



Continuous Integration

Szybki kurs ciągłej integracji w .NET

Autor: Marcin Kawalerowicz

Wersja dokumentu: 1.0

Artykuł opublikowany na stronie: www.continuousintegration.pl.

Wszystkie prawa zastrzeżone. Bezpłatne kopiowanie i rozpowszechnianie artykułu dozwolone pod warunkiem zachowania jego obecnej formy i treści.

Szybki kurs ciągłej integracji w .NET

Ciągła integracja (z ang. Continuous Integration lub CI) to proces budowy oprogramowania po każdorazowym wprowadzeniu kodu źródłowego do repozytorium.

Ten dokument to super szybki, praktyczny kurs CI w środowisku .NET. Dzięki niemu będziesz w stanie w bardzo szybkim tempie uruchomić swój pierwszy proces CI.

Oprogramowanie

Utwórz plik o nazwie HelloCi.cs w katalogu c:\Dev\Starter.

Umieść w nim następujący kod:

Kod – do uzupełnienia

```
public class HelloCi
{
    static void Main()
    {
        System.Console.WriteLine("Hello " + System.Environment.MachineName);
    }
}
```

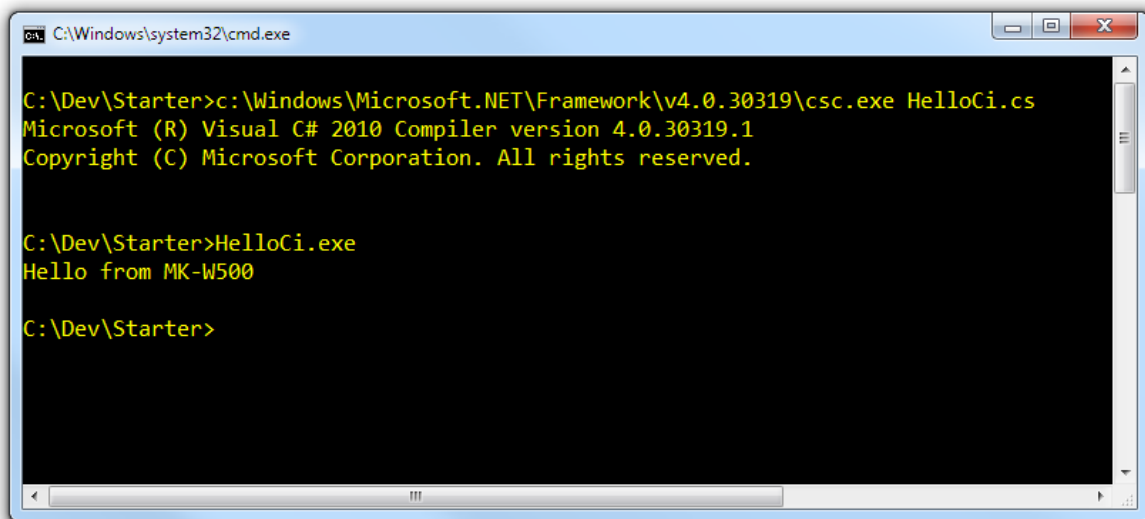
Program ten wyświetli na ekranie konsoli pozdrowienia od komputera, na którym zostanie wykonany.

Otwórz wiersz poleceń. Przejdź do katalogu c:\Dev\Starter i skompiluj program poleceniem:

```
c:\Windows\Microsoft.NET\Framework\v4.0.30319\csc.exe HelloCi.cs
```

Uwaga: zakładam, że masz zainstalowany .NET Framework w wersji 4.0 – to samo można zrobić z wykorzystaniem dowolnej innej wersji Frameworka.

