

ツールで爆速開発！

使える3倍速くなる 実践テクニック入門

JetBrains Rider (C#) hands on

#1 – Get Started 入門編1



youhei.yamada
[@sun flat yamada](https://sun.flat.yamada)

Hands On

#1.0 – 開始前の確認



Hands On は以下を前提に進めます

- RiderがInstall済み
- EntityFrameworkのサンプルをBuild確認済み
(Hands Onの中でもこのコードを使います)
- 使うPCの時刻がファシリテーターのPCとズれていない

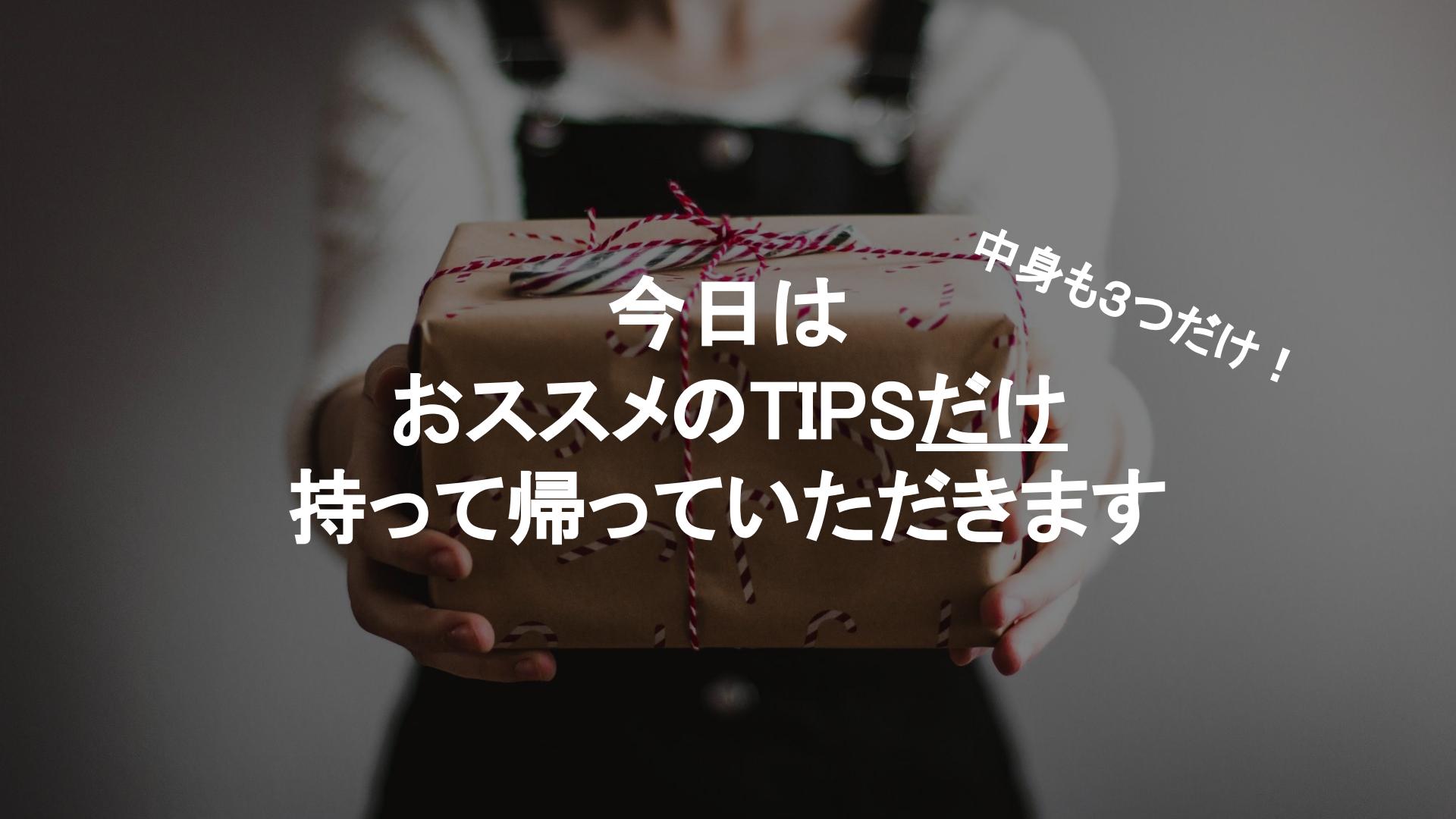
聞くだけも歓迎します！
ただ、フォローはHands On参加者を
優先させていただきますのでお含みおきください。

M はじめに
大事だと思う姿勢 D



欲張りすぎない

3つくらいの
トピックから
始めるので十分

A close-up photograph of a person's hands holding a rectangular gift box. The box is wrapped in brown paper and tied with red and white striped twine. The background is dark and out of focus.

今日は
おススメのTIPSだけ
持って帰っていただきます

中身も3つだけ！

では、
一緒に始めて行きます

Hands On

#1.1 – 現状を知る





自分のコード速度
すぐに言えますか？



今のコーディング
速度を計測します

例題コードを
全く同じように
書き写してください

ルール

- 見本からのコピペ禁止。ファイルは2つ、約30行 + 約45行 だけです。
- コード不要の記載がある部分を除き、コメントも対象です。
- いつも使っている開発環境を使って良いです。機能制限はありません。
- フォーマットは出来るだけ合わせてください。

インデント、スペース、改行、コメントスタイル等です。

- 完成したらファイルを保存し、閉じてください。
- ファイルの最終更新時刻を結果リストに記載してください。

記入先はファシリテーターより指示があります。

終わったら、次のようなことを書き出してみましょう。

- 一番速い人と比べ、どのくらい差がありましたか？
- 何が面倒でしたか？
- どうすればもっと早くコードできそうですか？

10minあたりを目途に締め切りますが、終わらなくともOK！
後で、ファイルサイズ比で完成までの時間を概算します。
(10minかけて80%だったら、残り20%に2minかかるとして)



