www.takbook.com

راهنمای کامنت گذاری در برنامه نویسی

گرد آورنده : نرگس ساعدی

www.houzan.blogfa.com

www.takbook.com

فهرست مطالب

www.takbook.com

مقدمه

کامنت گذاری یکی از مواردی هست که تاحد زیادی به خوانایی کدها کمک میکنه . تصور کنید بعد از یکسال قصد اعمال تغییرات و یا به روز رسانی نرم افزاری رو دارید . بدون شک کامنت ها کمک قابل توجهی می کنن. منتهی اصولی برای نوشتن یک کد خوب و تمیز وجود داره.

همه می تونن کد بزنن ولی فقط یه برنامه نویس خوب می تونه کدهای تمیز و خوانا بنویسه. زمانی که در مرحله آموزش بودم ، هروقت به چنین مقالاتی برمی خوردم یا چنین نصیحت هایی رو می شنیدم پیش خودم فکر می کردم حساسیت های بی مورد. چرا بقیه دوست دارن استرس منتقل کنن ، برای من فقط این مهم بود کد اجرا بشه و خروجی مطلوب باشه. از این تصور هم دست بر نداشتم ، تا زمانی که اولین کارم بعد از یکسال برای اعمال یه سری تغییرات دوباره مورد بررسی قرار گرفت . یادمه زمان موردنیاز رو یک ماه اعلام کردیم، البته برای اون تغییرات یک ماه زمان خیلی زیادی بود، شاید نهایتا یک الی دو هفته زمان نیاز داشت.

وقتی پروژه رو اجرا کردم هیچی یادم نبود. کلیات رو به خاطر داشتم ولی جزئیات رو نه. متاسفانه مجبور شدم هرچی نوشته بودم پاک کنم و از نو شروع کنم.

متوجه شدم که اون نصایح و توصیه ها بی مورد نبودن بلکه فقط شخصی که قبلا این رو تجربه کرده بود می خواست تجربیاتش رو در اختیار من بذاره که قبول نکرده بودم و خب ضررش رو هم دیدم. ضررش یک ماه کار بی وقفه و مداوم برای اعمال چندتا تغییر خیلی جزئی بود.

از اون به بعد شروع به مطالعه این دست مقالات کردم چون الان تاثیر و اهمیتشون رو متوجه شدم ، همچنین شروع کردم به گردآوری و انتشارشون در قالب متن های کوتاه ، فایل های pdf کوتاه و ....

www.takbook.com

امیدوارم که براتون مفید واقع بشه و باعث بشه مجبور نباشید یه کد رو کلا پاک کنید و از اول بنویسید .

موفق باشید.

www.takbook.com

کامنت چیست ؟

کامنت ها توضیحاتی هستند که جهت درک بهتر کد میان کدها نوشته می شوند. اگرچه مورد استفاده کامنت به منظور افزایش خوانایی است اما درصورتی که به درستی استفاده نشوند خود می توانند باعث کاهش خوانایی و همچنین شلوغی و نامرتبی کد باشند.

درواقع تنها زمانی باید از کامنت استفاده کرد که کد نوشته شده خود قادر به توضیح عملکرد خود نباشد. توجه داشته باشید ممکن است از نظر برنامه نویس کد مرتب و خوانا باشد ولی از نظر خواننده کد اینگونه نباشد.

www.takbook.com

در چه مواردی نباید از کامنت استفاده کنیم؟

- Redundant Comments : این نوع کامنت ها توضیح اضافی نیستند بلکه همان کد را تکرار کرده اند. برای مثال:

```
/// <summary>
/// Default Constructor
/// </summery>
Public User(){
}
```

این کامنت ها باعث افزایش حجم کد و شلوغی آن شده درحالی که ضرورتی ندارند و مطلب تازه ای به خواننده کد اضافه نمی کنند.

- Intent Comments : این کامنت ها غیر ضروری اند چون با اصلاحاتی در کد و بهتر نوشتن کد ، نیازی به این کامنت ها نیست . برای مثال :

```
// Assure user's account is deactivated
If(user.status == 2 )
```

درحالی که این قسمت می تواند توسط یک enum پیاده سازی شود ، که هم خوانایی کد را افزایش داده و هم مدیریت تغییرات را ساده تر می کند.

www.takbook.com


```
If(user.status == status.Inactive){


}
```

- Apology Comments : کامنت هایی که برنامه نویس جهت اصلاح کد نوشته شده خطاب به خواننده دوم می نویسد. برای مثال:

```
// I was too tired to refactor this ….
```

این کامنت ها نامناسب اند چون کمکی به شخص دوم نمی کنند. دو رویکرد داریم:

۱. رفع مشکل کد قبل از تحویل یا به اشتراک گذاری
۲. استفاده از کامنت نشانه گذاری شده با لفظ ToDo

- Zombie Code : این کامنت ها مربوط به کدهایی هستند که منقضی شده و کاربردی در کد فعلی ما ندارند، پس کدهای غیر ضروری باید حذف شوند.

- Divider Comments : کامنت هایی که برای ایجاد یک سکشن استفاده می شوند. برای مثال :

```
// start search for …..
.
.
```

www.takbook.com

.

// end of search for

چنین کامنت هایی نیز غیر ضروری اند درصورتی که نیاز به ایجاد سکشن داریم می توان از دستورات مربوطه باتوجه به IDE استفاده کرد.

- Brace Tracker Comments : کامنت هایی که در انتها یا ابتدای بلوک اضافه می کنیم. برای مثال :

```
If ( شرط ){
    // دستورات
} // end of Condition
```

- Bloated header Comments : کامنت هایی که توضیحات اضافی راجع به فایل برنامه می دهند.

```
// *****
// file name : appp
// Author : john
.
.
// ****
```

www.takbook.com

این کامنت ها اغلب توضیحاتی راجع به برنامه اند و در بالای فایل تعریف می شوند، که کاری نامناسب و غیر ضروری است. نام فایل ، نویسنده ، تاریخ و... باتوجه به فایل قابل تشخیص بوده و نیاز به توضیحات اضافی درون فایل برنامه نیست.

- Defect Log : توضیحات راجع به نحوه عملکرد برنامه را شامل می شود که غیر ضروری است. برای مثال :

```
// defect #5274 DA 12/10/2010
// we checking for null here
If( شرط ){

}
```

www.takbook.com

شکل صحیح کامنت ها

- برای کامنت کردن توضیحاتی راجع به کدهایی که بعدا باید تکمیل یا اصلاح شوند از ToDo استفاده می کنیم برای مثال

```
// ToDo refector out duplication
```

- Summery Comment : اگر نام کلاس به تنهایی منعکس کننده وظایف کلاس نیست ، می توان به طور خلاصه ای کوتاه یک دید کلی از عملکرد کلاس را در ابتدای آن درج کرد . برای مثال

```
// encapsulates logic for calculating
```

این مورد در رابطه با توابع نیز صادق است.

- Documentation Comments : فرض کنیم از کدی استفاده کرده ایم که خواندن توضیحات بیشتر درباره آن به خواننده کمک می کند، در این حالت می توان به صورت زیر کامنت گذاری کرد

```
// see www…… . com for documentation
```