Uruchom program poleceniem HelloCi.exe.



```
C:\Windows\system32\cmd.exe

C:\Dev\Starter>c:\Windows\Microsoft.NET\Framework\v4.0.30319\csc.exe HelloCi.cs
Microsoft (R) Visual C# 2010 Compiler version 4.0.30319.1
Copyright (C) Microsoft Corporation. All rights reserved.

C:\Dev\Starter>HelloCi.exe
Hello from MK-W500

C:\Dev\Starter>
```

Rysunek 1 – Program testowy, na którym oprzemy nasze wprowadzenie do CI może nie jest specjalnie rozbudowany, ale swoją rolę spełni w 100%.

Jeśli twój komputer przesłał ci pozdrowienia to jesteś gotowy na kolejny krok jakim jest automatyzacja procesu budowy.

Automatyzacja budowy

Do automatyzacji budowy naszego oprogramowania wykorzystamy MSBuild. W katalogu `c:\Dev\Starter` utwórz plik o nazwie `HelloCi.proj` i umieść w nim następujący kod:

Kod – do uzupełnienia

```
<Project xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <Target Name="Build" >
    <Delete Files="HelloCi.exe"/>
    <Exec Command="c:\Windows\Microsoft.NET\Framework\v4.0.30319\csc.exe HelloCi.cs"/>
  </Target>
</Project>
```

Otrzymasz w ten sposób skrypt MSBuild ze zdefiniowanym jednym celem o nazwie `Build` (`<Target Name="Build" >`), który ma do wykonania dwa zadania:

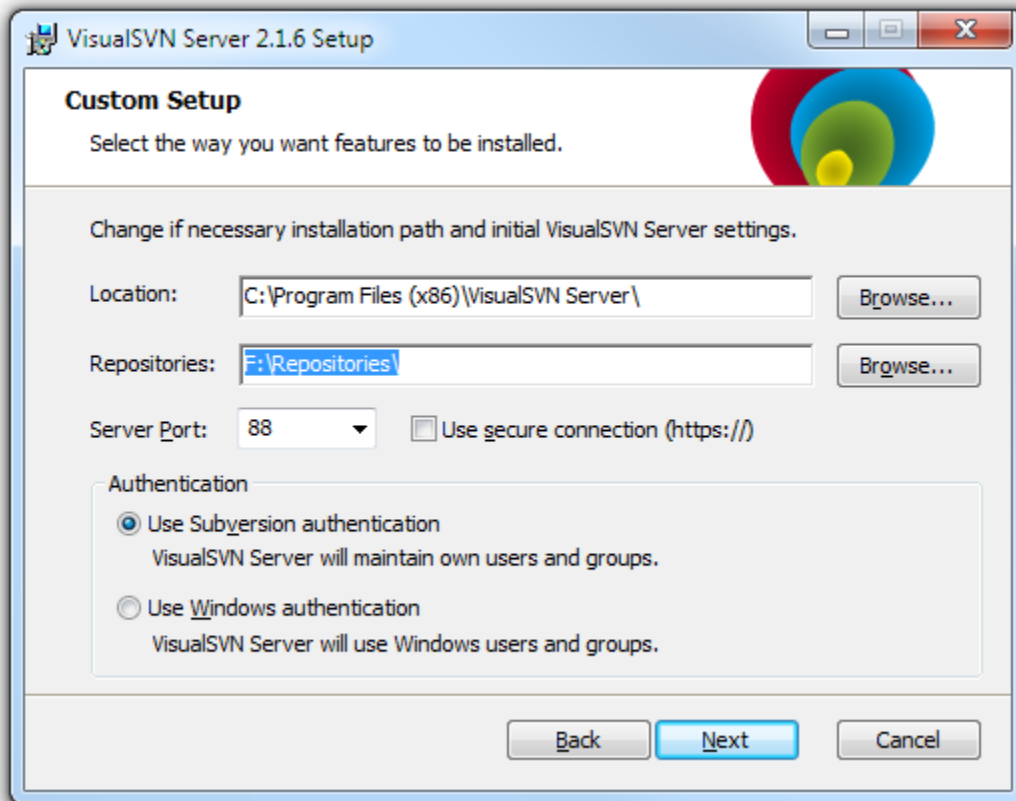
1. oczyszczenie pola budowy poprzez usunięcie ewentualnych pozostałości po poprzednich kompilacjach – `<Delete Files="HelloCi.exe"/>`
2. wykonania komendy, która skompiluje nasz program do postaci wykonywalnej. - `<Exec Command="..." />`