Hands On

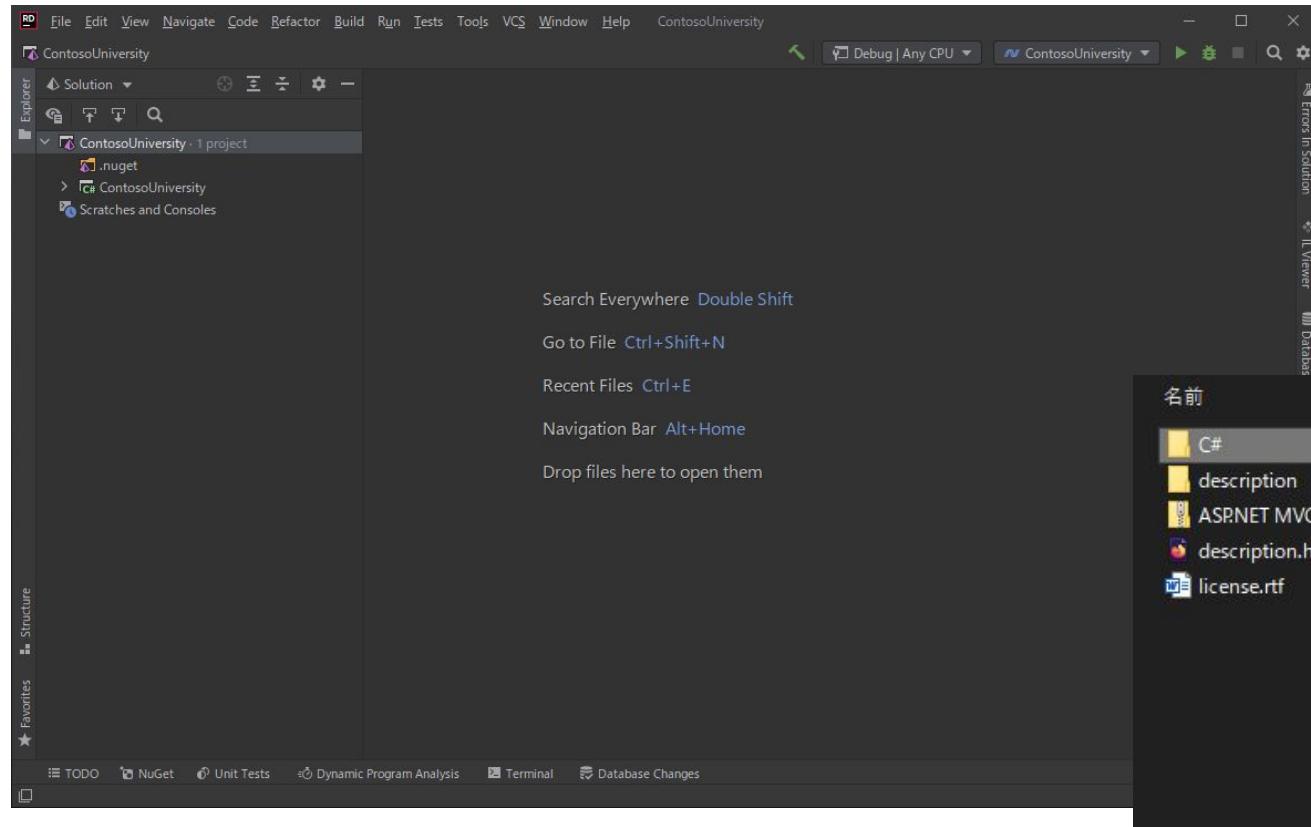
#1.2 – 簡単にコード速度を上げる、 3つのオススメTIPS



Riderの機能を
一緒に使ってみます

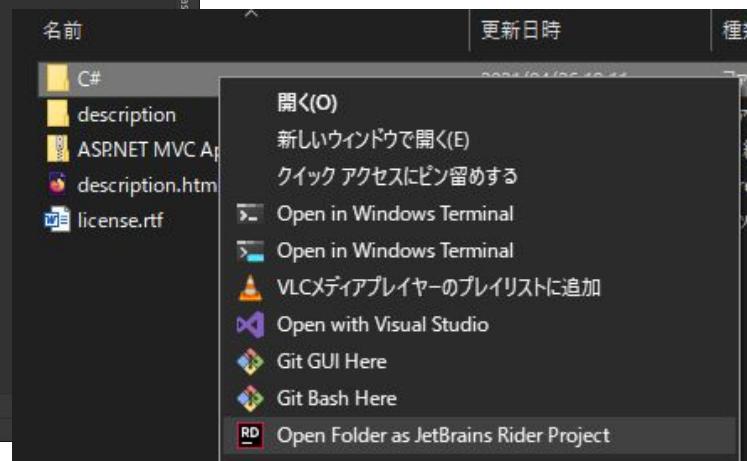


事前準備で使ったEFのサンプルをRiderで開いてください



‘ContosoUniversity’ というプロジェクトが開けていればOK

↓のような’C#’というディレクトリを開けばOK



CourseController.cs を開いてください

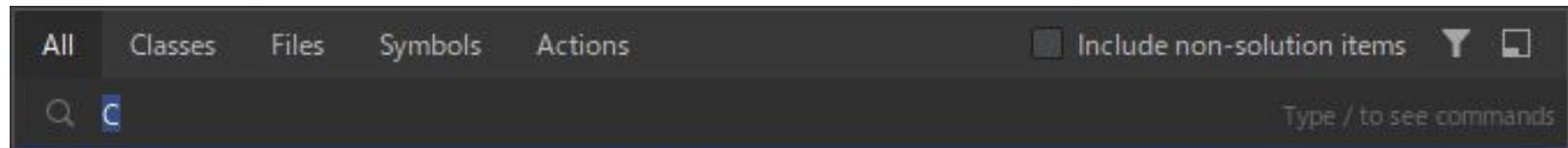
CourseControlTMを聞いてください
何それ？どこにあるの？



書く以前に
だいたい探すよね？

何でも探せる魔法のコマンドがあります

Shiftを2回タイプ！。何でも検索ウインドウが出せます！



↑ここに CourseController と入力すると…



(補足) VSCode の Ctrl + t も同様の機能が使えますが JetBrainsの方が高機能です

さらに裏技

(実は正式機能ですが)

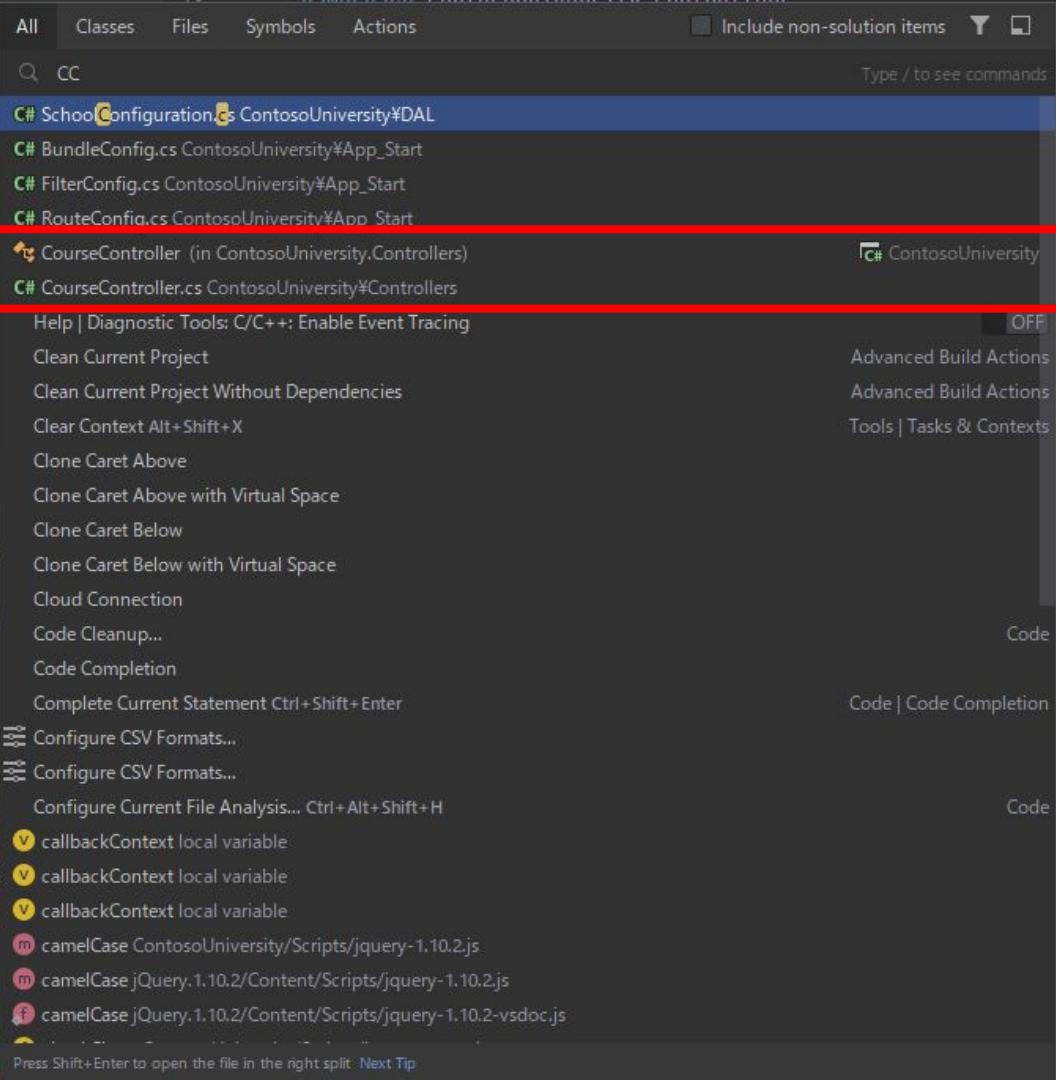
先ほどの何でも検索ウィンドウに
“CC”と入力してみてください。
CourseControllerが表示されます！

