

# Část I

# Základy ASP.NET

Chcete-li začít psát kód nějakého webu ASP.NET, musíte nejprve zvládnout jistou malou sadu základních dovedností. V této části probereme .NET Framework, který podporuje každá aplikace .NET (kapitola 1), designérský nástroj Visual Studio, jenž pomáhá budovat a testovat weby (kapitola 2), a infrastrukturu ASP.NET, díky níž weby pracují (kapitoly 3, 4, 5 a 6).

Přestože se mohou tato témata jevit profesionálnímu vývojáři ASP.NET jako až příliš základní přehled, obsahují několik kriticky důležitých a citlivých míst. Každý seriózní vývojář ASP.NET se potřebuje dobře vyznat v podrobnostech takových záležitostí, jako jsou životní cyklus webové stránky a webové aplikace, zpracování fronty požadavku ASP.NET (ASP.NET request processing pipeline), správa stavu (state management) a konfigurační model ASP.NET. Jejich pochopení je nejenom klíčové pro vytváření webových aplikací s vysokým výkonem, ale patří též mezi nezbytné dovednosti, chcete-li rozšiřovat infrastrukturu ASP.NET, což je téma, které budeme v kapitolách této části probírat průběžně.



# KAPITOLA 1

## Úvod do ASP.NET

Když u Microsoftu vytvářeli .NET, nesnili pouze o budoucnosti – na srdci jim také ležely různé neduhy a omezení současné generace technologií webového vývoje. Než začneme pracovat s ASP.NET 3.5, neuškodí, když se trochu poohlédneme zpět, a budeme se chvilku těmto problémům věnovat. Pak lépe pochopíte, jaká řešení .NET nabízí.

V této kapitole probereme historickou cestu, kterou urazil webový vývoj, a jež nás dovede až k ASP.NET. Prolétneme tryskem okolo nejvýznamnějších rysů .NET a zběžně se seznámíme s klíčovými změnami, které přicházejí s ASP.NET 3.5. Jestliže s ASP.NET začínáte, tato kapitola vám pomůže, abyste se rychle dostali do obrazu. Jste-li příležitostným vývojářem v .NET, máte dvě možnosti. Tou první je, že si kapitolu přečtete, abyste bryskně získali přehled o tom, jak to s ním vypadá dnes. Druhá možnost je taková, že rovnou přejdete k oddílu "ASP.NET 3.5 – příběh pokračuje", abyste se seznámili s tím, co má ve svém arzenálu ASP.NET 3.5.

## Evoluce webového vývoje

První přenos dat přes HTTP (Hypertext Transfer Protocol) uskutečnil Tim Berners-Lee už před více než deseti lety. Od té doby prodělalo HTTP exponenciální růst popularity, prolomilo hranice malé skupiny vizionářů počítačové vědy, a proniklo jak do soukromého sektoru, tak i do byznysu. Dnes je to téměř slovo patřící do běžného hovorového jazyka.

Když bylo HTTP zřízeno poprvé, bylo pro vývojáře nelehkou výzvou navrhovat aplikace, které se uměly vyhledat a vzájemně spolu komunikovat. Aby vývojáři mohli úspěšně čelit takové výzvě, byly pro ně vytvořeny různé standardy, jako HTML (Hypertext Markup Language) nebo XML (Extensible Markup Language). HTML definovalo prostý jazyk, s jehož pomocí je možné popsat, jak zobrazit komplikované dokumenty na prakticky jakékoliv počítačové platformě. XML zase vytvořilo sadu pravidel pro vytváření formátů dat neutrálních vzhledem k platformám, s jejichž pomocí si pak mohou rozličné aplikace vyměňovat informace. Tyto standardy garantovaly, že web mohl od té doby využívat kdokoli, odkudkoli, a s jakýmkoli typem počítačového systému.

Souběžně s tím výrobci softwaru čelili svým vlastním výzvám. Potřebovali vyvinout nejenom takové jazyky a programovací nástroje, které by uměly komunikovat s webem, ale také celé pracovní rámce, jež by vývojářům umožnily navrhovat nejenom architekturu aplikací, ale také je vyvíjet a rozšiřovat – a to, pokud možno,

co nejsnadnějším způsobem. Přední výrobci softwaru, IBM, Sun Microsystems či Microsoft, spěchali, aby uspokojili vzniklou potřebu hromadou nových produktů.