Wykonaj teraz polecenie `c:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe HelloCi.proj` i sprawdź czy znów otrzymałeś plik wykonywalny. Jeśli tak to zautomatyzowałeś swój pierwszy proces budowy oprogramowania. Pora na umieszczenie kodu w centralnym repozytorium.

System kontroli wersji

Jako systemu kontroli wersji użyjemy serwera Subversion (SVN). Na platformie Windows dobrym wyborem jest pakiet VisualSVN Server, którego darmową wersję można pobrać z <http://www.visualsvn.com/server/download/>.

Uruchom pobrany plik MSI i przejdź kilka pierwszych kroków instalacji. W oknie custom setup

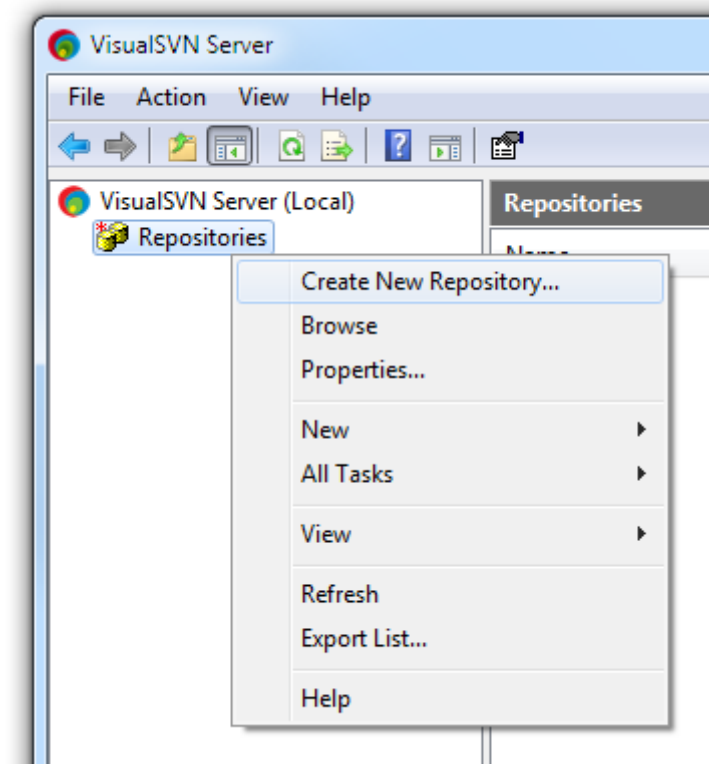


Rysunek 2 – Instalator VisualSVN Server. W tym kroku dokonasz wszystkich najważniejszych kroków podczas instalacji tego serwera Subwersji.

Wybierz dogodne miejsca do umieszczenia plików serwera (Location) oraz repozytoriów (Repositories). Zrezygnuj z zabezpieczania połączenia (wyłącz Use secure connections) i wybierz wolny port na którym będzie dostępny serwer SVN (Server Port). Zmień sposób autentykacji na Windows (Use Windows authentication).

