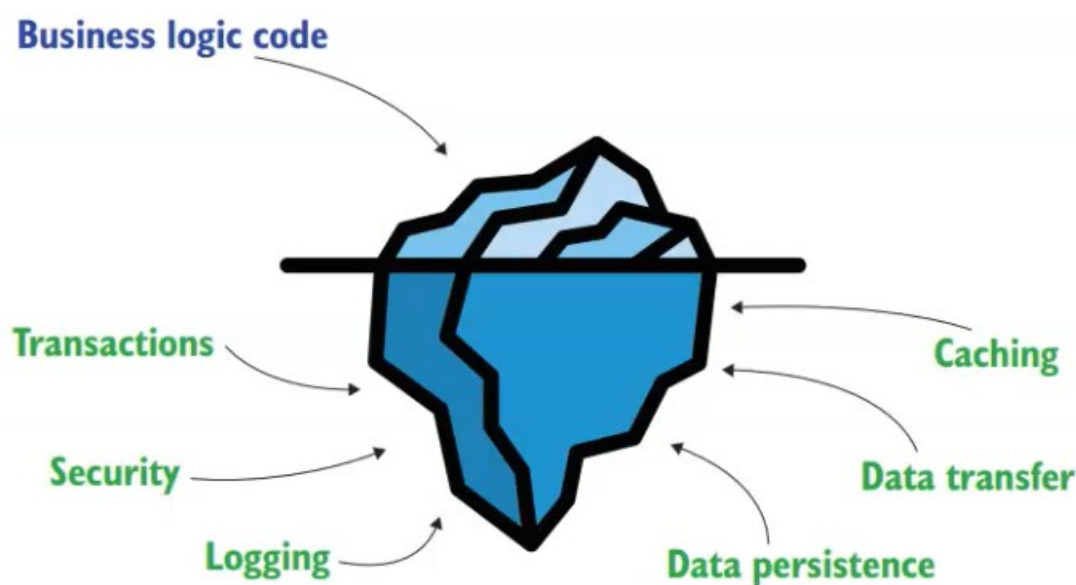
La natura delle applicazioni sviluppate tramite questo framework può essere veramente eterogenea, perché lo Spring viene utilizzato per lo sviluppo di molteplici tipologie di software, ad esempio web application, web api, applicazioni desktop, auto test e molto altro.

Gli application framework sono un insieme di funzionalità pronte all'uso, grazie alle quali possiamo creare una complessa applicazione facilmente ed in minor tempo.

I framework sono stati usati dagli sviluppatori sin dagli inizi della programmazione. Conservare parti di codice per poterli riutilizzare in altre applicazioni, modificando solo alcune parti, era già ai tempi una pratica normale.

Per essere più precisi, quando sviluppiamo una determinata applicazione, alcune parti tendono ad essere sempre le stesse, ad esempio le funzionalità che riguardano la sicurezza, il logging, la validazione, la connessione alle basi di dati, la gestione delle transazioni, il catching e così via.

Altre parti, invece, dipendono dalle esigenze di chi ha commissionato il software e rappresentano le cosiddette business logic code.



Quindi, come si vede dall'immagine, lo sviluppo è composto da una parte esterna che è rappresentata dallo business logic code e poi da tutta una serie di elementi interni che tendono ad essere sempre gli stessi.

Detto ciò, non ha senso creare ex novo quelle parti che sono sempre uguali, ma conviene farle gestire appunto da un application framework, che ci permette di andare ad utilizzare questi elementi semplicemente andando a cambiare piccole parti del codice.

Di conseguenza, il vantaggio grande dei framework è che permettono di concentrarci su elementi variabili, che poi sono quelli che interessano a chi ha commissionato il software, e di utilizzare parti che sono sempre uguali tra uno sviluppo e l'altro.



Anche se l'abbiamo definito un framework, in realtà lo Spring è a tutti gli effetti un ecosistema di framework che è composto da diversi elementi che sono stati creati per facilitare lo sviluppo di diverse tipologie di applicazioni.

Per fare alcuni esempi abbiamo:

- lo Spring Core, che è una delle fondamentali parti dello Spring, la cui caratteristica principale è quella di creare e gestire lo Spring Context, con il quale è possibile gestire le istanze dell'applicazione;
- lo Spring MVC, che permette lo sviluppo di web app che gestiscono le chiamate HTTP;
- lo Spring Data, che permette di gestire l'interazione delle web app con i database;
- lo Spring Security, che permette la gestione della sicurezza delle web app.

# INTRODUZIONE ALLO SPRING CONTEXT

Il Context dello Spring si può paragonare a tutti gli effetti ad un luogo virtuale, all'interno della memoria dell'applicazione, nel quale vengono inserite le istanze degli oggetti che verranno utilizzati dall'applicazione.

Di base lo Spring non conosce quali oggetti dovranno essere presenti nel Context, ma sarà necessario inserire gli oggetti nel Context stesso.

Le istanze degli oggetti che saranno presenti all'interno di questo contenitore (appunto il Context) vengono per convenzione chiamati beans.