ASP.NET 3.5 je nejnovější kapitolou ve stále probíhajících závodech ve zbrojení. Společnost Microsoft s technologií .NET vytvořila integrovanou skupinu komponent, která kombinuje budování částí pro web (značkovací jazyky a HTTP) s osvědčenou metodologií orientovanou na objekty.

## Vývojový svět před příchodem ASP.NET

Webové aplikace první generace se obtížně programovaly i udržovaly, a také čelily signifikantním výzvám co do výkonu a škálovatelnosti (scalability). Obecně se dá říci, že rané technologie webového vývoje spadají do dvou základních kategorií:

- **Oddělené, malinkaté aplikace, které se vykonávají voláním na straně serveru.** Dobrým příkladem jsou rané implementace CGI (Common Gateway Interface). Klíčovým problémem tohoto vývojového modelu je, že konzumuje obrovské množství serverových zdrojů, protože každý požadavek vyžaduje oddělenou instanci aplikace. Důsledkem je, že aplikace jsou mnohem méně škálovatelné (scalable) v prostředích s velkým počtem simultánních uživatelů.
- **Skripty, které interpretuje nějaký zdroj na serveru.** Do této kategorie patří klasické ASP (Active Server Pages) a rané implementace ColdFusion. Na těchto platformách vytváříte soubory obsahující kód HTML a vložený kód skriptu. Soubor skriptu prozkoumá při běhu parser, který střídá realizaci obyčejného HTML a vykonávání vloženého kódu. Tento proces je mnohem méně efektivní, než když se vykonává zkompilovaný kód.

ASP.NET znamená mnohem víc než prostou evoluci jednoho či druhého typu aplikace. ASP.NET není nějaká sada nemotorných háků, které umožňují na serveru spouštět aplikace nebo vykonávat komponenty. Je to zcela jinak. Webové aplikace ASP.NET jsou plnohodnotné aplikace .NET, které vykonávají kompilovaný kód a jež udržují runtime .NET. ASP.NET také používá všechny schopnosti .NET Frameworku – což je vyčerpávající soubor nástrojů tříd – právě tak snadno jako obyčejná aplikace Windows. ASP.NET v podstatě odstraňuje dělicí čáru mezi vývojem aplikací a webovým vývojem, protože rozšiřuje nástroje a technologie, které výhradně používali vývojáři desktopových aplikací, do světa webového vývoje.

## Co je špatného na klasickém ASP?

Jestliže jste doposud programovali s klasickým ASP, možná se divíte, proč Microsoft změnil v ASP.NET úplně všechno. Naučit se úplně nový pracovní rámec, to rozhodně není dětská slavnost, zvláště tehdy, když .NET přichází s hromadou nových pojmů a nabízí některé části, jejichž zvládnutí může být dosti zapeklitou záležitostí.

Všeobecně se dá říci, že klasické ASP je solidním nástrojem pro vývoj webových aplikací s využitím technologií Microsoftu. Ovšem, podobně jako je tomu u většiny vývojových modelů, ASP sice některé problémy řeší, nicméně také přináší několik nových problémů. V následujících částech si tyto problémy stručně popíšeme.

## Kód, co se táhne jako špagety

Pokud jste někdy vytvářeli aplikace s ASP, pravděpodobně jste se setkali s dlouhými stránkami, které obsahují směs skriptu realizovaného na straně serveru a HTML. Podívejte se na ukázkou. Kód v ní uvedený naplňuje rozevírací seznam (ovládací HTML prvek) výsledky získanými databázovým dotazem:

```

<%
    Set dbConn = Server.CreateObject("ADODB.Connection")
    Set rs = Server.CreateObject("ADODB.Recordset")
    dbConn.Provider = "sqloledb"
    dbConn.Open "Server=SERVER_NAME; Database=Pubs; Trusted_Connection=yes"
%>
<select name="cboAuthors">
    <%
        rs.Open "SELECT * FROM Authors", dbConn, 3, 3
        Do While Not rs.EOF
    %>
    <option value="<%=rs("au_id")%>">
        <%=rs("au_lname") & ", " & rs("au_fname")%>
    </option>
    <%
        rs.MoveNext
        Loop
    %>
</select>

```

V této ukázce je zapotřebí celých 19 řádků kódu na to, aby se vygeneroval jediný ovládací HTML prvek. To není zrovna působivé. Horší ale je, že takový styl psaní kódu má negativní dopad na výkon aplikace, protože se dohromady míchá HTML kód a skript. Až bude tuhle stránku zpracovávat ISAPI (Internet Server Application Programming Interface) ASP, což je rozšíření, které běží na webovém serveru, bude se muset skriptovací engine několikrát zapínat a vypínat, ačkoliv se jedná o zpracování jednoho jediného požadavku. Tím se zvyšuje doba nutná na zpracování celé stránky a její odeslání klientovi.