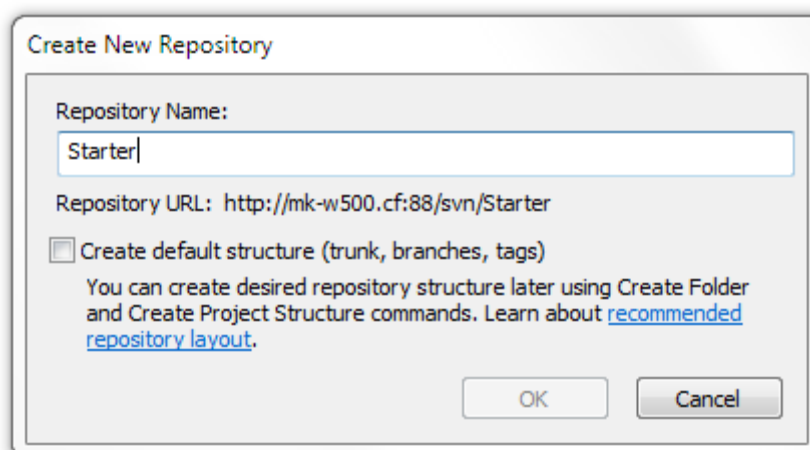
W ostatnim kroku uruchom VisualSVN Server Manager (będzie on również dostępny w Start menu).

W oknie VisualSVN Server Manager kliknij prawym przyciskiem myszy na Repositories i wybierz Create New Repository.



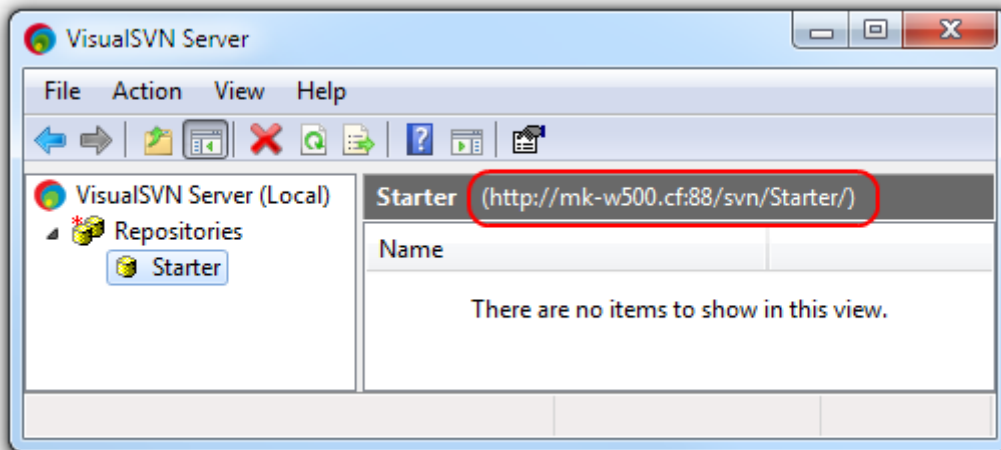
Rysunek 3 – Tworzenie nowego repozytorium w menadżerze VisualSVN Servera jest naprawdę bardzo proste.

Nadaj nowemu repozytorium nazwę „Starter” i nie twórz domyślnej struktury (pozostaw pole wyboru Create default structure puste).



Rysunek 4 – Pora na nadanie repozytorium nazwy oraz decyzję na temat domyślnej struktury katalogów (dla janości wywodu z niej zrezygnujemy).

Zapamiętaj adres nowoutworzonego repozytorium. Jest on widoczny obok jego nazwy w oknie VisualSVN Server Manager.



Rysunek 5 – Oto pełna ścieżka do repozytorium, którą będziemy używać podczas pracy nad projektem.

Pora wprowadzić nasz kod do repozytorium. Można tego dokonać korzystając z dowolnego klienta SVN. Tak się składa, że VisualSVN Server instaluje automatycznie również klienta SVN. Jeśli będziesz korzystał z zewnętrznego serwera SVN będziesz musiał zaopatrzyć się w własnego klienta SVN.

Z poziomu linii komend, będąc w katalogu `c:/Dev/Starter` wydaj komendę:

```
"c:\Program Files (x86)\VisualSVN Server\bin\svn.exe" import http://mk-w500.cf:88/svn/Starter --message "Initial import"
```

Zamieniając nazwę URL do repozytorium zgodnie z nazwą twojego komputera (jeśli korzystasz serwera SVN zainstalowanego na tym samym komputerze możesz użyć localhost) oraz ewentualnie ścieżkę do klienta SVN.

