آموزش برنامه نویسی شی گرا در سی شارپ (C#) قسمت ۵: سازنده ها (نسخه PDF)

در قسـمت قبلی آموزش در مورد خصوصیات یا Property ها و نحوه صـحیح اسـتفاده از آنها در کلاس ها صحبت کردیم. در این بخش در مورد سازنـده ها یا Constructors، مقدار دهی اولیه اشـیاء (Object Initialization) و نوع های بـدون نام (Anonymous Types) صحبت می کنیم.زمانی که شـما کلاسی را تعریف می کنید، این کلاس حاوی یکسری خصوصیات و یکسری رفتارها یا همان متدها می باشد. شما بعد از ایجاد شـئ، خصوصـیات را مقدار دهی کرده و از شئ استفاده می کنید. اما چندین راه دیگر برای مقدار دهی خصوصیات و آماده سازی اولیه کلاس وجود دارد. ما در این قسمت دو روش مختلف را بررسی می کنیم:

۱. مقدار دهی اولیه شئ یا Object Initialization
۲. استفاده از سازنده ها یا Constructors

مقدار دهی اولیه با کمک Object Initialization

در این روش، شـما زمانی که اقـدام به ایجاد یک شـئ می کنیـد، می توانیـد مقادیر خصوصـیات و فیلـدهای آن را مشـخص کنید. کلاس Person را در نظر بگیرید:

```
public class Person
{
    public string FirstName;
    public string LastName;
}
```

به صورت پیش فرض، شما یک شئ از کلاس ساخته و خصوصیات آن را مقدار دهی می کنید:

```
var person = new Person();
person.FirstName = "Hossein";
person.LastName = "Ahmadi";
```

مقدار دهی اولیه شئ کار ساده ایست، کافیست پس از نوشتن () بعد از نام کلاس در قسمت new بین علامت های {} مقادیر خصوصیات را مشخص کنیم:

```
var person = new Person()
{
    FirstName = "Hossein",
    LastName = "Ahmadi"
};
```

با این کار مقادیر FirstName و LastName در کلاس Person مقدار دهی اولیه خواهند شد. می توانید در این حالت، از نوشـتن () صرفنظر کنید:

```
var person = new Person
{
    FirstName = "Hossein",
```

```
    LastName = "Ahmadi"
};
```

دقت کنیـد، در هنگام مقـدار دهی اولیه قابلیت صدا زدن متدهای کلاس را نخواهید داشت و تنها می توانید فیلدها، خصوصـیات و برخی اعضای دیگر که در قسمت های بعدی با آن آشنا خواهیم شد را مقدار دهی کنید.

سازنده ها یا Constructors

اگر دقت کرده باشید، زمانی که شئ ای را تعریف می کنیم، بعد از نوشتن نام کلاس بعد از کلمه new از () استفاده می کنیم، مشابه زمانی که تصـمیم به صدا زدن یک متد دارید. دلیل اینکار، پروسه ایست که سی شارپ برای ایجاد کردن کلاس ها انجام می دهد. زمانی که شما شئ ای از یک کلاس ایجاد می کنید، سی شارپ قسمتی با نام سازنده یا Constructor را برای آن کلاس صدا می زند. این سازنده یک متد می باشـد که می تواند بدون پارامتر یا با پارامتر باشد و داخل آن کدی نوشـته می شود که می خواهیم در هنگام ایجاد شـئ اجرا شود. با یک مثال ساده سازنـده ها را بررسـی می کنیم. کلاس Person را در نظر بگیریـد، برای این کلاـس یک سازنـده تعریف می کنیم که مقادیر FirstName و LastName را به عنوان ورودی گرفته و خصوصیات مربوطه را مقدار دهی می کند:

```
public class Person
{
    public Person(string firstName, string lastName)
    {
        this.FirstName = firstName;
        this.LastName = lastName;
    }

    public string FirstName { get; set; }
    public string LastName { get; set; }
}
```

در کد بالا، قسمت سانده برای کلاس Person با دو پارامتر تعریف شده است. به شـیوه تعریف سازنده دقت کنید، ابتدا سطح دسترسی به سازنده مشـخص شده، سپس نام کلاس نوشته شده که برای سازنده ها، این نام دقیقاً باید معادل نام کلاس باشد، سپس پارامترهای مورد نظر و بعـد از آن ها بـدنه سازنده. دقت کنید سازنده ها مقدار بازگشتی ندارند. با توضـیحات گفته شده می توان ساختار کلی سازنده را به صورت زیر بیان کرد:

```
{access-modifier} {class-name}([parameters])
{
    // constructor body
}
```

