Connection Desing Pattern

הצעה לשיפור אלגנטי

ג'סטין-יוסף אנג'ל

ג'סטין-יוסף אנג'ל
the_j_angel@hotmail.com
http://blogs.israelblog.co.il/justin_angel/

שלום לכולם,

במהלך העבודה בכל שפת תכנות מתפתחות תבניות מסויימות שהתפתחו עם הזמן והניסיון של תכנתנים רבים. התבניות האלו מאופיינות על פי שהם עונות על צורך\בעיה מסויימת בצורה שעם הזמן הפכה להיות מקובלת על כלל התוכניתנים. השם המקצועי שלהן הוא - Design Patterns.

אחת מה-Design Patterns האלו היא Pattern Connection Design של דוט נט שהתפתחה עם הזמן וכיום מאוד מקובלת בעולם הדוט נט. אאל"ט היא שייכת לקבוצת ה-Behavioral Design Patterns. אפשרי שבין פתיחת חיבור למסד הנתונים ועד סגירתו תעלה שגיאה, מה שישאיר את החיבור פתוח עד לאתחול תהליך השרת (aspnet_wp.exe). למשל:

```
// new SqlConnection
SqlConnection sqlCon = new SqlConnection(ConnectionString);

SqlCon.Open();

// Raise Eeception:
// divide 1 by 0, Raises “Division by zero“ math exception
Response.Write ( (1/0).ToString() );

SqlCon.Close();
```

בדוגמה לעיל מה שקורה בפועל הוא שבין פתיחת החיבור למסד ועד סגירתו עלתה שגיאה ולכן החיבור לא נסגר (גם כאשר הדף נסגר). כמובן שבמקום איזה חלוקה ב-0 היה אפשר לדבר על כל שגיאה אחרת (לרוב, שגיאות מסד נתונים בהרצת שאילתות).

תכלית ה-Design Pattern הוא לתת מענה למצב זה בכך שתדאג שלא משנה אם תקפוץ שגיאה תמיד בסוף הפעולה יסגר החיבור למסד הנתונים. התבנית קוד היא:

```
SqlCon.Open();
try
{
  // Do Something...
}
finally
  {
    SqlCon.Close();
  }
```

כפי שאפשר לראות בדוגמה למעלה תמיד יסגר החיבור למסד הנתונים ברגע שנפתח. אם בשלב ה-Do Something עולה שגיאה וגם אם לא, תמיד יסגר החיבור. את התבנית מיישמים בכל מקום שבו פותחים חיבור למסד הנתונים ובכך מבטיחים שהוא תמיד יסגר.

סה"כ מדובר בתבנית שעונה באופן מושלם על הבעיה שהצגנו ובלי שום קוד עודף ושום בלאגנים.

אבל עדיין, מדובר פה על לכתוב (או בעזרת כלים אוטומטים, ללחוץ על כפתור) כל פעם מחדש את התבנית. ומה אם פעם אחת שכחנו? ונכון שזה מכוער להוסיף כל פעם מחדש את שורות הקוד האלו?

אני מציע פתרון אלטרנטיבי - **BasePage.OnError.**

בסופו של דבר כל שגיאה שתתרחש בדף תבעבע למעלה ותגרום לשגיאה ב System.Web.UI.Page. כלומר, ניתן ללכוד אותה בעזרת ה- OnError Delegate של ה-System.Web.UI.Page.

ההצעה שלי היא כזאת, לתת לכל הדפים לרשת מ-BasePage אחד שיורש כמובן מ-System.Web.UI.Page. לאותו BasePage יהיה תכונה ומשתנה פנימי מסוג SqlConnection. ל-OnError Delegate של ה-BasePage נוסיף פונקציה שבודקת אם החיבור פתוח ואם כן תסגור אותו.

```
public class BasePage : System.Web.UI.Page
{
  // BasePage Constructor
  public BasePage()
  {
   // Add BasePage_ErrorHandling to Error Delegate on BasePage
    this.Error += new EventHandler(BasePage_ErrorHandling);
  }

  private void BasePage_ErrorHandling(object sender, EventArgs e)
  {
   // Used to replace Connection Design Pattern
   // Check if Connection was initlized and open and
   // if true - close connection
   if ((sqlCon != null) && (sqlCon.State == System.Data.ConnectionState.Open))
     {
      _sqlCon.Close();
     }
  }

  // Property And internal SqlConnection used by the pages
  private System.Data.SqlClient.SqlConnection _SqlCon;
  public System.Data.SqlClient.SqlConnection SqlCon
  {
   get
   {
    return _SqlCon;
   }

   set
   {
    _SqlCon = value;
   }
  }
}
```