Działania związane z centralnym repozytorium takie jak import czy commit powinny być opatrywane komentarzem (parametr `--message "Tresc komentarza"`).

SVN poprosi cię o podanie hasła dla aktualnego użytkownika (hasło to zostanie zapisane, więc nie będziesz musiał podawać go następnym razem).

Pliki są już w repozytorium. Lokalne źródła już nie będą potrzebne. Za chwilę pobierzemy zawartość repozytorium na lokalny. Usuń cały katalog Starter i z poziomu `c:/Dev` wydaj komendę:

```
"c:\Program Files (x86)\VisualSVN Server\bin\svn.exe" checkout http://mk-w500.cf:88/svn/Starter
```

Lokalne repozytorium gotowe. Od teraz po dokonaniu zmian będziesz mógł wysłać je do centralnego repozytorium wydając komendę:

```
"c:\Program Files (x86)\VisualSVN Server\bin\svn.exe" commit HelloCi.cs --message "Change".
```

Od teraz checkout i commit powinny towarzyszyć tobie nieprzerwanie podczas pracy nad kodem źródłowym.

Serwer CI

Hudson CI to świetny darmowy serwer CI. Do działania wymaga jednak środowiska uruchomieniowego Javy. Jeśli go nie masz to pobierz go ze strony java.com. Sam serwer Hudson CI możesz pobrać ze strony <http://hudson-ci.org/>. Będzie to plik z rozszerzeniem war. Skopuj ten plik do katalogu c:/Hudson po czym z poziomu linii komend wydaj polecenie

```
java -jar hudson-[nr_wersji].war
```

gdzie [nr_wersji] to numer właśnie pobranej wersji serwera Hudson CI. Jeśli miałeś wolny port 8080 to na tym porcie będziesz miał dostępny panel sterownia twoim serwerem CI (jeśli port ten masz zajęty musisz wystartować Hudsona z następującym dodatkowym parametrem -- httpPort=[numer_portu]).

Otwórz przeglądarkę internetową i przejdź do strony

```
http://localhost:8080
```

Powinieneś zobaczyć panel administracyjny Hudson CI.

#	Status
1	Idle
2	Idle

Rysunek 6 – Panel administracyjny serwera ciągłej integracji Hudson CI gotowy do użytku.

Zanim zaczniemy definiować zadania dla serwera CI musimy zainstalować dodatkowy plug-in dla środowiska .NET. W menu wybierz „Manage Hudson”. Spośród dostępnych opcji wybierz zarządzania plug-inami „Manage Plugins”.



Manage Plugins

Add, remove, disable or enable plugins that can extend the functionality of Hudson.

Rysunek 7 – Hudson CI wyposażony jest w mechanizm wtyczek, którym zarządzać można po wybraniu powyższej opcji.

Przejdź to zakładki z dostępnymi plug-inami „Available”.

Updates			
Available	Installed	Advanced	
Install			
Name ↓	Version	Installed	
<input type="checkbox"/> Subversion Plugin This plugin adds the Subversion support (via SVNKit) to Hudson.	1.23	1.20	

This page lists updates to the plugins you currently use.

Rysunek 8 – Wtyczka do serwera SVN jest już zainstalowana i można ją zaktualizować (nie musimy tego robić w tej chwili).

Na liście poszukaj punktu „MSBuild Plugin” i oznacz go flagą.

<input checked="" type="checkbox"/>	MSBuild Plugin This plugin allows you to use MSBuild to build .NET projects.	1.3
-------------------------------------	---	-----

Rysunek 9 – To co potrzebujemy, by zautomatyzować proces budowy oprogramowania dla platformy .NET to wtyczka do MSBuild.

Kliknij przycisk „Install” u dołu strony.

Page generated: 08-Mar-2011 20:23:08 [Hudson ver. 1.396](#)

Rysunek 10 – Po wybraniu wtyczki do instalacji należy nacisnąć przycisk Install.