Dále, webové stránky psané tímto stylem mohou snadno a rychle nabobtnat do nezvladatelných délek. A přijdete-li do celé skládačky ještě vaše vlastní komponenty COM, které budete potřebovat na zprovoznění funkcionalit, které ASP poskytnout neumí, stane se noční můra ohledně údržby takových stránek ještě tíživější. Závěr je jasný. Bez ohledu na to, jaký přístup zvolíte, kód ASP vykazuje tendenci, že se postupně stane odpudivým, nadměrně dlouhým, a neuvěřitelně obtížně laditelným – pokud ovšem dokážete nějaké ladění ASP ve vašem prostředí vůbec zprovoznit.

V ASP.NET takové problémy neexistují. Webové stránky se píšou s ohledem na tradiční objektově orientované pojmy, a obsahují takové ovládací prvky, že s nimi pracujete velmi obdobným způsobem jako v případě desktopových aplikací. To znamená, že nemusíte dělat směs z HTML značek a inline kódu. Pokud přijmete při vytváření stránek ASP.NET přístup, kdy se používá kód v pozadí (code-behind), bude kód opravdu oddělen od prezentace, což zjednoduší údržbu kódu a umožní oddělit design webové stránky od obtížné práce spojené s vytvořením kódu pro webové stránky.

## Skriptovací jazyky

V době, kdy bylo vytvořeno, vypadalo ASP jako perfektní řešení pro desktopové vývojáře, kteří se přesouvali do světa webu. Místo toho, aby se museli učit nějaký úplně nový jazyk, nebo novou metodologii, jim ASP umožnilo, aby použili jim důvěrně známé jazyky, jako VBScript, na programovací platformě založené na serveru. Protože se už tehdy jako podkladová kostra používal populární programovací model COM (Component Object Model), skriptovací jazyky fungovaly i jako pohodlný dopravní prostředek pro přístup ke

komponentám a prostředkům serveru. Ale i když bylo ASP snadno pochopitelné pro vývojáře, kteří dovedně ovládali skriptovací jazyky, jako je VBScript, tahle důvěrná známost nebyla získána zadarmo. Protože ASP bylo založeno na starších technologiích, které byly původně navrženy pro použití u klientů, nemohly stejně dobře fungovat v novém prostředí webového vývoje.

Výkon nebyl jediným problémem. Každý objekt nebo proměnná, které byly použity v klasickém skriptu ASP, se vytvářely jako datový typ `variant`. Většina programátorů Visual Basicu dobře ví, že datové typy `variant` jsou vágně typované. Požadují větší množství paměti a jsou známé (a kontrolují se) až při běhu, důsledkem čehož je nižší výkon než u striktně typovaných proměnných. Kompilátor Visual Basicu a vývojové nástroje navíc nemohly tyto datové typy identifikovat už v návrhovém režimu. To způsobilo, že bylo téměř nemožné vytvořit opravdu integrované vývojové prostředí (IDE, integrated development environment), které by poskytlo programátorům ASP cokoliv podobného, jako jsou třeba vyspělé ladicí prostředky, IntelliSense, nebo kontroly chyb, což jsou věci běžné ve Visual Basicu a Visual C++. Bez ladicích nástrojů na odpovídající úrovni se programátoři ASP dostávali do těžkých stresů, když měli napravovat chyby, které vznikaly v jejich skriptech.

ASP.NET všechny takové problémy obchází velkým obloukem. Pro začátek stačí říct, že stránky a webové služby ASP.NET se vykonávají uvnitř CLR (common language runtime), takže mohou být napsány v jakémkoliv jazyku, pro který existuje kompilátor fungující v souladu s požadavky CLR. Už nejste omezeni pouze na VBScript nebo JavaScript – můžete používat i moderní objektově orientované jazyky, jako jsou C# nebo Visual Basic.