そう、単語の頭文字などで
検索できるんです！

CourseController

長いスペルを打つ時間すら
削れます！！

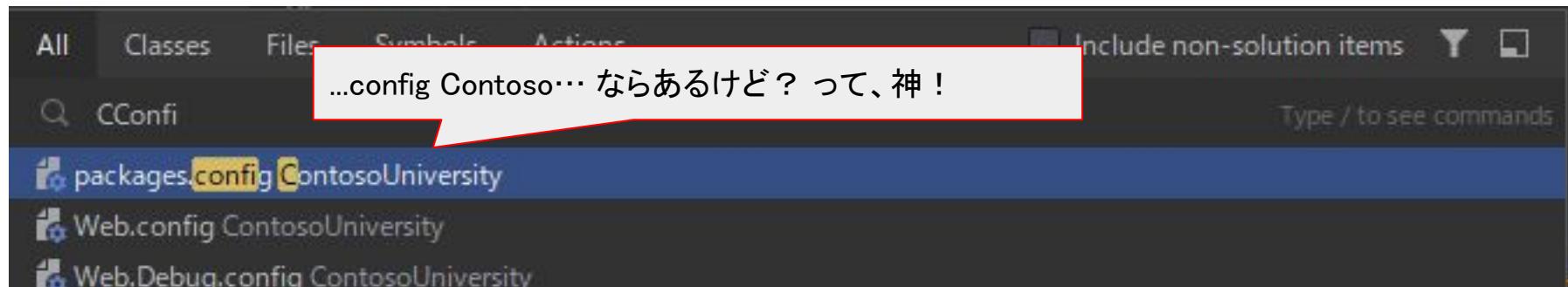
(補足) これも VSCode なら類似機能が使えます



さらに、その2

思い違いをしていても、優しく教えてくれます。

例えば、「C なんちやら Config みたいな何かだったな～」と思って
“CConfig”と打つと…



話を戻して CourseController.cs を開けましたね？

The screenshot shows the Visual Studio IDE interface with the following details:

- File Bar:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tests, Tools, VCS, Window, Help, ContosoUniversity.
- Solution Explorer:** Shows the project structure:
 - ContosoUniversity - 1 project
 - .nuget
 - ContosoUniversity
 - Dependencies
 - Properties
 - App_Data
 - App_Start
 - Content
 - Controllers
 - CourseController.cs (selected)
 - DepartmentController.cs
 - HomeController.cs
 - InstructorController.cs
 - StudentController.cs
 - DAL
 - SchoolConfiguration.cs
- Code Editor:** The `CourseController.cs` file is open, showing the following code:

```
1  using System;
2  using System.Collections.Generic;
3  using System.Data;
4  using System.Data.Entity;
5  using System.Linq;
6  using System.Net;
7  using System.Web;
8  using System.Web.Mvc;
9  using ContosoUniversity.DAL;
10 using ContosoUniversity.Models;
11 using System.Data.Entity.Infrastructure;
12
13 namespace ContosoUniversity.Controllers
14 {
```
- Status Bar:** Event Log: General Database, 2021/04/26, 19:38 Default SQL dialect: You can set Microsoft SQL Server as default SQL dialect for this project or configure SQL dialects.
- Bottom Navigation:** TODO, Dynamic Program Analysis, Unit Tests, Terminal, NuGet, Python Packages, Event Log.
- Bottom Status:** Default SQL dialect: You can set Microsoft SQL Server as default SQL dialect for this p... (10 minutes ago), ContosoUniversity, 6:18, CRLF, UTF-8, 4 spaces, AWS: No credentials selected.

まず、ハイライトの違うところが見えると思います

簡単な例でツールの使い方を説明します。

The screenshot shows a C# code editor with the file `CourseController.cs` open. The code lists various using statements:

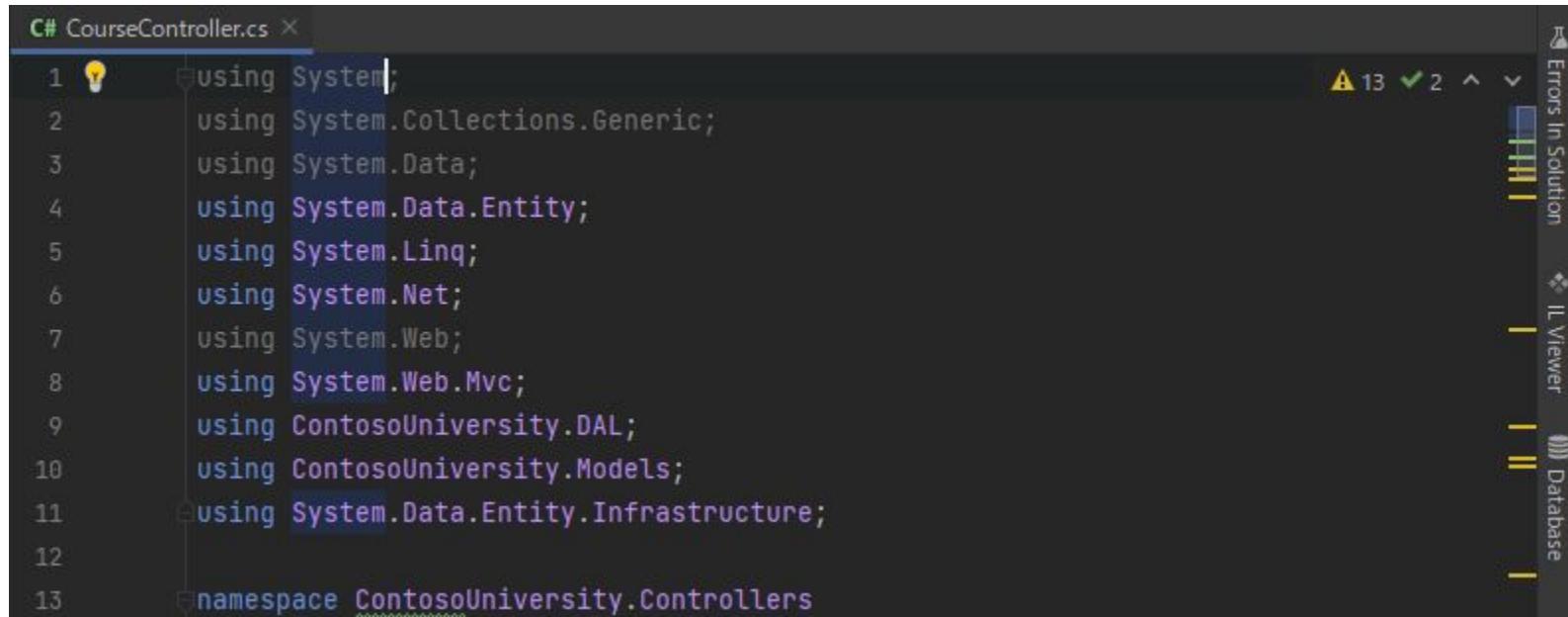
```
C# CourseController.cs ×
1  using System;
2  using System.Collections.Generic;
3  using System.Data;
4  using System.Data.Entity;
5  using System.Linq;
6  using System.Net;
7  using System.Web;
8  using System.Web.Mvc;
9  using ContosoUniversity.DAL;
10 using ContosoUniversity.Models;
11 using System.Data.Entity.Infrastructure;
12
13  namespace ContosoUniversity.Controllers
```

A red callout box highlights the line `using System;` with the text "グレーアウトされているところがある". A red arrow points from this text to the line in the code editor.

