ASP.Net Version 1.1 – רשמים אישיים

ניר אדר

לאחר כתיבת מערכת גדולה בסביבת Net., מערכת לניהול העובדים הארעיים של הפקולטה להנדסת חשמל בטכניון בשפת ASP.Net, ברצוני לסכם במסמך קצר זה את הרשמים משפה זו.

השפה מציגה צורת גישה חדשה לעבודה בעת פיתוח אתרי אינטרנט – גישה המנסה להתקרב אל הצורה בה אנחנו כותבים תוכנית Windows Application.

החידוש מתבטא במספר תחומים:

1. **שפת התכנות** – כברירת המחדל המומלצת, שפת C# משמשת למימוש הפונקציונליות של הדף. זאת בניגוד ל-VBScript ששימוש ב-ASP3 כברירת המחדל.
2. **צורת העבודה** – כאשר אנו באים לפתח בסביבת Visual Studio, הדף נראה כטופס רגיל של חלונות, אליו אנחנו גוררים פקדים. לכל פקד אנחנו מסוגלים לקשר אירועים, ממש כמו בתוכנית Windows Application רגילה. התחביר כדי להגיב לאירועי משתמש דומה לזה של Windows Application. צורת העבודה כברירת מחדל היא שימוש ב-Grid Layout, הנותנת למשתמש למקם את הפקדים באופן חופשי, ממש כמו בתוכנית חלונאית.
3. **אוסף פקדים רחב** – השפה מכילה פקדים מובנים חזקים, כגון: כפתורים, רשימות, פקדים המאפשרים בדיקת תקינות הערכים המוכנסים אל פקדים אחרים, פקדים לצורך עבודה מול בסיסי נתונים ועוד.
4. **הפרדה בין הקוד לממשק המשתמש** – חידוש נוסף של ASP.Net הוא הפרדה בין הקוד המשמש לעיצוב ממשק המשתמש (HTML) לבין הקוד הממש את הפונקציונליות של הדף (הנכתב ב-C# או VB.Net).
5. **שמירת ViewState** – אחד השיפורים המשמעותיים של השפה הוא שמירת מצב הפקדים שבדף, כאשר מתבצעת שליחתו לשרת. כלומר: כאשר היינו מתכנתים ב-ASP3, אזי בכל פעם שהיינו רוצים לשלוח את הדף ללקוח, היינו צריכים להגדיר מהו התוכן של כל ListBox, ComboBox וכדו' שבטופס שלנו. לדוגמא, נניח שפיתחנו טופס המקבל את פרטי המשתמש. כאשר המשתמש לחץ על אישור בדקנו בצד השרת שכל הפרטים תקניים. במידה והמצב לא היה כזה, שלחנו טופס חדש אל המשתמש, בו הפקדים מכילים את אותו מידע כמו ששלח המשתמש, ובנוסף, למשל, הודעת שגיאה בצבע אדום שאומרת לו ששדה X אינו חוקי.
ה-issue של מילוי הפקדים כל פעם מחדש אצל הלקוח נחסך מאיתנו ב-Net. סביבת העבודה מנהלת באופן אוטומטי את תוכן הפקדים במקרה של מעבר הדף בין השרת ללקוח, ואיננו צריכים לדאוג לכך באופן מפורש.

צורת העבודה החדשה הופכת בעצם את מלאכת בניית האתר למלאכה של תכנות אפליקציה. לכך יש יתרונות וחסרונות:

- אנחנו מסוגלים כעת לבטא דברים יותר מורכבים בעזרת השפה. אנחנו מסוגלים להגדיר פונקציות לשימוש חוזר, לבנות פקדים חדשים שיכילו פונקציונאליות שימושית וכדו'. אפילו יותר חשוב, אנו מסוגלים להכניס מושגים מורכבים מתחום ה-OOP לתוך האפליקציות שלנו. הקוד של כל דף ממומש על ידי מחלקה המתאימה לו. ניתן כמו כן להגדיר מחלקות נוספות כרצוננו.
- מצד שני, בניית אתרי אינטרנט נחשבה תמיד לקלה יותר מתכנות – HTML איננה ממש שפת תכנות, אלא שפת סימון – אנחנו מסמנים לדפדפן כיצד לצייר את המסמך שלנו.
- ASP3 הינה שפה מאוד אינטואיטיבית, המשתלבת בתוך קוד ה-HTML, ומוסיפה פונקציונליות. עיקר ASP3 התרכז בעיקר במספר אובייקטים בודדים: Application, Session, Request, Response שתפקיד כל אחד מהם היה ברור.
  ASP.Net לעומת זאת מאפשרת לנו להשתמש בכל המחלקות הנמצאות ב-CLR.
- ASP.Net בגירסתו הנוכחית אינו מספק תמיד את הפונקציונליות הדרושה – במקרים רבים יש צורך להוסיף קוד JSCRIPT לדף כדי לממש את כל הפונקציונליות – למשל כאשר רוצים לשלב Confirm Box כאשר לוחצים, למשל, על כפתור.
- צורת העבודה כברירת מחדל היא שימוש ב-Grid Layout, הנותנת למשתמש למקם את הפקדים באופן חופשי. גישה זו יוצרת בעיות רבות – מבחינת התאמות לדפדפנים שונים ולרזולוציות מסך שונות – לרוב אתר הנבנה בצורה כזו יתאים לרזולוציה בודדת ולגודל כתב אחד בלבד.