همچنین در بدنه سازنده بالا، به کلمه کلیدی this دقت کنید. کلمه کلیدی this به شئ جاری که روی کلاس ساخته شده است اشاره می کند. فرض کنید شـما ده ها شـئ از روی یک کلاس ساخته اید، زمانی که یک رفتار را صدا می زنید و داخل آن رفتار از کلمه کلیدی this استفاده می کنید، this به همان شـئ ای اشاره می کند که رفتار در آن صدا زده شده است. در این سازنده نیز کلمه کلیدی this به شئ ای اشاره می کند که سازنده برای آن صدا زده شده.پس از تعریف سازنده می توان هنگام ایجاد شئ، مقادیر مورد نظر را به سازنده ارسال کرد:

```
var person = new Person("Hossein", "Ahmadi");
```

با اجرای کد بالا، خصوصیت های FirstName و LastName هنگام ایجاد شئ مقدار دهی خواهند شد. اما باید به یک نکته در اینجا توجه

با اجرای کد بالا، خصوصیت های FirstName و LastName هنگام ایجاد شیء مقدار دهی خواهند شد. اما باید به یک نکته در اینجا توجه داشت، زمـانی که شـما سازنـده ای به همراه پارامتر برای یک کلاس تعریف می کنیـد، دیگر نمی توانیـد از کلاس بـدون ارسال پارامتر در سازنـده شـیء بسازیـد. دلیل این موضوع، عدم وجود سازنده ای به نام سازنده پیش فرض یا Default Constructor می باشد. سازنده پیش فرض، سازنـده ایست که هیـچ پـارامتری را به عنوان ورودی نمی گیرد. زمانی که شـما سازنـده ای برای یک کلاس تعریف نکرده ایـد، آن کلاس به صورت پیش Default Constrcutor برایش تعریف شـده است. اما زمانی که اقدام به ایجاد یک سازنده برای کلاس کردید، اگر می خواهید از آن کلاس بدون ارسال پارامتر برای سازنده شیء بسازید، باید سازنده پیش فرض را به صورت دستی برای آن کلاس بنویسید:

```
public class Person
{
    public Person()
    {
    }

    public Person(string firstName, string lastName)
    {
        this.FirstName = firstName;
        this.LastName = lastName;
    }

    public string FirstName { get; set; }
    public string LastName { get; set; }
}
```

شما می توانیـد سـطح دسترسـی به سازنـده ها را مشـخص کنید، برای مثال، حالتی پیش می آید که می خواهد یک سازنده فقط داخل همان کلاس در دسترس باشـد، یعنی شـما از یک کلاس داخل خودش، مثلاً داخل یک رفتار، می خواهید یک شـیء بسازید. برای این کار، می توانید سطح دسترسی سازنده مد نظر را private تعریف کنید. برای مثال:

```
public class Person
{
    public Person()
    {
    }

    private Person(string firstName, string lastName)
    {
        this.FirstName = firstName;
        this.LastName = lastName;
    }

    public Person CreateObject(string firstName)
    {
        return new Person(firstName, null);
    }
```

```
    public string FirstName { get; set; }
    public string LastName { get; set; }
}
```

در کد بالا، یک متد یا رفتار برای کلاس تعریف کردیم با نام CreateObject. این رفتار یک شئ از روی خود کلاس Person می سازد و پارامتر ارسالی به متد CreateObject را به سازنده ارسال می کند. اما خارج از شئ، دیگر نمی توانیم از سازنده ای که دو پارامتر را به عنوان ورودی می گیرد استفاده کنیم، زیرا این سازنده با سطح دسترسی private تعریف شده است.همانند متدها، سازنده ها می توانند overload داشته باشند، یعنی چند سازنده با signature های متفاوت. برای مثال، کلاس Person را به صورت زیر تغییر می دهیم:

```
public class Person
{
    public Person()
    {
    }

    public Person(string firstName)
    {
        this.FirstName = firstName;
    }

    public Person(string firstName, string lastName)
    {
        this.FirstName = firstName;
        this.LastName = lastName;
    }

    public string FirstName { get; set; }
    public string LastName { get; set; }
}
```

در اینجا دو سازنده برای کلاس تعریف کردیم که اولی یک پارامتر گرفته و دومی با دو پارامتر صدا زده می شود.

