מרכז ההדרכה 2000
תמיכה ועדכונים
עדכון מס' 83
אפריל 2003

השוואה בין Java ל- C#/.NET

מבוא

במאמר זה מובאת השוואה מעמיקה בין Java - המערכת והשפה - לבין C# / .NET. ההשוואה מבוססת על פרק 2 בספר "***C# / .NET - מדריך מקצועי***", בהוצאת "**מרכז ההדרכה 2000**".

Java פותחה ע"י חברת Sun באמצע שנות התשעים של המאה הקודמת, כחלק מאסטרטגיה כוללת של רצון לפתח יישומים שאינם תלויים במערכת הפעלה או בחומרה מסויימות. שפה זו פותחה תוך שימת דגש על פשטות, תמיכה ביישומוני דפדפנים בפרט ובאינטרנט בכלל, ותוך הפקת לקחים מחסרונותיה של C++.

מערכת .NET. והשפה C# פותחו ע"י חברת Microsoft בתחילת האלף השלישי עקב הפסדה של זו האחרונה לחברת Sun בתביעה משפטית. Microsoft, שעד אז ניסתה להתחרות ב- Sun בפיתוח סביבות פיתוח ל- Java (J++), הואשמה על ידה כי היא פוגעת בתקן של Java ובגרימת תלות בין יישומי Java המפותחים ב- J++ לבין Windows/COM.

הן השפה החדשה C#, והן ה- CLR (Common Language Runtime) שבמערכת .NET. פותחו בהשראתה הברורה של Java (לא מעטים טוענים להעתקה ברורה של כ- 70-80 אחוזים ממנגנוני Java!). על אף זאת, יש לציין שבוצעו שינויים והוספו מרכיבים חשובים ל- C#/.NET ביחס ל-Java.

בהשוואה זו נתייחס ל- :

- מינוחים/מושגים בשתי הטכנולוגיות
- הדמיון שבין שתי הטכנולוגיות
- הבדלים בין הטכנולוגיות
- תוספות ל- C#/.NET ביחס ל- Java
- השוואה בין כיווני ההתפתחות של שתי השפות

השוואה בין המינוחים

הטבלאות הבאות כוללות הקבלה בין מונחים בעלי שם שונה אך משמעות זהה. יש לשים לב שההקבלה אינה תמיד מלאה:

Java	.Net
Java (שפה יחידה)	C# - שפה ראשית
JVM - Java Virtual Machine	CLR - Common Language Runtime
Bytecode	IL & MSIL
Swing	Windows Forms
JSP & Servlets	ASP.Net
J2EE & EJB	.Net Enterprise Services
J2ME	.Net Compact Framework
JDBC & JDO	ADO.Net

Namespace	Package
dll, Assembly	Java Archive (JAR)
Active Directory	JNDI
PInvoke, COM Interop	Java Native Interface (JNI) and Remote Method Invocation (RMI)

מרבית המלים השמורות שבשתי השפות לקוחות מ- C/C++ ולכן הן זהות.

הבדלי שמות במילים השמורות:

C#	Java
const readonly sealed	final
bool	Boolean
using	import
namespace	package

הדומה בין השפות/טכנולוגיות

- שתי הטכנולוגיות כוללות סביבת ריצה - טכנולוגית Net. כוללת את ה-CLR, כפי שטכנולוגית Java כוללת את ה-JVM.
- שתי השפות מבוססות על שפות נפוצות אחרות:
  - Java מבוססת על C/C++.
  - C# מבוססת על VB, Java, C++.
- שתי השפות והטכנולוגיות כוללות Garbage Collector.
- בשתי השפות אין שימוש במצביעים (למעט קטעים המסומנים כ-unsafe בשפת C#).
- בשתי השפות אין שימוש ב-Header Files.
- בשתי השפות יש אבחנה ברורה בין מחלקות, שעצמים מהן מוקצים על ה- Heap, לבין טיפוסים אחרים (Java primitive types / .Net **Value Types**) המוקצים על המחסנית.
- שתי השפות מתייחסות למערכים פשוטים כאל עצמים לכל דבר.
- בשתי השפות אין Multiple Inheritance (הורשה מרובה).
- בשתי השפות קיים רעיון הממשק (Interface), ויש יכולת להורשה מרובה של ממשקים.
- בשתי השפות יש אפשרות להגדיר טיפוס בתוך טיפוס (Nested Types / Inner Class).
- בשתי השפות אין משתנים גלובליים או פונקציות גלובליות, הכל חייב להיכתב במסגרת טיפוס.
- בשתי השפות אופרטור נקודה ".", משמש לגישה לחברי מחלקה, במקום האופרטורים "->" ו "::" שבשפת C++.
- בשתי השפות null היא מילה שמורה (בניגוד ל-C++).
- בשתי השפות קיים טיפוס בוליאני (bool ב-C#, boolean ב-Java)