בשורה התחתונה, לדעתי האישית ASP.Net היא עדיין אינה השפה האידיאלית למימוש אפליקציות Web, למרות כל המאמץ ש-Microsoft השקיעו בה. מצד אחד ASP.Net נותנת למתכנת המתחיל הרגשה כאילו הוא מתכנת אפליקציית Windows. מצד שני, השפה עדיין לא הגיעה לרמה כזו שהיא יוצרת אבסטרקציה מלאה מעל הHTML - כאשר אנו כותבים תוכנית מורכבת, בעיקר כזו שמשתמשת ב-Client Side Scripts, אנו מתחילים להשתמש בקוד Jscript משולב בקוד התוכנית. כמו כן אנו מוסיפים ידנית תגי HTML נוספים לקוד ש-ASP.Net מייצר, מכיוון שסביבת Visual Studio לא תומכת באופן מובנה בכל האפשרויות שהפקדים ושפת HTML מציעים לנו.
פעמים רבות מדיי במהלך מימוש הפרויקט שלי התחבטתי בשאלה: "מהי הדרך הכי טובה לבצע את הפעולה X? האם להשתמש בהפשטות ש-ASP.Net מציעה לי או לכתוב קוד HTML?".

הנסיון להצמד לעקרונות של תכנות נכון יצרו במקרים רבים עבודה רבה נוספת.

נושא נוסף הוא יצירת פקדים באופן דינמי. יצירת פקדים באופן דינמי היא פעולה שימושית ביותר – בהתאם לכמות הנתונים שיש בידינו, אנו רוצים, למשל, ליצור מספר טבלאות בדף שנשלח אל המשתמש. ב-ASP3 היה מדובר פשוט בלולאה המייצרת את קוד ה-HTML עבור הטבלאות, ושולחת אותו אל המשתמש. ב-ASP.Net, אנו מנסים להשתמש באבסטרקציות שהשפה מציעה, ולכן אנו מייצרים פקדים, ובאופן דינמי מקצים פקדים חדשים ושולחים אותם אל המשתמש. וכאן מגיעה הבעיה – סביר להניח ש-Microsoft תכננו דרך מאורגנת כדי ליצור פקדים ולנהל אותם, אולם דרך זו ממש איננה טריויאלית. יצירת הפקד נעשית בדרך פשוטה, על ידי שימוש בקריאה לשגרה LoadControl. אולם, הרגע בו הפקד מאותחל והרגע בו (אם בכלל) מקושרים האירועים של הפקד אל הפקד הם מעורפלים במידה מסויימת. ייתכן גם מצב שכבר יצרנו פקד, אולם הפקדים שבתוכו עדיין לא אותחלו. בעבודה שביצעתי על הפרויקט שלי, הבאגים הקשים ביותר היו קשורים לפקדים שנוצרו באופן דינאמי.

השימוש ב-ASP.Net יצר תוספת עבודה רצינית לפרויקט. התוצאה הסופית יצאה משביעת רצון, אולם כמות הזמן שהושקעה בה הייתה עצומה ומעבר לכל תכנון מקורי.
ASP.Net היא צעד ענקי בכיוון שלדעתי הוא העתיד בפיתוח אתרים, אולם העתיד עדיין לא כאן. אולי הגירסה הבאה של ASP.Net, שצפויה לצאת ב-2005 תהיה פריצת הדרך הבאה. ASP.Net 1.1 בהחלט לא עומד בסטנדרטים שלי לסביבה מומלצת לעבודה.
הפרויקט הבא שלי, בניית אתר חדש עבור פרויקט UnderWarrior, ממומש ב-ASP3. כרגע, לאחר שבוע וחצי של עבודה, האתר כבר כמעט מוכן, ואני מרוצה.

EOF