To the right of the code editor is the Task List tool window, which displays several items:

- Errors In Solution (yellow icon)
- IL Viewer (green icon)
- Database (blue icon)

該当行にカーソルを置くと  イコンが表示されます

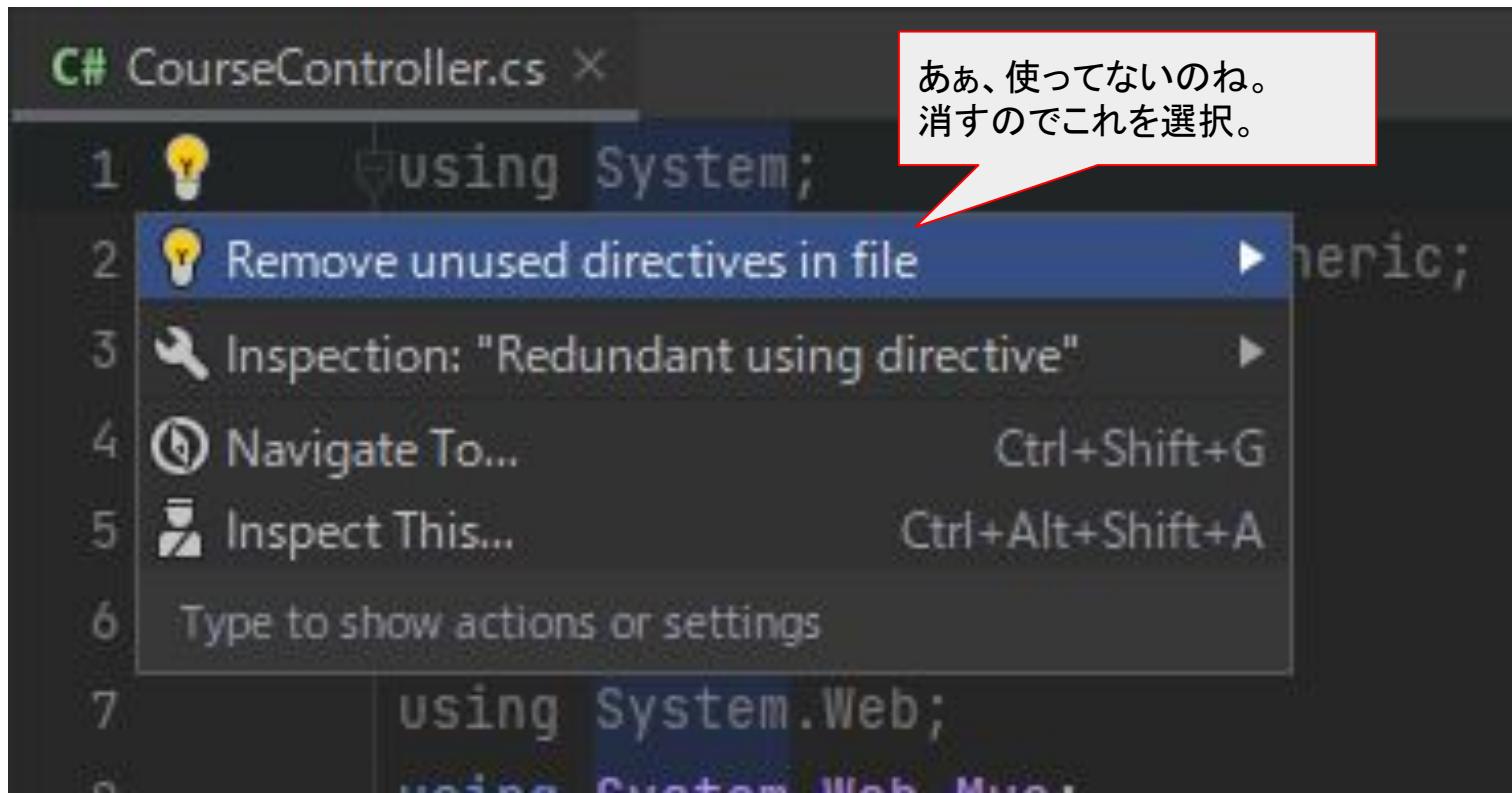


The screenshot shows a dark-themed code editor window for C# named "CourseController.cs". The code listed is:

```
C# CourseController.cs ×
1  using System;
2  using System.Collections.Generic;
3  using System.Data;
4  using System.Data.Entity;
5  using System.Linq;
6  using System.Net;
7  using System.Web;
8  using System.Web.Mvc;
9  using ContosoUniversity.DAL;
10 using ContosoUniversity.Models;
11 using System.Data.Entity.Infrastructure;
12
13 namespace ContosoUniversity.Controllers
```

A yellow lightbulb icon is positioned above the first line of code (line 1). The status bar at the bottom right shows "A 13" and "V 2". To the right of the editor is a vertical toolbar with icons for "Errors In Solution" (red exclamation mark), "IL Viewer" (blue assembly icon), and "Database" (yellow database icon).

ここで **Alt + Enter** !!



消えます。賢く全部。

そうそう、1発でこうしたかったのよ。

C# CourseController.cs ×

```
1  using System;
2  using System.Collections.Generic;
3  using System.Data;
4  using System.Data.Entity;
5  using System.Linq;
6  using System.Net;
7  using System.Web;
8  using System.Web.Mvc;
9  using ContosoUniversity.DAL;
10 using ContosoUniversity.Models;
11 using System.Data.Entity.Infrastructure;
12
13 namespace ContosoUniversity.Controllers
```

C# CourseController.cs ×

```
1  using System.Data.Entity;
2  using System.Linq;
3  using System.Net;
4  using System.Web.Mvc;
5  using ContosoUniversity.DAL;
6  using ContosoUniversity.Models;
7  using System.Data.Entity.Infrastructure;
8
9  namespace ContosoUniversity.Controllers
```

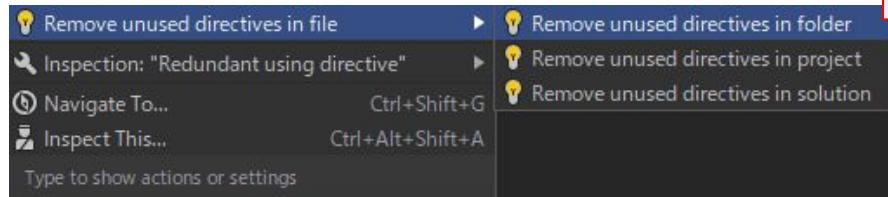


補足

未使用usingの削除機能はあたり前の機能で、VisualStudioでも出来ます。

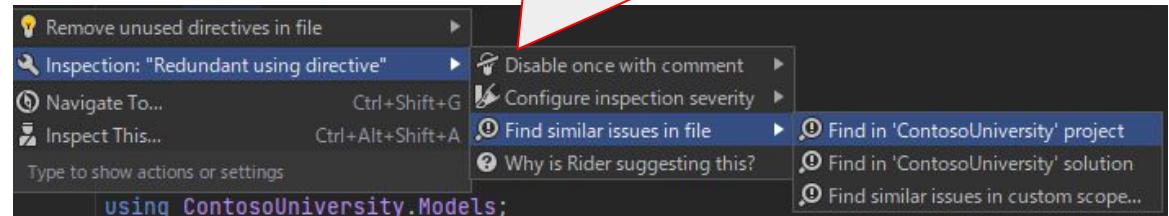
ここでは操作感を説明するための簡単な例として取り上げています。

JetBrains製のツールの良いところは、自動判定や対象範囲の選択など、痒いところに手が届くことです。



その場で適用範囲をすぐ選べる
(キーボード操作だけで)

