

Sviluppo



Di Michele Costabile



Lo sviluppo del software sta sempre più oltrepassando i confini del desktop. Oltre a smartphone, tablet, console per il gioco oggi si programma anche per orologi da polso. E i tool per lo sviluppo si adeguano.

Codice ovunque

Esaminiamo, questo mese, una raffica di novità nel panorama globale dello sviluppo di applicazioni, tutte accomunate dall'idea di ubiquità del codice.

Il software è sempre più pervasivo e sempre più massicciamente presente in ogni più piccolo congegno, basti ricordare che abbiamo cominciato a sviluppare in tempi molto recenti anche per gli orologi da polso. Diverse di queste novità provengono da Microsoft, la quale ha compreso che la supremazia sul desktop è di poco valore in un futuro così variegato, considerando soprattutto che la sua posizione di mercato sui telefoni e sugli altri device rimane minoritaria, nonostante gli innegabili sforzi compiuti negli anni appena trascorsi per perfezionare software e hardware degli smartphone Windows.

L'azienda di Redmond ha risposto a queste condizioni aprendosi a tutte le piattaforme, facendo leva sulla posizione di vantaggio che ha fra gli sviluppatori: anche quelli più tradizionalmente legati al software per desktop sentono la pressione o l'esigenza di creare prodotti su device di vario genere e non sono più così strettamente legati a un'unica piattaforma e un unico ambiente operativo.

MICROSOFT CODE

Si tratta di un editor per programmatori, un fratello minore di Visual Studio, scaricabile gratuitamente e disponibile anche su Linux e OS X. Probabilmente è il primo strumento di sviluppo di Microsoft a sbarcare sui due sistemi Unix più diffusi.

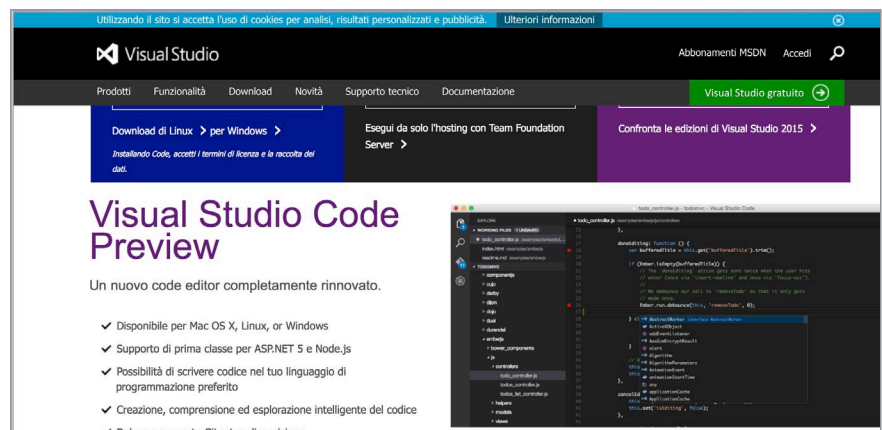
La home page del prodotto è all'indirizzo www.visualstudio.com/products/code-vs. Abbiamo sperimentato con la versione per OS X del programma. La prima piacevole sorpresa è nelle dimensioni: poco meno di 200 MByte, che rendono il download e l'avvio piuttosto rapido.

Quanto all'installazione, non c'è nessuna procedura, in pieno stile Mac.