בפשטות: יצרנו פקד SqlConnection פנימי שישמש את כל האפליקציה. כדי לאפשר לכל הדפים גישה אליו יצרנו גם Property שמחזירה ומשנה אותו. כתבנו פונקציה שמטרתה היא לבדוק במקרה של שגיאה אם החיבור פתוח ואם כן לסגור אותו. וככה, בלי שום שורות קוד בדפים עצמם - פתרנו בצורה יותר אלגנטית את הבעיה מאשר ה-DesignPattern.

כמובן שיש כמה הערות, יתרונות ואפשרויות להרחיב את המודל הזה:

1. ניתן להוסיף פונקציה נוספת ל-BasePage.OnLoad שמטרתה היא לאתחל את החיבור ולקבוע את מחרוזת החיבור של החיבור למסד הנתונים. הערה חשובה היא שבפונקציה של OnError בודקים אם החיבור לא null היות ויכול להיות ששגיאה עלתה באפליקציה עוד לפני שאותחל החיבור. אותו.

```
public class BasePage : System.Web.UI.Page
{
  public BasePage()
  {
    ...
    this.Load+= new EventHandler(BasePage_Load);
  }

  private void BasePage_Load(object sender, EventArgs e)
  {
    // Initilize internal SqlConnection
    this._SqlCon = new SqlConnection();

   // Set ConnectionString from Web.Config AppSettings
    _SqlCon.ConnectionString = “inital catlog= Northwind; computer= justin;”;
  }
}
```

2. את מחרוזת החיבור למעלה ניתן לאחזר מנתונים מקובץ ה-Web.config. ככה אם מחליטים מחר לשנות את צורת הגישה ל-Web.config משנים רק ב-BasePage.

```
public class BasePage : System.Web.UI.Page
{
  public BasePage()
  {
    ...
    this.Load+= new EventHandler(BasePage_Load);
  }

  private void BasePage_Load(object sender, EventArgs e)
  {
    // Initilize internal SqlConnection
    this._SqlCon = new SqlConnection();

   // Set ConnectionString from Web.Config AppSettings
    _SqlCon.ConnectionString = ConfigurationSettings.AppSettings["ConnectionString"];
  }
}
```

3. אפשר להוסיף פונקציה ל-BasePage.OnUnload שמטרתה היא לבדוק אם החיבור נסגר בסוף הדף ואם לא - לדאוג לסגור אותו. בנוסף למטרות של ניטור אפליקציה אפשר ומומלץ לרשום מצבים אלו ב-Log כלשהו.

```
public class BasePage : System.Web.UI.Page
{
  public BasePage()
  {
    ...
    this.Unload+= new EventHandler(BasePage_Unload);
  }

  private void BasePage_Unload(object sender, EventArgs e)
  {
   // On Page.Unload make sure connection is closed
   // Check if Connection was initlized and open and
   // if true - close connection
   if ((sqlCon != null) && (sqlCon.State == System.Data.ConnectionState.Open))
     {
      _sqlCon.Close();
     }
  }
}
```

4. ניתן ליישם מודל זה על יותר מחיבור אחד.

5. כל העבודה מול SqlCon שקופה לחלוטין שעובדים מתוך הדפים שיורשים את BasePage. זה בדיוק אותו דבר אילו SqlCon היה SqlConnection שנמצא על הדף בלבד.

```
public class myWebForm : BasePage
{

  private void Page_Load(object sender, System.EventArgs e)
  {
    SqlCon.Open();
    // Do Something...
    // If Exception is raised connection will still be closed!
    SqlCon.Close();
  }
}
```