Je také důležité připomenout, že stránky ASP.NET se neinterpretují, ale kompilují do tzv. assembly, což je termín .NET pro jakoukoliv jednotku zkompilovaného kódu. Jedná se o jeden z nejvýznamnějších zobecňujících příspěvků do webového vývojového modelu Microsoftu. I když třeba vytvoříte svůj kód C# nebo Visual Basicu v Poznámkovém bloku a zkopírujete ho přímo do nějakého virtuálního adresáře na webovém serveru, aplikace se dynamicky zkompile, jakmile k ní přistoupí nějaký klient, a její kopie se uloží do cache pro potřeby budoucích požadavků. Jestliže se po ukončení kompilačního procesu kterýkoliv ze souborů změní, aplikace se automaticky překompile, jakmile ji bude nějaký klient požadovat.

## ASP.NET

Vývojáři Microsoftu popsali ASP.NET jako svou šanci "odeslat příkaz pro zformátování systému", a začít se zcela novým a modernějším vývojovým modelem. Tradiční pojmy týkající se vytváření webových aplikací však ve světě .NET stále platí. Každá webová aplikace se skládá z webových stránek. Můžete zpracovávat bohatě vybavený HTML, používat JavaScript, vytvářet komponenty, do nichž zapouzdříte programovací logiku, nebo přizpůsobovat a vyladovat své aplikace za pomoci různých konfiguračních voleb. V pozadí však ASP.NET pracuje odlišně než tradiční skriptovací technologie, mezi které patří klasické ASP nebo PHP.

ASP.NET se v porovnání s dřívějšími platformami webového vývoje odlišuje v těchto věcech:

- ASP.NET nabízí úplný, objektově orientovaný programovací model, který obsahuje architekturu řízenou událostmi, založenou na ovládacích prvcích, což podporuje zapouzdřování kódu a jeho opětovné využívání.
- ASP.NET dává možnost psát kód v kterémkoliv z podporovaných jazyků .NET (mezi ně patří Visual Basic, C#, J# a mnoho dalších jazyků, které mají kompilátory od jiných výrobců).
- V ASP.NET je vše podřízeno vysokému výkonu. Stránky ASP.NET a komponenty se kompilují na požádání, a tudíž se neinterpretují pokaždé, když se použijí. ASP.NET také obsahuje vyladěný model pro přístup k datům a flexibilní ukládání dat do cache, aby bylo možné ještě více zvyšovat výkon.

Tohle je pouze několik ze základních rysů, mezi které dále patří rozšířená správa stavu (state management), praktické vázání dat, dynamická tvorba grafiky nebo robustní model bezpečnosti. S jednotlivými zdokonaleními se podrobně seznámíte v knize a zjistíte, jak do celého obrázku zapadá ASP.NET 3.5.

## Sedm důležitých faktů o ASP.NET

Jestliže s ASP.NET začínáte (nebo si prostě chcete osvěžit nějaké základy), budou pro vás následující oddíly zajímavé. Uvádí se v nich sedm nejdůležitějších faktů o .NET.

### Fakt 1: ASP.NET je integrováno s .NET Frameworkem

.NET Framework je rozčleněn do pečlivě propracované kolekce funkčních částí, zahrnující více než 10 000 typů (termín .NET pro třídy, struktury, rozhraní a další klíčové programovací ingredience). Než se můžete pustit do programování jakéhokoliv druhu aplikace .NET, je potřeba, abyste měli základní informace o těchto částech – a abyste chápali, proč jsou věci uspořádány tak, jak uspořádány jsou.

Obrovská kolekce funkcionality, kterou poskytuje .NET Framework, je zorganizována tak, že klasičtí programátoři Windows to budou považovat za šťastný krok vpřed. Každá z těch tisíců tříd v .NET Frameworku je seskupena do logického, hierarchického kontejneru, kterému se říká jmenný prostor (namespace). Různé jmenné prostory poskytují různé funkce. Když se to vezme všechno dohromady, jmenné prostory .NET nabízejí funkcionalitu téměř jakéhokoliv aspektu distribuovaného vývoje, od front zpráv (message queuing) až po otázky bezpečnosti. Této obrovské skupině nástrojů se říká knihovna tříd (class library).

Je zajímavé, že třídy .NET Frameworku používáte v ASP.NET stejně, jako v jakémkoliv jiném druhu aplikace .NET (mezi které patří samostatné aplikace Windows, služby Windows, utility pro příkazový řádek atd.). Jinak řečeno – .NET dává webovým vývojářům stejné nástroje, jaké poskytuje vývojářům bohatě vybavených klientských aplikací.