類似を探したい、といったアルアル事例や、
無視範囲の設定や警告レベルの調整など
も、気付いたところから出来る



次の例

結局は Alt + Enter すればOK、という話
Riderが見つけた指摘の修正方法

おや？なんだか警告が出ています。確認してみましょう。

The screenshot shows a C# code editor with the file `CourseController.cs` open. The code includes standard .NET namespaces like `System.Data.Entity` and `System.Web.Mvc`, along with specific project references to `ContosoUniversity.DAL` and `ContosoUniversity.Models`. A red box highlights the `using System.Data.Entity.Infrastructure;` line. In the top right corner of the editor, there is an `Errors` icon with a count of 9 errors and 2 warnings. A callout bubble with a red border points from the text "F2 キーをタイプします" to this icon.

```
C# CourseController.cs
1  using System.Data.Entity;
2  using System.Linq;
3  using System.Net;
4  using System.Web.Mvc;
5  using ContosoUniversity.DAL;
6  using ContosoUniversity.Models;
7  using System.Data.Entity.Infrastructure;
8
9  namespace ContosoUniversity.Controllers
```

F2 キーをタイプします

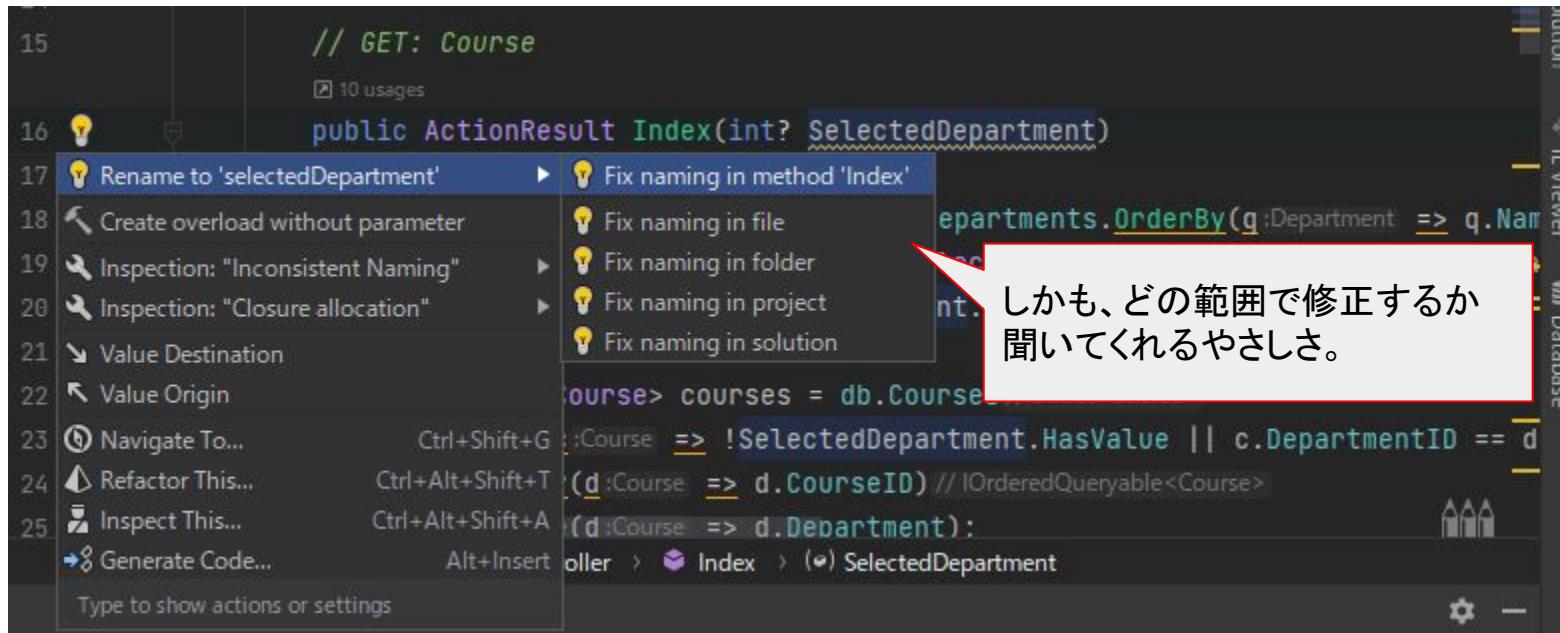
指摘箇所にジャンプしました。再び **Alt + Enter !!**

The screenshot shows a code editor for C# in Visual Studio. The file is CourseController.cs. The code is as follows:

```
C# CourseController.cs ×
12     {
13         private SchoolContext db = new SchoolContext();
14
15         // GET: Course
16         [10 usages]
17         public ActionResult Index(int? SelectedDepartment)
18         {
19             var departments:List<Department> = db.Departments.OrderBy(q:Department => q.Na
20             ViewBag.SelectedDepartment = new SelectList(departments, "Depa
21             int departmentID = SelectedDepartment.GetValueOrDefault();
```

The word "SelectedDepartment" is underlined with a red squiggle, indicating a potential issue. The status bar at the top right shows "A 9 ✓ 2 ^ v". To the right of the code editor is the Error List window, which is currently empty. The interface includes standard Visual Studio navigation and search tools.

コードスタイルに従ってないよ、と教えてくれています



この Hands On ではデフォルト設定のままなので JetBrains おススメのコードスタイルでチェックされていますが、任意の指定が可能です。`.editorconfig` や `.clang-format` で定義したコードスタイルの設定を使うこともできます。プロジェクトに合った設定にすれば、日々キレイになります。

次は147行目を見てください (Ctrl + Gで147と入力)

警告の理由、すぐわかりますか？

(補足)

L.147 は ファイル先頭の未使用usingを削除している場合の行数です。
サンプル元コードを変更していない人は行数が多少ズレます。

Alt + Enter で見てみましょう。

```
141         // POST: Course/Delete/5
142         [HttpPost, ActionName("Delete")]
143         [ValidateAntiForgeryToken]
144             ↗ 2 usages
145             public ActionResult DeleteConfirmed(int id)
146             {
147                 Course course = db.Courses.Find(id);
148                 db.Courses.Remove(course);
149                 db.SaveChanges();
150                 return RedirectToAction("Index");
151             }
152         }
```

null の可能性が考慮できていないこと、分かりました？

どうコードするかは設計次第ですが、その場で直せる候補が表示されます。

```
146     Course course = db.Courses.Find(id);  
147     db.Courses.Remove(course);  
148     ⚡ Coalesce with fallback value  
149     ⚡ Throw exception when null  
150     ↵ Check expression for null  
151     ↵ Assert expression is not null  
152     ↵ Add argument name 'entity'  
152     ⚡ Inspection: "Possible 'null' assignment to non-nullable entity" → edits()  
152     ⚡ Navigate To... Ctrl+Shift+G
```

