בס"ד

**תרגיל Service Lifetime & Dependency Injection**

תרגיל

מטרת התרגיל להדגים ולהתנסות בהבדלים באורך החיים של האובייקטים, לפי הצורה בה הם הוספו ל service collection.

הנחיות:

שלב א' – תרגיל קודם (של middleware)

שלב ב':

1. הוסיפי ל solution פרויקט נוסף מסוג class library.
2. הוסיפי ל class library את ה interfaces וה classes המופיעים בנספח
3. הוסיפי class ו interface - עם פונקציה אחת: CheckLifetime שלא מקבלת כלום ומחזירה רשימת מחרוזות. ה class יממש את ה Interface.
4. לפי שמות ה interfaces, השתמשי בפונקציה המתאימה עבור כל אחד להוספתו ל service collection. (**למנוע ספק: כמובן שהשמות נקבעו רק לצורך התרגיל אך אין משמעות לשם אלא רק לצורה בה האובייקט מתווסף ל !!!Service collection**).
5. הזריקי **למחלקה** ול Controllerאחד מכל סוג מה interfaces שיצרת.
6. בפונקציה במחלקה גשי למאפיין Guid של כל אחד מה services והוסיפי אותו לרשימה של המחרוזות.
7. גם ב action ב controller הוסיפי את ה Guid של כל אחד מה services לרשימה של מחרוזות.
8. הזריקי ל controller את הסוג שיצרת בסעיף 9, והוסיפי ב action קריאה לפונקציה CheckLifetime.
9. מה controller החזירי רשימה של תוצר הפונקציה CheckLifetime + אלו שהוספת בסעיף 13.
10. בצעי 2 קריאות get ל Action (מבלי לעצור את ההרצה בין שתי הקריאות), והשווי בין המזהים שקבלת.
11. יש להגיש עם התרגיל פלט של שתי requests.

שימי לב:

מה השתנה בין הקריאות?

מה השתנה בתוך הקריאות – עם שתי הצבעות לאובייקט?

מה היה זהה בשתי הבקשות?

פלט לדוגמה:

פלט request ראשונה:

[

"In Controller transient: c617e627-4c3b-49bb-9239-1dd81f8616d1",

"In Controller scoped: f5582240-2c41-4c72-a121-f2bfd2a77c2d",

"In Controller singleton: 475cdad9-a112-441d-ba23-47242176f7ba",

"In Service transient: ae295382-eb06-4eb6-8a94-d8c25611f337",

"In Service scoped: f5582240-2c41-4c72-a121-f2bfd2a77c2d",

"In Service singleton: 475cdad9-a112-441d-ba23-47242176f7ba"

]

פלט request שניה:

[

"In Controller transient: 6890442f-b871-4eb0-a445-d57c7c44ed15",

"In Controller scoped: caf7fce7-0e74-4a91-aa1b-0101f4ba721e",

"In Controller singleton: 475cdad9-a112-441d-ba23-47242176f7ba",

"In Service transient: ae7142c2-3774-4c46-9d7a-66c0ff2e41b9",

"In Service scoped: caf7fce7-0e74-4a91-aa1b-0101f4ba721e",

"In Service singleton: 475cdad9-a112-441d-ba23-47242176f7ba"

]

נספח לתרגיל

1. Classes ו- interfaces לצורך התרגיל

public interface IOperation

{

Guid OperationId { get; }

}

public interface IOperationTransient : IOperation

{

}

public interface IOperationScoped : IOperation

{

}

public interface IOperationSingleton : IOperation

{

}

public class Operation : IOperationTransient,

IOperationScoped,

IOperationSingleton

{

public Operation() : this(Guid.NewGuid())

{

}

public Operation(Guid id)

{

OperationId = id;

}

public Guid OperationId { get; private set; }

}

1. GUID

על מנת להבדיל בין במופעים השונים נעשה שימוש ב GUID.

GUID – Globally Unique Identifier – מזהה ייחודי עולמי, הוא מספר אקראי המשמש לזיהוי ייחודי של אובייקט בכל הקשר שהוא. קיימים 2**122** מספרים כאלו, כך שיתכן שמזהה מסוים יוגרל פעמיים, אבל הסבירות לכך נמוכה מאד. הסבירות ששני מספרים יוגרלו באותו הקשר (למשל, באותה תכנית), קטנה אף יותר.

ב C# ישנו struct מובנה ב FW שמאפשר ליצור GUID וכן פונקציונליות נוספת הקשורה ל GUID.

נציין: guid הוא struct ותיק ב C#, הוא לא קשור דווקא לweb api core וניתן להשתמש בו בכל סוגי הפרויקטים ב C#.