**Lazy Loading**, **Eager Loading** und **Explicit Loading** in Entity Framework (Core)

**1. Was bedeutet Lazy Loading in Entity Framework?**

a) Alle verwandten Daten werden sofort mitgeladen.  
b) Verwandte Daten werden nur bei Zugriff aus der Datenbank geladen.  
c) Daten werden nie automatisch geladen.  
d) Daten werden nur manuell geladen.

**2. Wie wird Eager Loading typischerweise in Entity Framework aktiviert?**

a) Mit der Methode .Load()  
b) Durch die Nutzung von .Include() in der Abfrage  
c) Automatisch durch die Navigationseigenschaften  
d) Durch das Setzen einer Konfiguration in DbContext

**3. Was ist der Hauptnachteil von Lazy Loading?**

a) Es lädt zu viele Daten auf einmal.  
b) Es verursacht zusätzliche Datenbankabfragen während der Laufzeit.  
c) Es ist nicht einfach zu implementieren.  
d) Es funktioniert nur mit SQL Server.

**4. Bei welchem Lade-Verfahren werden alle benötigten Daten schon bei der ersten Abfrage geladen?**

a) Lazy Loading  
b) Explicit Loading  
c) Eager Loading  
d) Deferred Loading

**5. Wie kann man Explicit Loading in Entity Framework auslösen?**

a) Durch .Include() in der Abfrage  
b) Durch Aufruf von context.Entry(entity).Collection(...).Load() oder Reference(...).Load()  
c) Durch automatische Navigationseigenschaften  
d) Durch die Verwendung von virtual-Eigenschaften

**6. Welche Voraussetzung ist notwendig, damit Lazy Loading in EF Core funktioniert?**

a) Navigationseigenschaften müssen virtual sein.  
b) Die Datenbank muss SQL Server sein.  
c) Es darf keine Fremdschlüssel geben.  
d) Der DbContext muss manuell geöffnet werden.

**7. Was passiert, wenn man bei einer Abfrage keine .Include() verwendet und auch kein Lazy Loading aktiviert ist?**

a) Es werden keine Daten geladen.  
b) Es werden nur die Haupt-Entity-Daten geladen, keine Navigationseigenschaften.  
c) Es wird ein Fehler geworfen.  
d) Alle verwandten Daten werden automatisch geladen.

**8. Welche der folgenden Methoden entspricht Explicit Loading?**

a) context.Customers.Include(c => c.Orders).ToList()  
b) context.Entry(customer).Collection(c => c.Orders).Load()  
c) var orders = customer.Orders (bei aktivem Lazy Loading)  
d) context.Customers.ToList()

**9. Wann ist Eager Loading besonders sinnvoll?**

a) Wenn nur wenige Daten geladen werden sollen.  
b) Wenn man alle verwandten Daten schon bei der Abfrage benötigt.  
c) Wenn man Daten erst später im Programm laden möchte.  
d) Wenn die Navigationseigenschaften nicht virtual sind.

**10. Welche Eigenschaft muss man bei Navigationseigenschaften setzen, damit EF Core Lazy Loading unterstützt?**

a) static  
b) virtual  
c) readonly  
d) abstract

**11. Wie kann man in EF Core Lazy Loading explizit aktivieren?**

a) Durch Installation des Pakets Microsoft.EntityFrameworkCore.Proxies und Aktivierung in DbContextOptions  
b) Durch Hinzufügen von .Include() in der Abfrage  
c) Durch Aufruf von .Load() bei der Navigationseigenschaft  
d) Durch Verwendung von async-Methoden

**12. Was ist ein Nachteil von Eager Loading, wenn man sehr viele Navigationseigenschaften lädt?**

a) Es kann zu sehr großen und langsamen SQL-Abfragen führen.  
b) Es ist kompliziert zu programmieren.  
c) Es lädt nie die gewünschten Daten.  
d) Es funktioniert nur mit relationalen Datenbanken.

**13. Welche Ladeart würde man wählen, wenn man gezielt bei Bedarf verwandte Daten lädt und nicht bei jeder Abfrage?**

a) Lazy Loading  
b) Explicit Loading  
c) Eager Loading  
d) Automatic Loading

**14. Kann man in EF Core mehrere Navigationseigenschaften mit Eager Loading gleichzeitig laden?**

a) Nein, nur eine Navigationseigenschaft pro Abfrage  
b) Ja, mit mehreren .Include()-Aufrufen oder .ThenInclude()  
c) Ja, aber nur mit Lazy Loading  
d) Nein, das ist nicht möglich

**15. Was passiert bei Explicit Loading, wenn die Navigationseigenschaft bereits geladen ist?**

a) Es wird eine Ausnahme geworfen.  
b) Es wird eine neue Abfrage ausgeführt, die Daten werden neu geladen.  
c) Es passiert nichts, die Daten bleiben unverändert.  
d) EF Core lädt nur neue, noch nicht geladene Daten.