次は Project_Readme.html を
開いてみてください

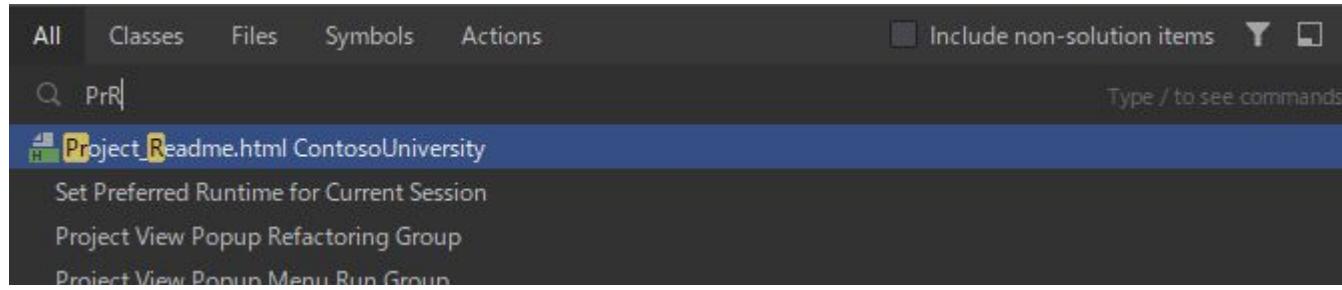
探し方、もうわかりますよね？

え？ Project_Read...と打ってます？

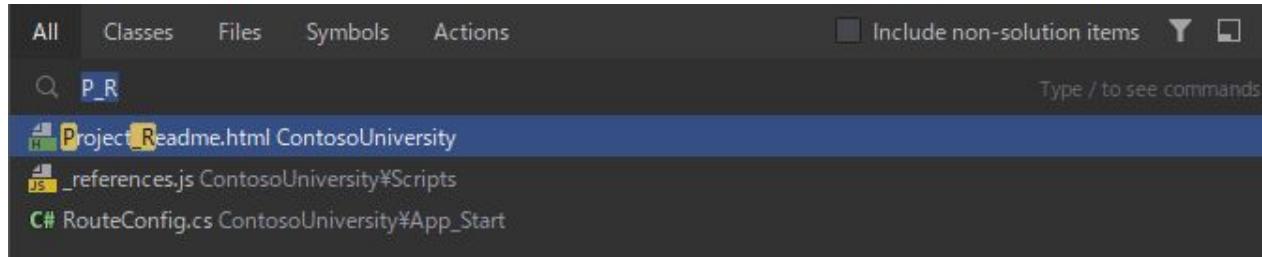
まさか、ね。

ベストな探し方は PrR です

応用、できました？



P_R でも行けますが、shiftを使う記号だと打ちにくい…？



【shift ×2】と【Alt+Enter】を紹介しました。
次が3つ目、本日の目玉機能です。

何が起こるか試してみてください

<body> の1行下に #myid と打って Tab をタイプしてみてください。

The screenshot shows a code editor window with the file 'Project_Readme.html' open. The code is as follows:

```
94      </style>
95  </head>
96  <body>
97  #myid
98  <div id="header">
99    <h1>Your ASP.NET application</h1>
100   <p>Congratulations! You've created a project</p>
101 </div>
```

A red callout box points from the text '#myid' in line 97 to a tooltip containing the Japanese text '#myid と打ち終わったら
続けてTab' (After typing #myid, press Tab). Another red callout box points from the bottom-left corner of the code editor area to the text 'Ctrl + G → 96 (行目)' at the bottom left.

Ctrl + G → 96 (行目)

何と！面倒なHTMLタグが勝手に生成される！



A screenshot of the Visual Studio IDE showing a file named "Project_Readme.html". The code editor displays the following HTML structure:

```
94     </style>
95 </head>
96 <body>
97     <div id="myid"></div>
98     <div id="header">
99         <h1>Your ASP.NET application</h1>
100        <p>Congratulations! You've created a project</p>
101    </div>
```

The line `<div id="myid"></div>` is highlighted with a red box. A callout bubble with a red border points from this box to the text:
タイプ量が半分で済む！
括弧の数を間違ったりもしない！
スラッシュ抜けもない！

カンの良いあなた、もうお気づきですね？
id が # で出るなら、classだったら…



A screenshot of the Visual Studio IDE showing a file with the following code:

```
96 <body>
97     <div id="myid"></div>
98     .myclass
99     <div id="header">
```

The class definition `.myclass` is highlighted with a red box. A callout bubble with a red border points from this box to the text:
CSS 記法だと id は # 、
Class は .(ドット) だから…

はい。期待通り生成されます。

```
96      <body>
97          <div id="myid"></div>
98          <div class="myclass">|</div>
99          <div id="header">
```

これは emmet (エメット) という記法です。

HTML や CSS を独自の省略形でタイプすると自動補完(自動生成)します。

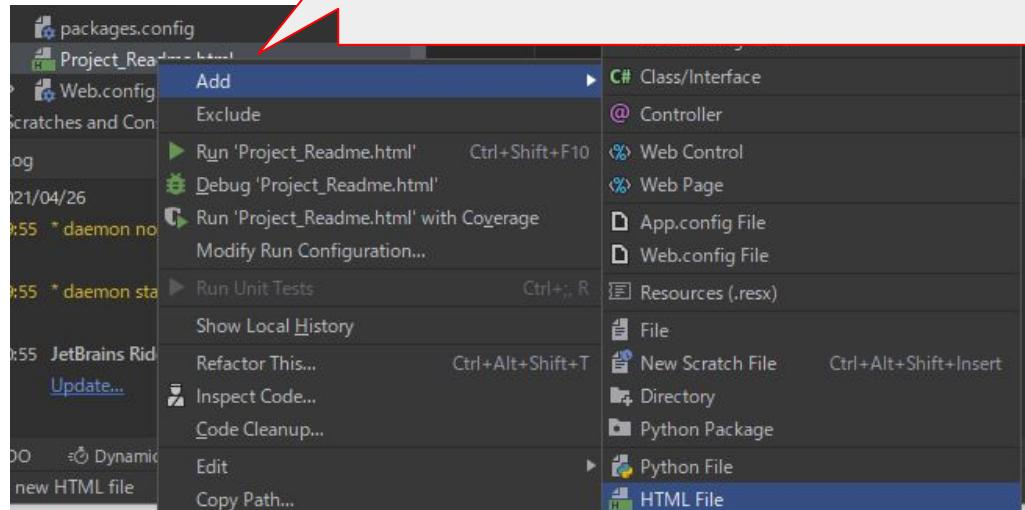
https://www.jetbrains.com/help/rider/Settings_Emmet.html

他のツールでもプラグインが提供されていることもあるので、HTML系を書く場合はおススメのテクニックです。単純にタイプ量が半分以下になり、ミスも減ります。その分、早く仕事が終わります。

emmet、もっとすごいこと出来るんじゃないの？

カラのHTMLファイルを作つてみてください。

(参考) HTMLファイルの作り方
ファイルブラウザ上で
[右クリック] > [Add] > [HTML File]



あれ？勝手にテンプレートが出るんですが…？

```
sample.html ×
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>Title</title>
6   </head>
7   <body>
8
9   </body>
10 </html>
```

これも emmet の機能が自動的に動く仕掛けで実現されています。

せっかくですが、一度全部消してから、
‘!’と打ってTabで補完してみてください。

もっと詳しいテンプレートが出ます

The screenshot shows a code editor window with the file 'sample.html' open. The code is an HTML document with standard tags like <!doctype html>, <html lang="en">, <head>, and <body>. Several parts of the code are highlighted with red boxes and arrows pointing to explanatory text boxes.

```
1  <!doctype html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport"
6          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
7      <meta http-equiv="X-UA-Compatible" content="ie=edge">
8      <title>Document</title>
9  </head>
10 <body>
11
12 </body>
13 </html>
```

Red box 1 (top left): Points to the 'lang' attribute of the <html> tag. The text says: '枠線で囲まれているブロックは、Enterを押すと順に変更できる領域です。' (The area enclosed in a frame can be changed sequentially by pressing Enter.)

Red box 2 (middle left): Points to the 'content' attribute of the first <meta> tag in the head. The text says: '適当で良いので変えたり、そのままEnterしたり、してみてください。' (It's fine to change it or press Enter directly.)