زنجیره سازنده ها یا Constrcutor Chaining

زمانی که شما چندین سازنده دارید، می توانید از کد های نوشته شده داخل یک سازنده در سازنده دیگر استفاده کنید. برای اینکار از قابلیت constructor chaining استفاده می شود. با یک مثال ادامه می دهیم، در کد قبلی سه سازنده داشتیم، سازنده پیش فرض، سازنده ای که تنها FirstName را می گرفت و سازنده ای که FirstName و LastName را به عنوان پارامتر می گرفت. در سازنده دوم، می توان از سازنده سوم جهت مقدار دهی استفاده کرد. برای این کار، کد بالا را به صورت زیر تغییر می دهیم:

```
public class Person
{
    public Person()
    {
    }
```

```
    public Person(string firstName) : this(firstName,null)
    {
    }

    public Person(string firstName, string lastName)
    {
        this.FirstName = firstName;
        this.LastName = lastName;
    }

    public string FirstName { get; set; }
    public string LastName { get; set; }
}
```

سازنده دوم ما به صورت زیر تغییر کرده است:

```
public Person(string firstName) : this(firstName,null)
{
}
```

دقت کنید، بعد از بستن پرانتز پس از علامت : از کلمه کلیدی this مانند یک متد استفاده کرده ایم، در این روش، سازنده کلاس صدا زده شده و به عنوان پارامتر اول، firstName که در سازنده تعریف شده را ارسال کرده و عنوان پارامتر دوم مقدار null را ارسال کرده ایم. قابلیت constrcutor chaining، در کاهش تعـداد خطوط نوشـته در برنامه کمک زیادی به ما می کند.به این نکته توجه داشـته باشـید که نمی تواند سازنـده را به صورت متـد از داخـل کلاس جایی غیر از خود سازنـده ها صـدا زد. همچنین سازنـده ها تنها برای مقـدار دهی خصوصـیات استفاده نمی شوند، شما می توانید هر کدی را داخل سازنده بنویسید.

نوع های بدون نام یا Anonymous Types

نوع های بدون نام، به ما این امکان را می دهند تا شئ ای بدون تعریف کلاس ایجاد کنیم. این شئ تنها می تواند شامل خصوصیات باشد و قـابلیت تعریف رفتـار برای آن را نخواهیم داشت. در مثال زیر یک شـئ بـدون نام ایجاد کرده ایم که سه خصوصـیت با نام FirstName و LastName و Age دارد:

```
var anonymous = new
{
    FirstName = "Hossein",
    LastName = "Ahmadi",
    Age = 29
};

Console.WriteLine(anonymous.FirstName + " " + anonymous.LastName);
```

همانطور که در کـد مشاهـده می کنیـد، کافیست بعد از کلمه کلیدی new بلافاصله به سـراغ عملیات Object Initialization برویم و نیازی به نوشـتن نام کلاس نیست. نوع های بـدون نام کاربردهای زیادی در بخش LINQ دارند که در همین وب سایت دوره آموزشـی LINQ توسط بنـده نوشـته شـده و در ایـن دوره آموزشـی نیز مروری کوتـاه بر این قـابلیت خـواهیم داشت. در بخش بعـدی آمـوزش، مبحث وراثت یا

Inheritance که مهمترین مبحث در زمینه برنامه نویسی شئ گرا می باشد را آغاز خواهیم کرد. **ITPRO باشید**

نویسنده : حسین احمدی

منبع : جزیره برنامه نویسی وب سایت توسینسو

#آموزش_زبان_سی_شارپ #برنامه_نویسی_شئ_گرا #ایجاد_object_در_c# #آموزش_ایجاد_شیء_در_سی_شارپ
#آموزش_برنامه_نویسی_شئ_گرا #برنامه_نویسی_شئ_گرا_چیست؟ #سازنده_ها_در_برنامه_نویسی_شئ_گرا
#آموزش_برنامه_نویسی #مقدار_دهی_اولیه_اشیاء_در_سی_شارپ #نوع_های_بدون_نام_در_سی_شارپ

مطلب اصلی