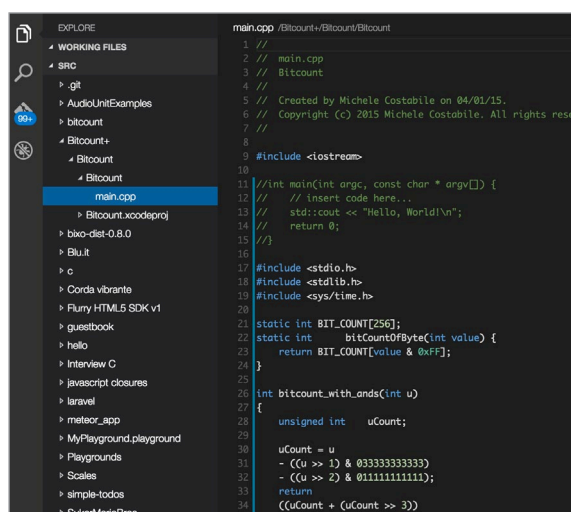
Il look dell'applicazione è in linea con Visual Studio, sia nel tema scuro, sia nel tema chiaro. Il set di colori fa sentire immediatamente a casa chi ha sviluppato una certa consuetudine con Visual Studio.

Dobbiamo impostare una cartella di progetto. Dopo averla selezionata, possiamo lavorare con una vista divisa in tre fasce verticali: una sottile barra di strumenti, una vista ad albero sulla cartella principale e una vista sul codice, che può essere ulteriormente divisa in sezioni, per l'apertura parallela di più file.

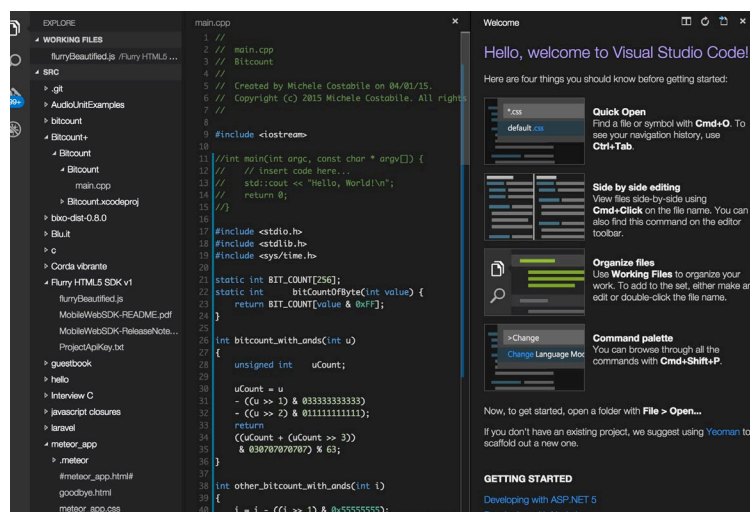
Code consente il debugging di applicazioni C# e Node.js, ma il supporto per altri ambienti è in arrivo. L'ambiente C# richiede su Linux e OS X la presenza di una versione di Mono successiva a



La home page di Visual Studio Code, un editor per programmatori semplice, snello e funzionale.



Ecco la finestra di Code in una sessione di editing C++. Sulla sinistra l'albero della cartella di lavoro.



Code consente la gestione di finestre di codice affiancate, la soluzione è molto utile per confrontare file simili o diverse versioni dello stesso programma.

3.10, come consiglia lo stesso ambiente di sviluppo Microsoft. La colorazione secondo la sintassi supporta i principali linguaggi, come JavaScript, C#, C, C++, Html e Php, ma non Java.

Non mancano le caratteristiche distintive di un buon code editor, per esempio si possono confrontare due file per esaminare le differenze fra uno e l'altro in modo efficace e si può gestire correttamente un repository Git.

Code consente di creare progetti, lanciare compilazioni e analizzare gli errori in modo simile a quello abituale su Visual Studio. Un tutorial interessante sulla creazione di un progetto Typescript con Visual Studio Code, su un Mac, si trova a questo indirizzo <http://michaelcrump.net/using-typescript-with-code>. Nella nostra esperienza su un Mac, Code ha creato una lista di errori anche per file

JavaScript slegati da un progetto.