Plug-in do MSBuilda zostanie zainstalowany.

Installing Plugins/Upgrades

⚠ Once the installation is completed, Hudson needs to be restarted for changes to take effect.

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

MSBuild Plugin ● Success

Rysunek 11 – Mechanizm wtyczek w Hudson CI jest bardzo wygodny. Po automatycznej instalacji należy tylko zrestartować tylko serwer.

Zacznie on działać dopiero po restarcie serwera. Wyłącz go wybierając z menu „Manage Hudson” oraz „Prepare for Shutdown”.



Prepare for Shutdown

Stops executing new builds, so that the system can be eventually shut down safely.

Rysunek 12 – Przed restartem serwera trzeba go przygotować do wyłączenia.

I naciskając Ctrl+C w oknie wiersza poleceń.

Po ponownym wystartowaniu serwera CI wybierz z menu „Manage Hudson” oraz „Configure System”.



Configure System

Configure global settings and paths.

Rysunek 13 – Po instalacji wymaganych wtyczek należy je skonfigurować.

W obszarze MSBuild Builder naciśnij przycisk “Add”.

MSBuild Builder

MSBuild installation

Add

List of MSBuild installations on this system

Rysunek 14 – Konfiguracja wtyczki MSBuilda polega na dodaniu danych na temat jego położenia.

I zdefiniuj ścieżkę do MSBuild.exe.

MSBuild Builder

MSBuild installation

name

MSBuild 4.0

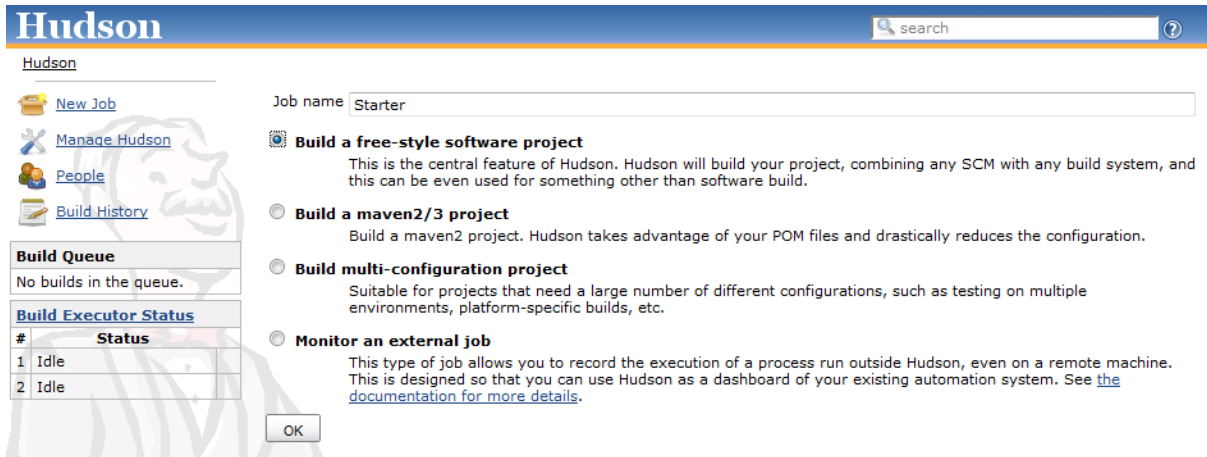
Path To msbuild.exe

c:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe

Rysunek 15 – My zdefiniujemy położenie MSBuilda w wersji 4.0. Jeśli masz .NET Framework w wersji Extended na komputerze to masz również MSBuilda.

Po czym naciśnij przycisk „Save” u dołu strony.

Teraz możesz zdefiniować tak zwanego joba. Job to zadanie, które wykonuje serwer ciągłej integracji. Kliknij link „create new job”.