- שתי השפות הן Type Safe.
- בשתי השפות משתנים לא מקומיים מאותחלים ל-null, 0, false (בהתאם לטיפוס) לפני השימוש בהם.
- שתי השפות מושתתות על שימוש נרחב ב-Exceptions (try, throw, catch, finally).
- בשתי השפות נקודת הכניסה (Entry Point) היא הפונקציה הסטטית Main / main של מחלקה כלשהי (ולא פונקציה גלובלית יחידה, כמקובל ב C/C++), המקבלת מערך פרמטרים לתכנית.
- ביסוס הטכנולוגיה על ספרייה עשירה - טכנולוגית Net. כוללת את Net Class Library., כפי שטכנולוגית Java כוללת ספריית מחלקות עשירה. שתי הספריות מונחות עצמים ומחולקות למרחבי שמות / Packages. שתי הספריות כוללות:
  - מודל שיקוף חזק (Reflection Model)
  - תמיכה מובנית וכוללת ב-Threading
  - ספריית אוספים (Collections)
  - ספריית ממשקי משתמש
  - תמיכה במסדי נתונים
  - תמיכה בתקשורת (TCP/IP, פרוטוקולי אינטרנט)
  - תמיכה ב- XML

ההבדלים שבין השפות

- ריבוי שפות - טכנולוגיית Java מבוססת על שפת פיתוח אחת בלבד – Java, בעוד במערכת Net. ניתן לפתח במספר שפות (בסביבת MS-Windows) כגון: C++, VB ולשלב ביניהן.
- Properties - #C מיסדה את השימוש הנפוץ ב- Preperties בממשקי COM על ידי הצגת חבר מחלקה מסוג חדש – Property (תכונה). זוהי מעין מתודת getX ו/או setX, הנראית למשתמש כשדה ולא כמתודה. הרעיון מוכר היטב למשתמשי VisualBasic ו Delphi:

	Java	C#

הגדרה	`private int _size;`
	
	`public int getSize()`
	`{`
	`return _size;`
	`}`
	`public void setSize(int s)`
	`{`
	`_size = s;`
	`}`
		`private int _size;`
		`public int Size`
		`{`
		`get`
		`{`
		`return _size;`
		`}`
		`set`
		`{`
		`_size = value;`
		`}`
		`}`

שימוש	`obj.setSize(this.getSize()+1);`	`obj.Size++;`

- Indexers - מאפשרים להגדיר גישה לאיברים באובייקט. ניתן לראות Indexer-ים כמעין הרחבה של ה-Property לקבלת פרמטרים, אנשי C++ יראו בה תחליף להעמסת האופרטור [ ]:

	Java	C#

הגדרה	`class JavaLibrary`
	`{`
	`private String[] _books = new String[9];`
	`public String getBook(int i)`
	`{`
		`class CSLibrary`
		`{`
		`private string[] _books = new string[9];`
		`public string this[int i]`
		`{`

	return _books[i];
	}
	public void setBook(int i, String book)
	{
	_books[i]=book;
	}
	}
		get
		{
		return _books[i];
		}
		set
		{
		_books[i]=value;
		}
		}
		}

שימוש	this.setBook(2,this.getBook(1));	this[2] = this[1];

- מודל אירועים: Delegates & Events - בעוד מודל האירועים של Java עושה שימוש נרחב בממשקים פשוטים, או בטיפוסים Observer וObservable, #C מאפשרת שימוש ב -Delegates וב Events.
- foreach - לעומת המערך הפשוט ב- Java המיוצג כעצם מיוחד היורש מ- Object, בעוד שבשפת #C גם המערך הפשוט יורש מהטיפוס Collection. כמו כן נוספה בשפת #C המילה השמורה foreach המאפשרת מעבר פשוט ונוח על איברי Collection.
- Value Types - ב- Java עצמים מוקצים אך ורק על ה- Heap. ב- #C קיימת משפחה נרחבת של טיפוסים המנוהלים על המחסנית ולא על ה-Heap, ומועברים by value ולא by reference, בדומה לטיפוסים פרימיטיביים.
- Operator Overloading - בדומה ל++C, ובניגוד ל-Java, #C מאפשרת לבצע operator overloading.
- פולימורפיזם - בעוד ב-Java כל המתודות הן ווירטואליות בברירת מחדל, בשפת #C יש להצהיר על כך (בדומה ל++C) בעת ההצהרה על המתודה, על ידי המילה השמורה virtual.
  - בעת דריסת מתודה יש להשתמש בשפת #C במילה השמורה override.
  - בעת הסתרת מתודה יש להשתמש בשפת #C במילה השמורה new.
- switch - אופן השימוש במשפטי switch שופר ב-#C בהשוואה ל-++C ול- Java:
  - ב-#C יש אפשרות לשים גם ערך מטיפוס מחרוזת - בנוסף לטיפוסים האינטגרליים - בביטוי התנאי שבמשפט ה-switch.
  - ב-#C הקומפיילר אינו מספק את תכונת ה- Falling-Through בברירת מחדל (כלומר, מעבר רציף בין ה-case-ים השונים), ולכן בד"כ אין חשש מהיעדר לא מכוון של הוראת break.
- Packages, Namespaces & Assemblies - בעוד ב-Java קיים קשר הדוק בין החלוקה ל-Packages בקוד לבין האופן בו המחלקות ימוקמו בעת התקנתן אצל הלקוח, במערכת Net. קיימת אבחנה בין השתיים:

  namespace – הוא המקבילה למרחב השמות המוגדר ב-Java על ידי Package.

  assembly – היא יחידת ההפצה הבסיסית ביותר.

  הבדל עקרוני זה גורר אחריו גם הבדלים משמעותיים בין רמות הבקרה השונות: ברירת המחדל של Java בהגדרת שדה היא package, בעוד שב- #C ברירת המחדל היא internal, וזו מתייחסת ל- assembly.

תוספות ב- C#/.NET

- ניהול גרסאות - מערכת Net. מכילה אלמנטים רבים לניהול גרסאות וביניהן יכולת להריץ מספר גרסאות שונות של אותו assembly באותו היישום, הכללת מספר הגירסה כחלק אינטגרלי מה-assembly ועוד.
- מאפייני פרמטרים - בשפת #C ניתן לאפיין פרמטרים כך:

  ref – להעברת משתנים by reference, מחייב השמת ערך למשתנה לפני העברתו למתודה.

  out - מחייב השמת ערך למשתנה במימוש המתודה.
- params - שפת #C מאפשרת הגדרת מתודות שמקבלות מספר לא ידוע מראש של פרמטרים, בדומה

ל-"..." שבשפת C/C++ (למשל, בשימוש בפונקציות scanf/printf), מה שלא קיים ב Java.

- Attributes - #C מאפשרת להגדיר באופן גמיש מידע שיצורף לכל אלמנט תוכנה לצורך תיאורו
- unsafe - #C מאפשרת עבודה עם pointer-ים, בדומה ל++C. זאת, רק בתחומים מוגדרים מראש על ידי המילה השמורה unsafe. יכולת זו לא קיימת ב Java.
- Rectangular Arrays - #C מאפשרת להגדיר הן Jagged Arrays (כמו Java) והן Rectangular Arrays.

כיווני ההתפתחות של שתי השפות

Java הוצגה לעולם ע"י חברת Sun והושרשה כטכנולוגית פיתוח על ידי קואליציה של חברות, בעוד C#/.Net פותחה על ידי ענקית התוכנה - חברת Microsoft. להבדל מהותי זה יהיו מספר השלכות, לטוב ולרע:

- סביר להניח שלאורך זמן Java תהיה יותר נאמנה לסיסמה "Write once run anywhere", ותאפשר ניידות בין מערכות הפעלה, בעוד C#/.Net תהיה קשורה יותר למערכות הפעלה מבוססות Microsoft ותציע פתרונות מהירים יותר שיתאמו באופן אופטימלי למערכת ההפעלה Windows. דוגמאות:
  - קיים קשר הדוק בין Net Enterprise Services. לבין COM+. קשר שלא קיים בין Java 2 Enterprise Edition לרכיב ספציפי של מערכת הפעלה מסוימת.
  - קיים קשר הדוק בין Net Windows Forms. לבין מערכת החלונות של Windows, מה שלא קיים ב Swing של Java, הכוללת ממשק גרפי בלתי תלוי במערכת ההפעלה.
- סביר להניח שקצב חילופי הגרסאות של רכיבי Net. יהיה יותר מהיר, בשל אי התלות של הטכנולוגיה במספר גדול של גופים עסקיים.
- סביר להניח שלאורך תקופה ארוכה מאוד כלי הפיתוח העיקרי ל-Net. יהיה של חברת Microsoft, בעוד שב -Java יהיה קיים מבחר של כלי פיתוח מתחרים של יצרנים שונים.
- בשל היתרון של חברת Microsoft בנתחי השוק שלה, סביר שלמערכת Net. יהיה יתרון מהותי בפיתוח מערכות Client מבוססות Windows.

כל הזכויות שמורות © מרכז ההדרכה 2000