Code è un editor semplice e snello, con pochissimi fronzoli, basato sulla tastiera, privo di qualsiasi pulsante per agire sul testo e può essere molto gradevole per chi ama il minimalismo senza fronzoli, come la maggioranza degli sviluppatori. La combinazione di tasti Ctrl-Shift-P (o Command-Shift-P) apre una finestra con la lista di comandi dell'editor, come una specie di Powershell interna all'editor, come Copy, Delete line o Find. La lista di comandi è navigabile con Intellisense e accanto a ogni comando è presentato lo shortcut da tastiera che lo lancia. Oltre ai comandi abbiamo i task, ossia i processi di compilazione e esecuzione che possono essere attivati dall'editor. Visual Studio Code è in grado di compilare un progetto e lanciare una sessione di debug con TypeScript o Node.js.

Questo programmer editor ci è stato simpatico fin dall'inizio, sia per il set di colori notturno, sia per la pragmatica semplicità di un insieme di istruzioni lanciabili da una riga di comando interna all'editor (questa funzione sarà molto familiare a chi ha conosciuto gli editor tradizionali di Unix).

Ci sono mancati alcuni comandi, come la selezione all'interno di una coppia di parentesi graffe, ma il progetto è ancora in fase di crescita, come mostra anche una voce del menu Help che consente di richiedere lo sviluppo di nuove funzionalità. In definitiva Code è leggero, ma non è così scarso da non farsi valere nel confronto con gli altri editor gratuiti, se non si considerano i classici ambienti Unix, come *vi* e *emacs*, che hanno decine di release alle spalle

WINDOWS SU RASPBERRY PI

L'idea di eseguire Windows su un sistema microscopico, con la Cpu di un telefono e la memoria e il disco in una scheda SD, avrebbe suscitato risate in passato, quando Windows era alla versione 98 o 2000, o quando Windows CE richiedeva dieci volte le risorse di memoria e di calcolo dei sistemi per palmari concorrenti, come PalmOS o Symbian. Muovendo la moviola in avanti veloce ai giorni nostri, l'esperienza accumulata con Windows Phone e la convergenza con il sistema operativo primario, hanno dato a Microsoft un sistema operativo che può essere magro e performante. D'altra parte bisogna ricordare che Windows ha sempre avuto un nucleo centrale di gestione della memoria e dei processi, che non meritava la cattiva reputazione del guscio esterno del sistema. Windows 10 per Raspberry Pi fa parte del progetto Internet of Things (IoT) di Microsoft, che alla pagina [dev](#)



➤ windows.com/it-IT/iot raccoglie una nutrita quantità di progetti da cui trarre ispirazione. La ragione per mettere una versione di Windows 10 su un Raspberry Pi invece di una distribuzione specifica di Linux, sta nella possibilità di sviluppare applicazioni con un framework .net ridotto e il linguaggio C# o Visual Basic, sfruttando la produttività di un ambiente familiare a moltissimi sviluppatori, che forse non hanno mai pensato di potersi dedicare all'automazione o alla robotica. Insomma, abbiamo una ulteriore conferma del fatto che Microsoft si sta posizionando nella stessa area di Java, come una piattaforma disponibile sui sistemi desktop, sui cellulari, anche con iOS o Android, e anche sui sistemi utilizzati dai maker per creare nuovi prodotti, come il Raspberry Pi. Naturalmente, non ci si può aspettare di non vedere mai la riga di comando: ci sarà da manovrare un po' la Powershell di Windows, soprattutto per la creazione della scheda di avvio del sistema. Le istruzioni dettagliate per chi vuole farsi un giro con la versione più smilza di Windows sono su Github, all'indirizzo ms-iot.github.io/content/win10/SetupRPI.htm. Occorre avere installato Windows 10 sul computer di sviluppo per lavorare con il microcomputer collegato alla porta Usb e Microsoft consiglia di preparare la scheda SD con il sistema su una macchina reale, per avere accesso completo alla porta Usb. Infine, la documentazione è abbondante, anche se purtroppo solo la pagina di facciata è in italiano.