Hudson

Hudson

[New Job](#)
[Manage Hudson](#)
[People](#)
[Build History](#)

Job name

Build a free-style software project
 This is the central feature of Hudson. Hudson will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Build a maven2/3 project
 Build a maven2 project. Hudson takes advantage of your POM files and drastically reduces the configuration.

Build multi-configuration project
 Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Monitor an external job
 This type of job allows you to record the execution of a process run outside Hudson, even on a remote machine. This is designed so that you can use Hudson as a dashboard of your existing automation system. See [the documentation for more details](#).

Build Queue
 No builds in the queue.

Build Executor Status

#	Status
1	Idle
2	Idle

Page generated: 08-Mar-2011 20:17:34 [Hudson ver. 1.396](#)

Rysunek 16 – Definiowanie nowego procesu (tzw. Joba) w Hudson CI.

Nadaj nazwę nowemu zadaniu „Starter” i wybierz „Build a free-style software Project”.

W obszar „Source code management” zdefiniuj ścieżkę do centralnego repozytorium.

Source Code Management

None
 CVS
 Subversion

Modules Repository URL
 Local module directory (optional)

Use update
 If checked, Hudson will use 'svn update' whenever possible, making the build faster. But this causes the artifacts from the previous build to remain when a new build starts.

Revert
 If checked, Hudson will do 'svn revert' before doing 'svn update'. This slows it down, but will prevent files being modified from build to build.

Repository browser

Rysunek 17 – Definicja położenia repozytorium z kodem źródłowym aplikacji.

W obszarze „Build Triggers” wybierz okresowe odpytywanie repozytorium „Poll SCM” oraz interwał tego odpytywania w notacji Crona (* * * * * oznacz co minutę).

Build Triggers

Build after other projects are built ?
 Build periodically ?
 Poll SCM ?
 Schedule ?

```
*****|
```

Rysunek 18 – Definicja wyzwalacza integracji. W tym wypadku to zwykłe odpytywanie serwera wersji co 1 minutę (powoduje je znany z crona zapis `*****`).

W obszarze „Build” wybierz „Build a Visual Studio project or solution using MSBuild.”.

Build

Add build step ▼

- Invoke Ant
- Execute shell
- Execute Windows batch command
- Invoke top-level Maven targets
- Build a Visual Studio project or solution using MSBuild.

Rysunek 19 – Definicja narzędzia, które ma być użyte do budowy.

Wybierz wcześniej skonfigurowaną wersję MSBuild 4.0 oraz wpisz nazwę skryptu MSBuild, który utworzyliśmy wcześniej.

Build

☰ **Build a Visual Studio project or solution using MSBuild.**

MsBuild Version ▼
 MSBuild 4.0

MsBuild Build File ?
 HelloCi.proj

Command Line Arguments ?

Delete

Rysunek 20 – Narzędziem, które dokona integracji będzie MSBuild w wersji 4.0, a definicja zostanie podana w pliku HelloCi.proj.

Całość zapisz używając przycisku „Save” u dołu strony.

Gotowe! Twój pierwszy proces CI jest gotowy. Spróbuj teraz zmienić coś w repozytorium. Wprowadzić zmiany za pomocą komendy `svn commit` i sprawdź czy Hudson CI zadziała tak jak powinien. Czy będzie budował twoje oprogramowanie po każdym wprowadzeniu zmian do systemu kontroli wersji.

To pierwszy tutorial z serii przybliżającej zagadnienia CI. Więcej szczegółów na stronie: www.continuousintegration.pl.

Dziękujemy naszemu sponsorowi firmie CODEFUSION Sp. z o.o. Dzięki nim mógł powstać ten dokument.

Marcin Kawalerowicz



Artykuł opublikowany na stronie: www.continuousintegration.pl.

Wszystkie prawa zastrzeżone. Bezpłatne kopiowanie i rozpowszechnianie artykułu dozwolone pod warunkiem zachowania jego obecnej formy i treści.



CODEFUSION Sp. z o.o.
ul. Powstańców Śląskich 25/18
45-086 Opole, Poland