فصل یک

راهنمایی بر اسمارت کتل

فصل دو

معماری اسمارت کتل

مقدمه

در این فصل در رابطه با معماری بخش های مختلف اسمارت کتل صحبت خواهد شد. نرم افزار اسمارت کتل برای تغذیه داده های خود، نیاز دارد تا به سرور اسمارت کتل وصل شود. شیوه برقراری ارتباط با سرور اسمارت کتل در فصول بعدی شرح داده خواهد شد. داده ها پس از دریافت از سرور اسمارت کتل به کاربر نمایش داده می شوند. شکل زیر در مورد معماری دریافت داده اسمارت کتل توضیحاتی را بیان می کند.

Farm sensor Data

SmartCattle Listener

Sql server

IPARS

MongoDB

RabbitMQ

SmartCattle Software

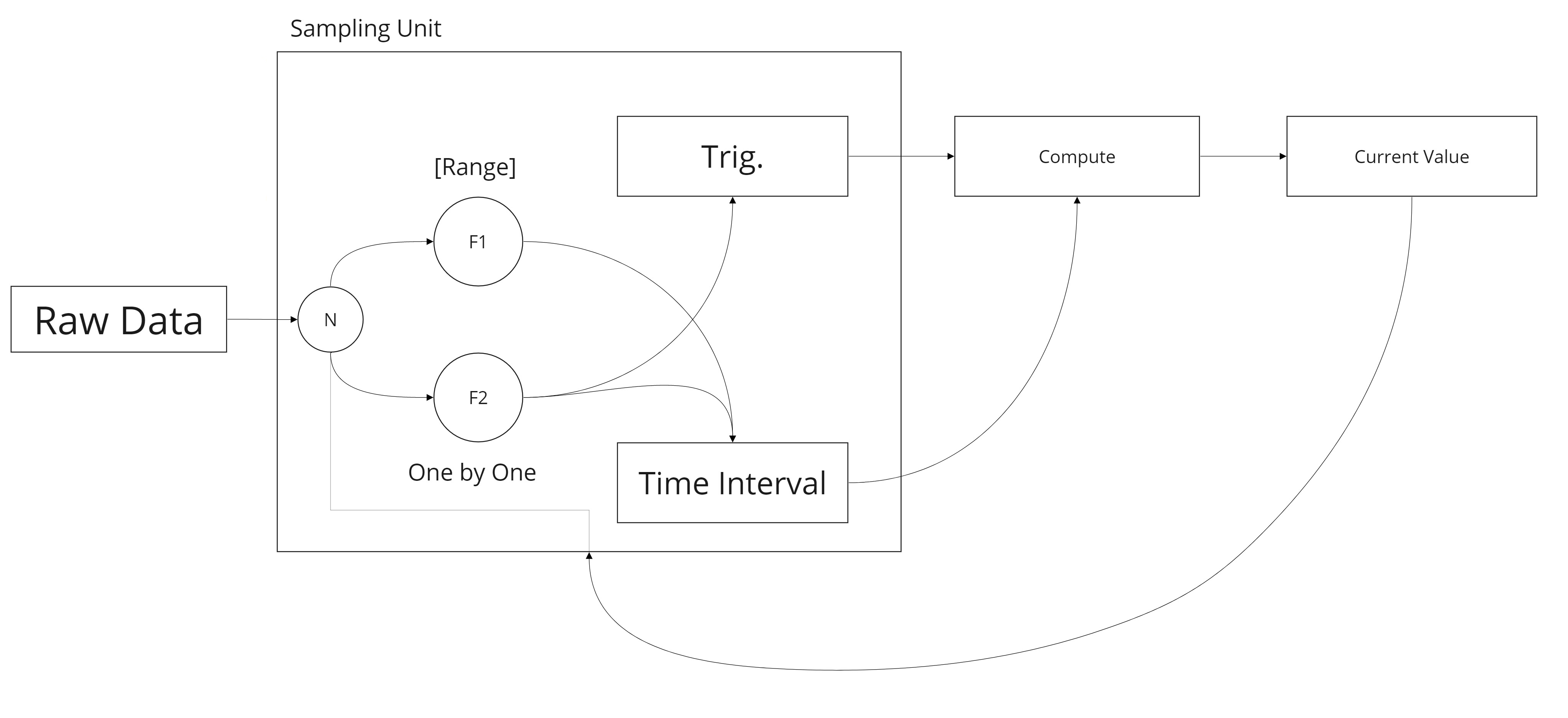
SmartCattle Services

همانطور که در تصویر بالا مشخص شده است، داده ها در سرور آی پارس در یک صفی قرار می گیرند. برای هر فارم یا به عبارتی برای هر مشتری مستقل، صف جدایی تعریف می شود و داده های مربوط به سنسورهای محیطی، دمای بدن دام و فعالیت های دام در داخل آن ریخته می شود. Listener اسمارت کتل به محض آن که داده ای در صف قرار گیرد متوجه آن می شود و اطلاعات را خوانده و در دیتابیس ذخیره می کند. در ادامه هر زمان که کاربر وارد نرم افزار اسمارت کتل شد، داده های مورد نیازش را با درخواست شناسه ها و پارامتر های مشخصی دریافت می کند. در مورد پارامترهای مورد نیاز برای فراخوانی داده ها از فصل مربوط به توضیح اکشن های اسمارت کتل به صورت مفصل توضیح داده خواهد شد. همچنین در فصل مربوط به توضیح API ها، پارامترهای مورد نیاز برای فراخوانی سرویس های اسمارت کتل به صورت مفصل بررسی خواهند شد.

----Break----

معماری نونیفیکیشن

در شکل زیر ساختار کلی نوتیفیکیشن نمایش داده شده است. در این ساختار داده ها در ابتدا وارد واحد نمونه برداری می شود. در این واحد دو فیلتر f1 و f2 وجود دارد که یکی روی یک بازه و دومی روی یک فیلد می باشد.



دو حالت برای انجام محاسبات روی داده های ورودی وجود دارد. در حالت اول Trig، زمانی رخ می دهد که یک تغییر در یک فیلد مشخص از داده اتفاق بیفتند. این تغییر می تواند یکی از دو عمل Update, Insert باشد. به عنوان مثال وقتی داده جدیدی برای دمای بدن دام ثبت می شود یک تریگ رخ می دهد و داده مربوطه به واحد محاسبات ارسال می شود.