---

**TIP**      *Jedním z nejlepších zdrojů, v němž se dozvíte o všech nových zákoutích .NET Frameworku, je jeho referenční příručka knihovny tříd (.NET Framework class library reference), která je součástí referenční knihovny MSDN Help. Máte-li nainstalované Visual Studio 2008, dostanete se do knihovny MSDN Help tak, že zvolíte Start -> Programy -> Microsoft Visual Studio 2008 -> Microsoft Visual Studio 2008 Documentation (přesný text prvků menu závisí na vaší konkrétní verzi Visual Studia). Jakmile jednou nápovědu načtete, referenční informace o třídách naleznete uspořádané podle jmenných prostorů pod uzlem .NET Development -> .NET Framework SDK -> Class Library Reference.*

---

### Fakt 2: ASP.NET se neinterpretuje, ale kompiluje

Jedním z hlavních důvodů degradace výkonu ve skriptech ASP je to, že v kódu webových stránek s ASP se používají interpretované skriptovací jazyky. To znamená, že když se aplikace vykonává, musí skriptovací hostitel na serveru interpretovat kód a přeložit jej do strojového kódu nižší úrovně, pěkně řádek po řádku. Tento proces je – jak každý ví – velmi pomalý.

---

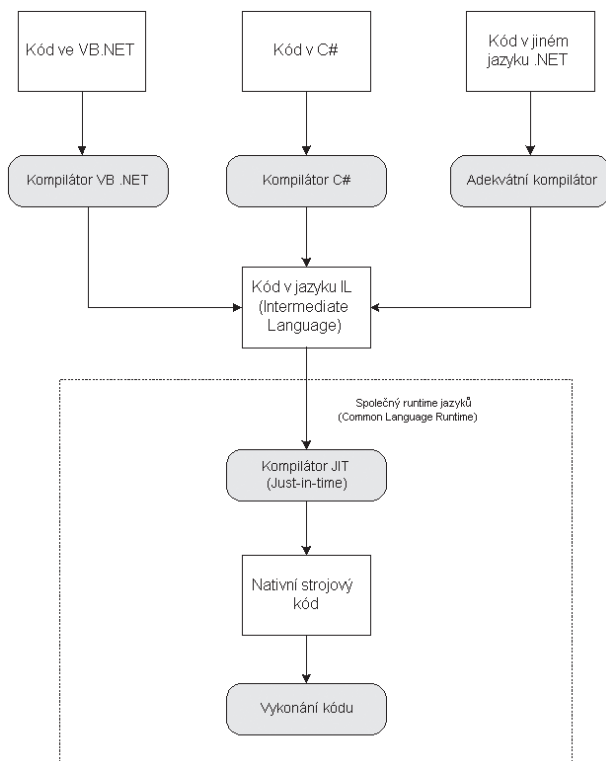
**POZNÁMKA**      *V tomto případě je reputace o něco horší než skutečnost. Interpretovaný kód je pomalejší než zkompileovaný kód, ale rozdíl ve výkonu není až tak výrazný, abyste s ASP nemohli vytvářet profesionální weby.*

---

Aplikace ASP.NET se kompilují vždy – a skutečně, je nemožné spouštět kód C# nebo Visual Basicu, aniž by se předtím nejprve nezkompiloval.

Aplikace ASP.NET procházejí dvěma kompilačními etapami. V první etapě se kód C#, který jste napsali, zkompiluje do přechodného jazyka, jenž se nazývá Microsoft Intermediate Language (MSIL), nebo prostě jen IL. Tento první krok je fundamentální příčinou toho, že v .NET je možné používat různé programovací jazyky. Všechny jazyky .NET (včetně C#, Visual Basicu, a mnoha dalších) se totiž zkompilují do virtuálně identického kódu IL. První kompilační krok může nastat automaticky, když se stránka poprvé požaduje, nebo se může vykonat předem (to je proces, kterému se říká předběžná kompilace, precompiling). Zkompilované- mu souboru s kódem IL se říká assembly.

Druhá úroveň kompilace nastává těsně předtím, než se stránka skutečně vykoná. V tomto okamžiku se kód IL zkompiluje do nativního nízkourovňového strojového kódu. Tato etapa se nazývá kompilace just-in-time (JIT) a probíhá stejně pro všechny aplikace .NET (včetně například aplikací Windows). Oba kroky kompilačního procesu vidíte na obrázku 1-1.