Non tutti gli oggetti che vengono utilizzati dall'applicazione potranno essere gestiti da Spring e, quindi, essere inseriti all'interno del Context.

Per andare a creare un nuovo progetto utilizzando lo Spring, normalmente si utilizza il Maven.

Il Maven è un tool dell'Apache Software Foundation che facilita il build dei progetti basati su JAVA.

Non è direttamente connesso al framework Spring, ma viene utilizzato come build tool di riferimento per i progetti che utilizzano questo framework.

Il Maven può essere facilmente configurato con un determinato file in formato XML, chiamato pom.xml, grazie al quale potremo andare a:

- scaricare e validare le dipendenze del progetto;
- eseguire i test di funzionamento della nostra applicazione;
- validare la sintassi dell'applicazione;
- compilare ed eseguire il build del nostro progetto, creando i file eseguibili dello stesso, che potranno essere Jar o War.

## PRIMO TEST SPRING CONTEXT

Prima di partire con lo scrivere il codice, è necessario inserire la seguente dipendenza nel file pom.xml di Maven:

```
<dependencies>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.3.9</version>
  </dependency>
</dependencies>
```

Il nostro obiettivo è quello di creare una nuova classe, chiamata Clienti, che conterrà un solo parametro “nome”. Con questa classe dovremo fare in modo di istanziarla e inserirla all’interno del Context della nostra applicazione.

Quindi la classe Clienti sarà la seguente:

### Clienti.java

```
package main;

public class Clienti {

    private String nome;

    public String getNome() {
        return nome;
    }

    public void setName(String nome) {
        this.nome = nome;
    }
}
```

Adesso che abbiamo creato la nostra classe, dobbiamo fare in modo di istanziare la stessa ed inserirla nel nostro Context. Per fare questa creiamo una classe di configurazione, come la seguente:

### ConfigApp.java

```
package config;

import main.Clienti;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class ConfigApp {

    @Bean
    Clienti cliente1(){
        Clienti cliente = new Clienti();
        cliente.setName("Nicola");

        return cliente;
    }
}
```

Quando abbiamo a che fare con una classe di configurazione che deve funzionare all'interno della nostra applicazione Spring, la prima cosa che dobbiamo fare è inserire l'annotation `@Configuration`.

L'annotation `@Configuration` identifica la classe a cui è stata assegnata come classe di configurazione.

Abbiamo in seguito creato un metodo `clienti1()` che semplicemente istanzia la nostra classe `Clienti` e poi va a popolare la proprietà `nome`. Questo determinato metodo dovrà essere avviato in automatico dal nostro Spring in fase di avvio, per cui al di sopra di esso è stata aggiunta un'altra annotation `@Bean`. In questo modo stiamo andando a specificare che `clienti1()` sarà un metodo che dovrà essere avviato nella fase di creazione del Context.

Il Context dovrà essere creato in una classe con un metodo `main()`, per cui creiamo una terza classe:

#### Main.java

```
package main;

import config.ConfigApp;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Main {

    public static void main(String[] args) {

        AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext(ConfigApp.class);

        Clienti cliente = context.getBean(Clienti.class);
        System.out.println(cliente.getNome());

        context.close();
    }
}
```

Facendo partire l'applicazione, otteniamo questo risultato:

```
Nicola
```

Come possiamo vedere, ciò che abbiamo ottenuto dall'applicazione è il nome del nostro ipotetico cliente, che era stato istanziato nella classe di configurazione. Quindi, la nostra classe `Clienti` è entrata nel Context e dal Context abbiamo ottenuto il dato che era stato precedentemente inserito nella classe di configurazione.

## DISTINGUERE I BEAN CON IL NOME O L'ANNOTATION @PRIMARY

Nel passato paragrafo abbiamo visto un primo esempio di utilizzo dello Spring Context, dove abbiamo creato:

- 1) una semplice classe Clienti con all'interno un'unica proprietà "nome";
- 2) una classe di configurazione con un metodo classificato di tipo bean, grazie all'utilizzo dell'annotation @Bean, in cui instanziamo la nostra classe Clienti e ne popoliamo la proprietà "nome";
- 3) una classe Main in cui abbiamo creato il nostro Context, utilizzando l'AnnotationConfigApplicationContext, specificando in quale classe il nostro Context riuscirà a trovare i bean. Una volta fatto questo, abbiamo utilizzato il metodo getBean(), specificando in ingresso il tipo di bean che dovrà andare a cercare il Context e passando il risultato alla variabile cliente.

Arrivati a questo punto, la domanda che ci poniamo riguarda il cosa può succedere se, anziché esserci un solo bean di tipo Clienti, ce ne fossero 2. La risposta a questa domanda è che lo Spring non riesce a capire quali dei due bean vogliamo utilizzare, poichè stiamo utilizzando un identificativo generico che è il tipo della classe, per cui abbiamo un errore di compilazione.