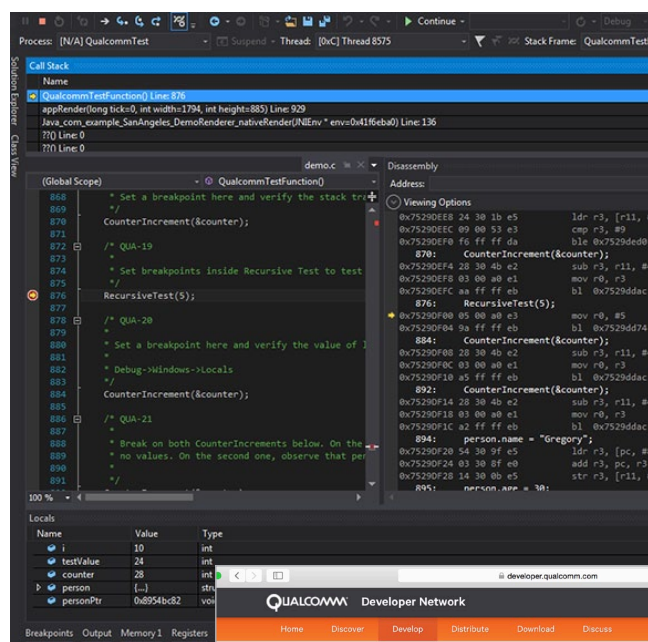


L'approccio Internet of Your Things
Internet of Things (IoT) unisce i dispositivi, i sensori, il cloud, i dati e la tua immaginazione.
Crea ciò che più conta per te.

La pagina Microsoft dedicata alla Internet of Things nel sito per sviluppatori dev.windows.com

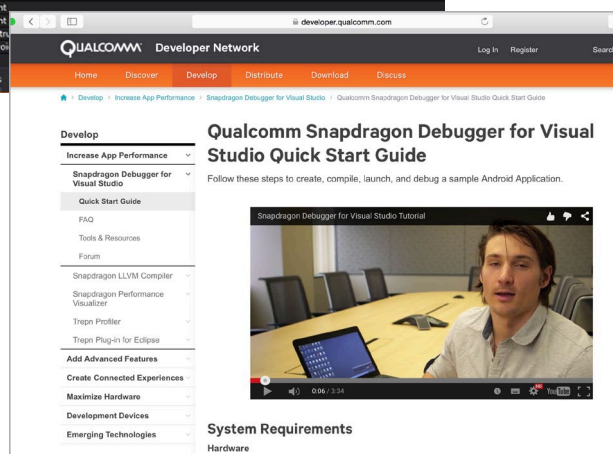


Un kit per la creazione di un robot con Windows 10 e un raspberry pi, da hackster.io.



Il plugin per Visual Studio di Qualcomm in azione durante una sessione di debug

La documentazione che accompagna il plugin di debug per Android è su developer.qualcomm.com



DEBUG PER ANDROID ANCHE IN VISUAL STUDIO

Se non bastassero gli sforzi di Microsoft nel cercare l'ubiquità, si può contare anche sull'aiuto di terze parti, come Qualcomm. La casa produttrice delle Cpu più diffuse nel mondo della telefonia, gli Snapdragon, ha realizzato un plugin per Visual Studio che permette di eseguire il debugging di un'applicazione su un terminale fisico, collegato a una porta Usb, senza muoversi dall'ambiente di sviluppo Microsoft. Il plugin di Qualcomm offre la possibilità di realizzare applicazioni Android per qualsiasi device, indipendentemente dalla Cpu di cui è equipaggiato e supporta il debugging di terminali dotati di processore Snapdragon.

I breakpoint possono essere impostati su codice sorgente, codice oggetto e anche su accesso alla memoria. Il codice può essere eseguito in single step, anche in questo caso sul codice

sorgente o sul disassemblato. Le attuali viste sulla memoria e sui registri sono supportate, così come sono gestiti i thread. Si può anche eseguire il debug post mortem analizzando i file *logcat* o *tombstone* di processi terminati con errore, una funzione estremamente utile per analizzare i problemi più critici. I requisiti di sistema sono moderati: il plugin può essere installato su Windows 7 o Windows 8 e eseguito in Visual Studio 2012 o 2013. Gli ambienti software esterni richiesti sono Apache Ant, il Java Development Kit, lo Sdk Android versione r10d e lo Ndk Android.