次は最初に書いたHTMLを思い出してみてください

```
<ul>  
  <li>1</li>  
  :  
  <li>12</li>  
</ul>
```

これ、簡単に出来るんじゃないの？

できます。

挿入したい場所に `ul>li*12` とタイプしてTabを押してみましょう

```
96 <body>
97     <div id="myid"></div>
98     <div class="myclass"></div>
99     ul>li*12
100    <div id="header">
```

うわ、一気に生成された！！

```
<ul>
  <li></li>
  <li></li>
</ul>
```

でも、白枠が間に表示されています。何でしょう？

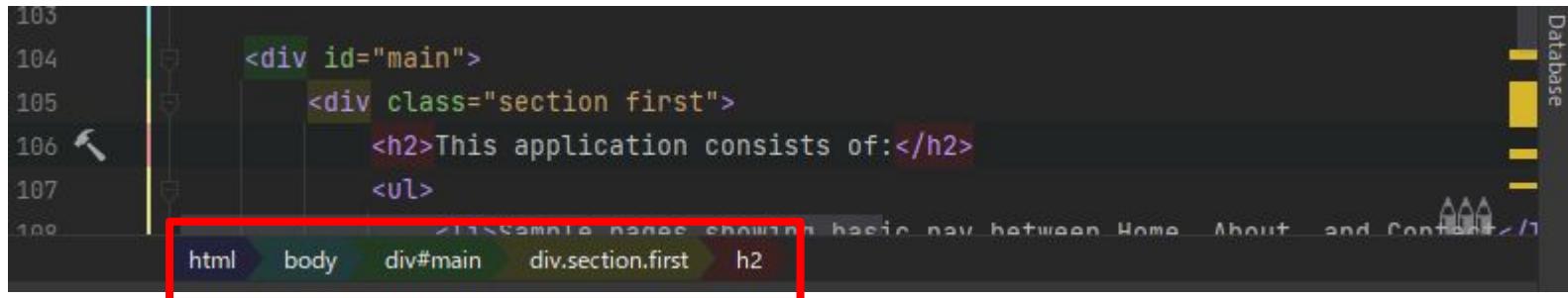
試しに 1 と入力してEnterを押すと…

賢く次の編集場所に移ってくれる！

```
<ul>
  <li>1</li>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
  <li></li>
</ul>
```

順番に入力したい内容だけタイプすれば完成できますね！

(おまけ) ネスト階層だって凄く見やすい



A screenshot of a code editor showing a snippet of HTML code. The code is as follows:

```
103
104     <div id="main">
105         <div class="section first">
106             <h2>This application consists of:</h2>
107             <ul>
108                 <li>...</li>
109             </ul>
110         </div>
111     </div>
112 
```

The status bar at the bottom of the editor shows the current selection path: `html > body > div#main > div.section.first > h2`. This path is highlighted with a red rectangular box.

HTMLやCSSを書いているヒトであれば、このありがたみが分かるはず！

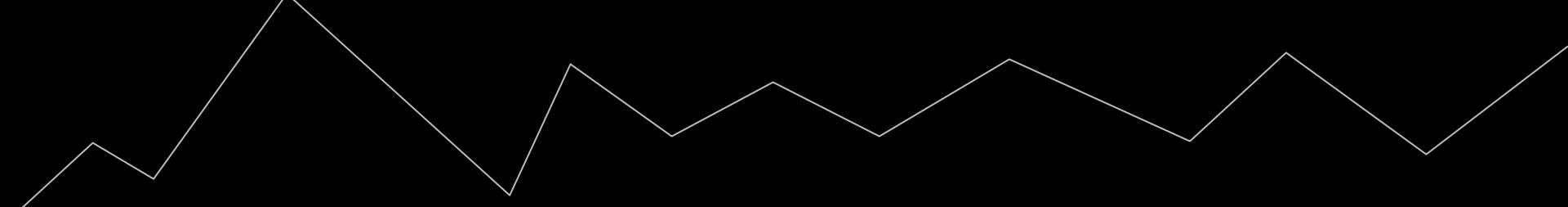
C#でも、同じようなコト出来るんじゃないの？

もちろん、できます。

C#ファイル内で以下タイプしてから Tab で補完してみてください。

- Exception
- psvm
- out
- (Main関数の中で) outv
- asrtn
- psr
- (GUIDが欲しい箇所で) nguid

もう、お気づきですね？
さあ、最初のコードを
もう一度書いてみましょう。



Hands On

#1.3 – Updateされた自分を知る





最初のコードを
もう一度
書いてください

覚えたTIPSで
どのくらい速くなる
でしょうか？



安心してください。
カンニングペーパー、出します。

さあ、やってみよう！ – Tab補完キーワード一覧

HTML		C#	
!	(HTMLのひな形生成)	psr	<code>public static readonly</code>
#hands-on-id	<code><div id="hand-on-id"></div></code>	psvm	<code>public static void Main(string[] args)</code>
.hands-on-class	<code><div class="hand-on-class"></code>	out	<code>System.Console.Out.WriteLine("");</code>
ul>li*12	<code> 1 2 : 11 12 </code>	asrtn	<code>System.Diagnostics.Debug.Assert(EXPR != null, "MESSAGE");</code>
		outv	<code>System.Console.Out.WriteLine("args = {0}", args);</code>
		Exception	(Exceptionクラスのひな型生成)
		/// (対象メソッドが完成後)	(コメント自動挿入)

仕事はテストではありません。暗記しないとダメという誤解は捨てましょう。

終わったら、次のようなことを書き出してみましょう。

- どのくらい速くなりましたか？
- 具体的に何倍でしたか？
- 10ksコードする仕事があった場合、何日早く終えられますか？
- ヒトによって改善率に差がありましたか？あったとしたら何故でしょう？

まとめ: 持って帰っていただきたい
3つのTIPS

困ったら【shift × 2】

で何でも検索

気軽に【Alt + Enter】

でおススメを確認

【Tab】

で自動補完(生成)する



(補足) ショートカットキーのチートシート、入ってます

The image shows the Rider IDE interface. On the left, the Explorer panel displays a project named 'ContosoUniversity'. The Help menu is open, with 'Keymap Reference' highlighted. A tooltip below the menu says 'Drop files here to open them'. On the right, a browser window shows a PDF titled 'Rider_default_win_shortcuts.pdf' from the JetBrains website, which lists various keyboard shortcuts for Rider.

印刷してデスクサイドに置きましょう！

Rider Default Keymap

CREATE AND EDIT

- Show context actions **Alt+F9**
- Locate code in file **Ctrl+Space**
- Smart code completion **Ctrl+Shift+Space**
- Mark current file in solution **Ctrl+Shift+M**
- Compare statements **Ctrl+Shift+I**
- Diff current file **Ctrl+Shift+D**
- Open and analyze documentation **Ctrl+Shift+Q**
- Generate code comments **Ctrl+Shift+C**
- Generate code comments for selected members **Ctrl+Shift+G**
- Standard with line / block comment **Alt+Shift+/**
- Comment / uncomment selection **Alt+Shift+Z**
- Comment / uncomment line **Alt+Shift+W**
- Comment / uncomment block **Alt+Shift+B**
- Auto indent lines **Ctrl+Alt+L**
- Copy document body **Ctrl+Shift+C, Ctrl+V**
- Copy document history **Ctrl+Shift+H**
- Copy document memory **Ctrl+Shift+M**
- Copy document outline **Ctrl+Shift+O**
- Move line up / down **Ctrl+Shift+Up / Ctrl+Shift+Down**
- Uncomment line **Ctrl+Shift+U**
- Jump / split in caret **Ctrl+Shift+J**
- Jump / split in caret **Ctrl+Shift+K**
- Toggle case **Ctrl+Shift+T**
- Expand / collapse code block **Ctrl+Shift+E**
- Expand / collapse all **Ctrl+Shift+D**