Quando abbiamo una situazione di questo tipo, dobbiamo aggiungere un nome ai nostri bean. Più precisamente, il nome può essere inserito come parametro in ingresso all'annotation @Bean:

```
package config;

import main.Clienti;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class ConfigApp {
    @Bean(name = "Cliente1")
    Clienti cliente1(){
        Clienti cliente = new Clienti();
        cliente.setNome("Nicola");

        return cliente;
    }

    @Bean(name = "Cliente2")
    Clienti cliente2(){
        Clienti cliente = new Clienti();
        cliente.setNome("Marco");

        return cliente;
    }
}
```

Per poter identificare i bean anche a livello di utilizzo, nel metodo `getBean()` dobbiamo aggiungere il corrispondente identificativo:

```
package main;

import config.ConfigApp;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Main {

    public static void main(String[] args) {

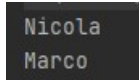
        AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext(ConfigApp.class);

        Clienti cliente1 = context.getBean("Cliente1", Clienti.class);
        System.out.println(cliente1.getNome());

        Clienti cliente2 = context.getBean("Cliente2", Clienti.class);
        System.out.println(cliente2.getNome());

        context.close();
    }
}
```

Avviando l'applicazione, questa volta riusciamo a vedere il nome di entrambi i clienti:



```
Nicola
Marco
```

Oltre ad identificare i nostri bean tramite un nome, laddove stiamo utilizzando dei bean di tipologia uguale, abbiamo anche la possibilità di specificare quale dei due è in uno stato primary, utilizzando appunto l'annotation `@Primary`:

```
package config;

import main.Clienti;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;

@Configuration
public class ConfigApp {

    @Bean
    @Primary
    Clienti cliente1(){
        Clienti cliente = new Clienti();
        cliente.setNome("Nicola");

        return cliente;
    }

    @Bean(name = "Cliente2")
    Clienti cliente2(){
        Clienti cliente = new Clienti();
        cliente.setNome("Marco");

        return cliente;
    }
}
```

```

package main;

import config.ConfigApp;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Main {

    public static void main(String[] args) {

        AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext(ConfigApp.class);

        Clienti cliente1 = context.getBean(Clienti.class);
        System.out.println(cliente1.getNome());

        Clienti cliente2 = context.getBean("Cliente2", Clienti.class);
        System.out.println(cliente2.getNome());

        context.close();
    }
}

```

Facendo in questo modo, a parità di tipi che vengono restituiti dai due bean, si prenderà come riferimento il bean che ha l'annotation `@Primary`.

Nell'esempio con l'annotation `@Primary`, l'applicazione ci risponderà allo stesso modo del caso precedente:

```

Nicola
Marco

```

Questo secondo metodo con l'annotation `@Primary` è molto utile soprattutto quando, ad esempio, stiamo creando dei bean con il quale specifichiamo le stringhe di connessione ad una base di dati. Se abbiamo più basi dati, possiamo specificare quali delle due è la connessione primaria.



## ANNOTATION @COMPONENTSCAN

Abbiamo visto come sia fondamentale che l'oggetto della nostra classe sia inserito all'interno del Context e come quest'ultimo dev'essere consapevole che c'è quel determinato oggetto.

Per inserire un oggetto all'interno del Context, abbiamo anche un altro metodo alternativo a quello precedente.

Partiamo con il modificare il codice precedente, in particolare la classe Clienti, inserendoci sopra l'annotation @Component:

```
package main;

import org.springframework.stereotype.Component;

@Component
public class Clienti {

    private String nome;

    public String getNome() {
        return nome;
    }

    public void setName(String nome) {
        this.nome = nome;
    }
}
```

L'annotation @Component identifica un componente dell'infrastruttura Spring.

Una volta che abbiamo fatto questo, andiamo sulla classe ConfigApp, eliminiamo i due bean che andavano ad istanziare la nostra classe e andiamo ad aggiungere l'annotation @ComponentScan(basePackages = "nome\_package"):

```
package config;

import main.Clienti;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;

@Configuration
@ComponentScan(basePackages = "main")
public class ConfigApp {

}
```

Tramite l'annotation @ComponentScan(basePackages = "nome\_package") stiamo andando a specificare in quale package di base lo Spring Context dovrà andare a cercare i componenti. Per essere più chiari, lo Spring Context andrà a cercare nel package specificato tutti gli elementi che sono delle classi (con l'annotation @Component) che dovranno entrare nel Context.

Possiamo ora utilizzare la nostra classe Clienti nel main, apportando le dovute modifiche:

```
package main;

import config.ConfigApp;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Main {

    public static void main(String[] args) {

        AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext(ConfigApp.class);

        Clienti cliente1 = context.getBean(Clienti.class);
        cliente1.setNome("Nicola");
        System.out.println(cliente1.getNome());
    }
}
```

Il risultato di quest'esempio sarà il seguente:

```
Nicola
```

Quindi, questa è la seconda tecnica che può essere utilizzata per inserire un oggetto all'interno del Context, rispetto a quella già vista utilizzando le annotation `@Configuration` e `@Bean`.