**1. Le Annotazioni in Java**

**Domanda 1: Cosa sono le annotazioni in Java e come vengono utilizzate?**

Le annotazioni in Java sono metadati che forniscono informazioni aggiuntive al compilatore o ad altri strumenti di sviluppo. Vengono utilizzate per modificare il comportamento del codice a livello di compilazione o runtime senza modificare il flusso di esecuzione del programma. Possono essere applicate a classi, metodi, proprietà (campi), e parametri.

**Domanda 2: Come si crea un'annotazione in Java?**

Per creare un'annotazione in Java, si utilizza il costrutto @interface, seguito dal nome dell'annotazione. Per esempio:

java

Copia codice

public @interface MyAnnotation {

String nome();

int eta();

}

**Domanda 3: Qual è la differenza tra @Retention e @Target nelle annotazioni?**

* **@Retention** definisce il periodo di vita dell'annotazione. Può essere:
  + **SOURCE**: l'annotazione è disponibile solo durante la compilazione e non nel bytecode.
  + **CLASS**: l'annotazione è mantenuta nel bytecode, ma non è disponibile durante l'esecuzione.
  + **RUNTIME**: l'annotazione è mantenuta nel bytecode ed è disponibile durante l'esecuzione, permettendo l'uso della reflection.
* **@Target** specifica dove può essere applicata l'annotazione (classe, metodo, proprietà, etc.). Alcuni esempi:
  + **TYPE**: applicata a una classe o interfaccia.
  + **FIELD**: applicata a una proprietà di una classe.
  + **METHOD**: applicata a un metodo.

**Domanda 4: Come si può ottenere un'annotazione da una classe tramite reflection?**

Si può usare la reflection per verificare se una classe ha una determinata annotazione e poi estrarne i valori. Ecco un esempio:

java

Copia codice

if (classeSaluto.isAnnotationPresent(AnnotazioneDellaClasse.class)) {

AnnotazioneDellaClasse annotazione = classeSaluto.getAnnotation(AnnotazioneDellaClasse.class);

System.out.println(annotazione.nome());

}

**Domanda 5: Come funziona la reflection in Java?**

La reflection è una funzionalità di Java che consente di ispezionare o manipolare classi, metodi e oggetti durante l'esecuzione del programma. Può essere usata per ottenere informazioni su metodi dichiarati (con getDeclaredMethods), parametri, campi, e altro, senza avere accesso diretto al codice sorgente.

**2. Il Pattern Proxy e la Reflection**

**Domanda 6: Cosa è un proxy in Java?**

Un **proxy** in Java è una classe che agisce come intermediario per un'altra classe, implementando un'interfaccia. I proxy vengono usati, per esempio, nei framework come MyBatis, per intercettare le chiamate ai metodi e aggiungere logiche extra, come la gestione delle transazioni o l'esecuzione di query SQL.

**Domanda 7: Cos'è l'InvocationHandler e come si usa con i proxy?**

L'**InvocationHandler** è un'interfaccia che definisce il metodo invoke, che viene chiamato ogni volta che viene invocato un metodo su un proxy. La sua implementazione definisce cosa fare quando un metodo del proxy viene chiamato.

Ecco un esempio:

java

Copia codice

public class MyInvocationHandler implements InvocationHandler {

private Object target;

public MyInvocationHandler(Object target) {

this.target = target;

}

@Override

public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {

// Logica aggiuntiva prima di invocare il metodo reale

System.out.println("Metodo chiamato: " + method.getName());

return method.invoke(target, args);

}

}

**Domanda 8: Come si crea un proxy in Java?**

Un proxy in Java si crea tramite la classe Proxy, specificando l'interfaccia che deve implementare e l'InvocationHandler che gestirà le chiamate ai metodi.

java

Copia codice

MyInvocationHandler handler = new MyInvocationHandler(new RealClass());

SomeInterface proxy = (SomeInterface) Proxy.newProxyInstance(

SomeInterface.class.getClassLoader(),

new Class[] { SomeInterface.class },

handler);

**3. MyBatis: Framework di Mappatura Object-Relational**

**Domanda 9: Cosa è MyBatis e come funziona?**

MyBatis è un framework di mappatura object-relational che consente di eseguire query SQL direttamente, mappando i risultati su oggetti Java. In MyBatis, le query SQL sono definite in file XML separati, che sono associati a mapper Java.

**Domanda 10: Come funziona la relazione tra Mapper XML e il suo Mapper Java in MyBatis?**

In MyBatis, ogni file XML contiene le query SQL, che sono mappate a metodi definiti nell'interfaccia Mapper Java. Il Mapper Java fornisce l'API per invocare le query SQL definite nel file XML. Esempio:

* **Mapper XML (BookMapper.xml)**:

xml

Copia codice

<mapper namespace="com.example.BookMapper">

<select id="findAllBooks" resultType="com.example.Book">

SELECT \* FROM books

</select>

</mapper>

* **Mapper Java (BookMapper.java)**:

java

Copia codice

public interface BookMapper {

List<Book> findAllBooks();

}

**Domanda 11: Come vengono eseguite le query in MyBatis?**

Le query vengono eseguite tramite il SqlSession fornito da MyBatis. Esempio di esecuzione di una query:

java

Copia codice

SqlSession session = sqlSessionFactory.openSession();

BookMapper mapper = session.getMapper(BookMapper.class);

List<Book> books = mapper.findAllBooks();

session.close();