VERSION CONTROL

- XCode git status pop-up **Alt+Shift+G**
- Commit changes **Ctrl+Shift+C**
- Update project **Ctrl+Shift+U**
- Discard changes **Ctrl+Shift+D**
- New / review changes **Ctrl+Shift+N / Ctrl+Shift+R**

MASTER YOUR IDE

- Find action **Ctrl+Shift+F**
- Find and replace in project **Ctrl+Shift+F**
- Open a tool window **Alt+O**
- Tool windows **Ctrl+Shift+T**
- Quick switch scheme **Ctrl+Shift+Q**
- Configure scheme **Ctrl+Shift+P**
- Jump to source **Alt+Shift+J**
- Jump to last test window **F12**
- Last test window **Shift+F12**
- Go to next / previous editor tab **Alt+Shift+Left / Alt+Shift+Right**
- Close active tab / window **Ctrl+Shift+F4 / Ctrl+F4**

ANALYZE AND EXPLORE

- Report file **Ctrl+Shift+H+A**
- Code coverage description **Ctrl+Shift+H+C**
- Next / Previous highlight entry **Alt+F7 / Alt+Shift+F2**
- Locate code inspection in file **Ctrl+Shift+H+I**
- Run / debug selected configuration **Alt+F5 / Alt+Shift+F5**
- Go to file member **Ctrl+H**
- Go to file member **Ctrl+Shift+H**
- Go to file member **Ctrl+Shift+H+I**
- Go to file member **Ctrl+Shift+H+R**
- Go to file member **Ctrl+Shift+H+T**
- Go to file member **Ctrl+Shift+H+U**
- Go to file member **Ctrl+Shift+H+V**
- Go to file member **Ctrl+Shift+H+X**
- Go to file member **Ctrl+Shift+H+Y**
- Go to file member **Ctrl+Shift+H+Z**
- Go to file member **Ctrl+Shift+H+0**
- Go to file member **Ctrl+Shift+H+1**
- Go to file member **Ctrl+Shift+H+2**
- Go to file member **Ctrl+Shift+H+3**
- Go to file member **Ctrl+Shift+H+4**
- Go to file member **Ctrl+Shift+H+5**
- Go to file member **Ctrl+Shift+H+6**
- Go to file member **Ctrl+Shift+H+7**
- Go to file member **Ctrl+Shift+H+8**
- Go to file member **Ctrl+Shift+H+9**

REFACTOR AND CLEAN UP

- Refactor file **Ctrl+Shift+Shift+T**
- Copy file **Ctrl+Shift+C**
- Move file **Alt+Delete**
- Replace file **Shift+Alt+Ctrl+F5**
- Replace file **Shift+Alt+Ctrl+F6**
- Extract method **Ctrl+Shift+R**
- Introduce variable **Ctrl+Shift+V**
- Introduce constant **Ctrl+Shift+C**
- Introduce template **Ctrl+Shift+T**
- Introduce regular **Ctrl+Shift+R**
- Introduce code **Ctrl+Shift+I**

FIND EVERYTHING

- Find file / file history **Double Shift + F**
- Find file / file history **Ctrl+Shift+F**
- Find file in paths **Ctrl+Shift+F**
- Find / Previous occurrence **F3, Shift+F3**
- Find / Next occurrence **F3, Shift+F3**
- Find / All in caret **Alt+Shift+F3**
- Go to file member **Ctrl+H**
- Go to file member **Ctrl+Shift+H**
- Go to file member **Ctrl+Shift+H+I**
- Go to file member **Ctrl+Shift+H+R**
- Go to file member **Ctrl+Shift+H+T**
- Go to file member **Ctrl+Shift+H+U**
- Go to file member **Ctrl+Shift+H+V**
- Go to file member **Ctrl+Shift+H+X**
- Go to file member **Ctrl+Shift+H+Y**
- Go to file member **Ctrl+Shift+H+Z**
- Go to file member **Ctrl+Shift+H+0**
- Go to file member **Ctrl+Shift+H+1**
- Go to file member **Ctrl+Shift+H+2**
- Go to file member **Ctrl+Shift+H+3**
- Go to file member **Ctrl+Shift+H+4**
- Go to file member **Ctrl+Shift+H+5**
- Go to file member **Ctrl+Shift+H+6**
- Go to file member **Ctrl+Shift+H+7**
- Go to file member **Ctrl+Shift+H+8**
- Go to file member **Ctrl+Shift+H+9**

BUILD, RUN, DEBUG

- Build solution **Ctrl+Shift+B**
- Build configuration **Ctrl+Shift+B0**
- Run / Debug selected configuration **Alt+F5 / Alt+Shift+F5**
- Run / Debug current configuration **Shift+F5**
- Stop build **F7**
- Stop build **Shift+F7**
- Stop build **Alt+F5**
- Locate to cursor **Ctrl+Alt+H**
- Locate to cursor **Alt+Shift+H**
- Evaluate expression **Alt+H**
- Evaluate expression **Ctrl+Shift+H**
- Go to background processes **Ctrl+Shift+F**
- Logfile in the File Monitor **Ctrl+L**
- Logfile in the File Monitor **Shift+Ctrl+L**
- Logfile in the File Monitor **Alt+Shift+L**
- Logfile in the File Monitor **Ctrl+Shift+L**
- Logfile in the File Monitor **Shift+Alt+L**
- Logfile in the File Monitor **Ctrl+Shift+Alt+L**
- Logfile in the File Monitor **Shift+Ctrl+Alt+L**

Navigate from symbols

- Navigation to symbol **Ctrl+Shift+G**
- Navigation to symbol **Ctrl+Shift+H**
- Navigation to symbol **Ctrl+Shift+I**
- Navigation to symbol **Ctrl+Shift+L**
- Navigation to symbol **Ctrl+Shift+R**
- Navigation to symbol **Ctrl+Shift+T**
- Navigation to symbol **Ctrl+Shift+U**
- Navigation to symbol **Ctrl+Shift+V**
- Navigation to symbol **Ctrl+Shift+X**
- Navigation to symbol **Ctrl+Shift+Y**
- Navigation to symbol **Ctrl+Shift+Z**

Navigate context

- Select last focused / changed file **Alt+F1**
- Last edit location **Ctrl+Shift+Block**
- Next / previous / forward **Alt+Shift+Left / Alt+Shift+Right**
- Go to previous / next method **Alt+Shift+Up / Alt+Shift+Down**
- Go to previous / next class / struct **Alt+Shift+PageUp / Alt+Shift+PageDown**
- Add to favorites **Alt+Shift+F**
- Go to type definition **Ctrl+Shift+T**
- Go to type definition with immediate **Ctrl+Shift+I**
- Go to type definition with immediate **Ctrl+Shift+T**
- Show bookmarks **Ctrl+Shift+Q**

UNIT TESTS

- Last running quick fix **Alt+Shift+1**
- Last running quick fix **Alt+Shift+2**
- Stop execution **Ctrl+Shift+D**
- Stop execution **Ctrl+Shift+P**
- Stop execution **Ctrl+Shift+R**
- Stop execution **Ctrl+Shift+T**
- Stop execution **Ctrl+Shift+U**
- Stop execution **Ctrl+Shift+V**
- Stop execution **Ctrl+Shift+X**
- Stop execution **Ctrl+Shift+Y**
- Stop execution **Ctrl+Shift+Z**

jetbrains.com/rider @jetBrainsRider

A group of people are exercising outdoors in a park-like setting. In the foreground, a man in a black t-shirt and shorts is seen from behind, running towards a blue cylindrical obstacle. To his left, a woman in a black tracksuit walks away. On the right, two women in dark athletic wear are jogging. In the background, a man in a white t-shirt and dark pants runs towards the camera. The scene is set on a paved path with trees and buildings in the background.

さあ、使ってみよう！

THANKS