Per riassumere, gli sviluppatori che non si sentono a proprio agio in Eclipse come in Visual Studio, hanno a disposizione uno strumento che consente lo sviluppo più hard core su Android in pieno comfort. Tutte le informazioni sono sul sito per sviluppatori di Qualcomm, developer.qualcomm.com.

JAVA 9 IN ARRIVO

Parlando di ubiquità del codice, non possiamo trascurare il capostipite del write once, run everywhere, cioè Java: il linguaggio che ha reso popolare l'idea di una macchina virtuale e di un ambiente applicativo multiplatforma e sta per vedere la sua nona versione.

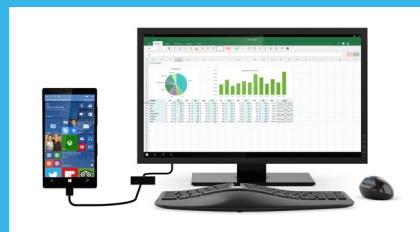
Una email sulla mailing list di sviluppo del jdk 9 (bit.ly/jdk9schedule) ha fissato la data entro cui deve essere completa la funzionalità del codice per la fine di ottobre di quest'anno, mentre il completamento dei test è previsto per febbraio 2016. Per aprile 2016 ci si aspetta il cosiddetto zero bug bounce, ossia il momento in cui il backlog di problemi aperti viene risolto e i bug arrivano a zero. Naturalmente lo zero non viene conservato a lungo perché nuovi bug si aprono, come fa intendere la parola rimbalzo (bounce). Il momento, però, è rilevante nella gestione di una release, perché è il punto in cui il team che risolve i problemi è più efficace di quello che li trova, indicando che il progetto ha raggiunto un punto in cui è sotto controllo. Fra le novità che dovrebbero entrare nel nuovo Jdk, troviamo il progetto Kulla, che si propone di creare un ciclo Repl (read, evaluate, print, loop) ossia un ambiente in cui eseguire Java in modo interattivo, come è possibile con la console JavaScript di un browser, con Python, Ruby e tutti i linguaggi interpretati.

Un altro progetto interessante, il cui nome in codice è Valhalla, ha per obiettivo la razionalizzazione e l'estensione del sistema dei tipi in Java. Il progetto è articolato e raggruppa diversi obiettivi, come avvicinare la semantica di int e java.lang.Integer e migliorare la funzionalità dei generics e la compatibilità con i tipi

value nativi. Java ha scelto fin dall'inizio un approccio misto all'orientazione agli oggetti, con tipi nativi efficienti, come in C, e oggetti come in tanti altri linguaggi, senza optare per un approccio radicale, come Smalltalk o Ruby, in cui ci sono solo oggetti nel sistema dei tipi. C# ha seguito un approccio simile, ma temperandolo con boxing e unboxing automatico, puntando a consentire di trattare ogni cosa come un oggetto, mantenendo l'indubbia efficienza dei tipi nativi fino a quando la logica applicativa di un programma lo consente. Il progetto Valhalla intende portare una razionalizzazione e una sistemazione migliore in questo settore del linguaggio. Una transizione definitiva verso una natura totalmente object oriented è uno dei contenuti di Java 10, quindi non è esattamente dietro l'angolo. Può sembrare che la scelta di stringere i tempi forzando un obiettivo breve termine, come il completamento di tutte le nuove funzionalità nel giro di quattro mesi, sia un po' uno strattone dato alla comunità di sviluppo, ma si tratta di una espressione della volontà di Oracle di dare una cadenza fissa alle release di aggiornamento di Java, puntando a un ciclo di vita triennale per ogni versione del linguaggio. Come è già accaduto con Java 8, si preferisce conservare il passo e rimandare i contenuti, piuttosto che distribuire un aggiornamento quando tutte le funzionalità proposte sono pronte per il rilascio.