**Obrázek 1-1.** Kompilace webové stránky ASP.NET.

Kompilace .NET je rozdělena do dvou kroků proto, aby se vývojářům mohlo nabídnout co největší pohodlí a co nejlepší přenositelnost. Předtím, než může kompilátor vytvořit nízkourovňový strojový kód, potřebuje znát typ operačního systému a hardwarovou platformu, kde bude aplikace běžet (například 32bitový nebo 64bitový operační systém Windows). Když jsou obě kompilační etapy dokončeny, můžete vytvořit zkompilovanou assembly s kódem .NET, kterou je možné distribuovat na více než jednu platformu.



Kompilace JIT by samozřejmě nebyla tak užitečná, kdyby se musela provádět pokaždé, když nějaký uživatel požádá o zobrazení nějaké webové stránky. Aplikace ASP.NET se naštěstí nemusejí kompilovat při každém požadavku na webovou stránku. Kód IL se vytvoří pouze jednou, a pak už se generuje jen tehdy, když dojde k modifikaci zdroje. Obdobně se uchovávají i kopie souborů s nativním strojovým kódem, a sice v systémovém adresáři, jehož cesta je obvykle `c:\Windows\Microsoft.NET\Framework\v2.0.50727\TemporaryASP.NET Files`.

---

**POZNÁMKA** *Možná se divíte, proč se dočasné soubory ASP.NET nacházejí v adresáři s číslem verze 2.0, a nikoliv s číslem verze 3.5. ASP.NET 3.5 v zásadě používá engine ASP.NET 2.0 (s několika novými funkcemi navíc). O tomto designu se další informace dozvíte později v této kapitole, v sekci "ASP.NET 3.5 – příběh pokračuje".*

---

Jak se dozvíte v kapitole 2, skutečné místo, kde se váš kód kompiluje do IL, závisí na tom, jak vytvoříte a rozmístíte svou webovou aplikaci. Budujete-li svůj webový projekt ve Visual Studiu, zkompiluje se kód do IL, když zkompilujete daný projekt. Jestliže však budujete odlehčený (lighterweight) web bez projektu, kompiluje se kód každé stránky jen tehdy, když o ni požádáte poprvé. Každopádně kód při prvním vykonání projde druhým krokem kompilace (z IL do strojového kódu).

ASP.NET také obsahuje nástroje pro předběžnou kompilaci, s nimiž můžete aplikaci kompilovat rovnou do strojového kódu, jakmile jste ji už rozmístili na ostrý webový server. Umožňuje to vyhnout se zátěži představenou první kompilací, když rozmísťujete dokončenou aplikaci. Tím také zabráníte jiným lidem, aby mohli manipulovat s vaším kódem. Předběžná kompilace se popisuje v kapitole 18.

---

**POZNÁMKA** *Přestože jsou počítačové testy výkonnosti hodně kontroverzní, zajímavé porovnání Javy a ASP.NET najdete na <http://msdn2.microsoft.com/en-us/vstudio/aa700836.aspx>. Mějte ale na paměti, že skutečné nesnáze omezující výkon se obvykle vztahují ke konkrétním "úzkým hrdlům" systému, jakými jsou například rychlost přístupu na disk, zatížení CPU, rychlost síťového připojení atd. V mnoha testech výkonnosti vykazuje ASP.NET lepší výsledky než jiná řešení, protože podporuje ty schopnosti platformy, které mají vliv na výkon, například ukládání dat do cache. Není to následek nárůstu rychlosti, který je způsoben tím, že kód je zkompilovaný.*

---

## Fakt 3: ASP.NET je vícejazyčné

I když při vývoji svých aplikací patrně dáváte přednost jednomu z jazyků před ostatními, tahle volba neurčuje, čeho všeho budete moci docílit ve svých webových aplikacích. Je to proto, že ať použijete kterýkoliv jazyk, kód se vždy zkompiluje do IL.

IL je odrazovým můstkem každé řízené aplikace (řízená aplikace, managed application, je jakákoliv aplikace, která byla napsána pro .NET a vykonává se uvnitř řízeného prostředí CLR). V jistém smyslu je jazykem .NET právě IL. Je to jediný jazyk, kterému CLR rozumí. Abyste IL snadněji pochopili, ukažme si jednoduchý příklad. Podívejme se na kód, který byl napsaný v jazyku C#:

```
using System;
namespace HelloWorld
{
    public class TestClass
    {
```