Un passo costante nell'innovazione, si propone anche di scoraggiare l'idea che Java sia un linguaggio che ha avuto una crescita e l'ha terminata. Simon Ritter, il product manager del linguaggio alla QCon del 2012 ha affermato che Java non è il nuovo COBOL, un accostamento rivelatore.

UN CONTINUUM DI INTERFACCE



Microsoft ha annunciato parecchie novità interessanti, fra queste una forma di ubiquità che ben si coniuga con quello che abbiamo appena presentato. La feature, chiamata Continuum porta agli utenti la possibilità di collegare schermo, tastiera e mouse di un altro device e utilizzarli nella maniera nativa. Collegando video e tastiera di un PC a un telefono, si può continuare a lavorare su un documento Word aperto in precedenza sul telefono, per esempio sui mezzi, sulla via di casa, e continuare a lavorarci. L'applicazione in uso, per esempio Word, rimane in esecuzione sul telefono. I dati, cioè il documento, non migrano, ma rimangono dove si trovano. L'interfaccia utente diventa quella di Word sul desktop. Questo è una conseguenza del fatto che il sottosistema di Windows che gestisce le interfacce è sempre più astratto. Direttive condizionali in Xaml permettono di descrivere l'interfaccia in modo esaustivo per ogni dimensione dello schermo e ogni capacità hardware, come la presenza di un mouse. Questo ha come conseguenza che collegando la tastiera di un Surface, nel passaggio al modo tablet, vedremo l'interfaccia utente semplificarsi, con pulsanti più larghi per il touch. A un livello più sofisticato, potremo definire il layout di un'interfaccia utente per diverse dimensioni dello schermo, più o meno come si fa da tempo. Questo ha conseguenze profonde sulla struttura del codice: la presentazione dell'interfaccia è separata dalla parte interna del codice in modo netto, mentre il sistema di presentazione dell'interfaccia acquista la proteiforme adattabilità di un browser, mantenendo l'efficienza di un sistema nativo. Purtroppo questo non riguarda la versione corrente dei telefoni Windows, perché richiederà nuovo hardware.

La conseguenza filosofica è che un telefono Windows potrebbe essere il device primario, quello con le applicazioni e i dati più importanti. La differenza principale rispetto a Handoff, la funzione omologa di iOS, è che da un device all'altro si passano gli elementi necessari per la ripresa dell'attività, facendo più affidamento sulla rete e la dotazione software dei telefoni, tablet e computer della rete

OpenJDK

- OpenJDK FAQ
- Installing
- Contributing
- Sponsoring
- Developers' Guide
- Mailing lists
- IRC - Wiki
- Bylaws - Census
- Legal
- JEP Process
- Source code
- Mercurial
- Bundles (6)
- Groups (overview)
- 2D Graphics
- Adoption
- AWT
- Build
- Compiler
- Conformance
- Core Libraries
- Governing Board
- HotSpot
- Internationalization
- JMX
- Members
- Networking
- NetBeans Projects
- Porters
- Quality

JDK 9

The goal of this Project is to produce an open-source reference implementation of the Java SE 9 Platform, to be defined by a forthcoming JSR in the Java Community Process.

The schedule and features of this release are proposed and tracked via the JEP Process, as amended by the JEP 2.0 proposal.

Schedule	
2015/12/10	Feature Complete
2016/02/04	All Tests Run
2016/02/25	Rampdown Start
2016/04/21	Zero Bug Bounce
2016/06/16	Rampdown Phase 2
2016/07/21	Final Release Candidate
2016/09/22	General Availability

The milestone definitions are the same as those for JDK 8.

Features

JEPs targeted to JDK 9, so far

102: Process API Updates

La familiare pagina dello openJDK, con l'attività del gruppo di lavoro per Java 9