در حالت دوم، داده های ورودی در یک Time Interval از پیش تعیین شده ای جمع آوری می شوند و به واحد محاسبات ارسال می شوند.

با توجه به این که واحد محاسبات به عنوان یک ماژول مستقل طراحی و پیاده سازی شده، می توان انواع محاسبات را برای بررسی رفتارهای مختلف سیستم در آن تعریف کرد. به عنوان مثال می توان هر ده دقیقه یک بار داده های دمای بدن دام یک بهاربند را جمع کرد و از آن ها میانگین گرفت. سپس نتیجه در CurrentValue با یک Dynamic Label ذخیره می شود. یعنی اگر از همین نوع دیتا قبلا در جدول ذخیره شده باشد، مقدار جدیدی Insert نمی شه و مقدار قبلی Update می شه.

بر روی جدول CurrentValue یک تریگر تعریف شده که هر بار مقدار جدیدی در آن قرار می گیرد یا سطری آپدیت می شود، آن تریگر فعال می شود و مقادیر درج شده را با تنظیماتی که در جدول نوتیفیکیشن درج شده است مقایسه می کند. اگر مقدارهای جدید Over Range باشند، یک نوتی جدید ایجاد و در جدول نوتیفیکیشن ها ذخیره می شود و همزمان با آن، یک پیام از طریق سوکت به نرم افزار اسمارت کتل ارسال می شود. اگر کاربر در نرم افزار حضور داشته باشد، این پیام را به صورت یک پاپ آپ مشاهده می کند. در غیر این صورت هر بار که به نرم افزار لاگین شود در بالا سمت راست می تواند لیست نوتی های جدید را مشاهده کند.

فصل سه

شرح API ها

مقدمه

در این فصل در مورد API های پیاده سازی شده به طور مفصل شرح داده خواهد شد. ساختار توضیح داده شده به این صورت است که ابتدا آدرس API به همراه پروتکل ارتباطی آن بیان می شود. به عنوان مثال: POST: <http://www.domainname.com> یعنی که که پروتکل ارتباطی مورد نیاز برای برقراری ارتباط با این API پست می باشد که بر روی آدرس مذکور فعال هست.

در انتهای URL سرویس شرح داده شده ابتدا SubDirectory های سرویس و بعد از آن پارامترهایی که باید برای آن ارسال شود درج می شود. پارامترها برای آن که از SubDirectory ها مجزا شوند، در داخل آکولاد قرار داده شدند. پس از بیان ساختار سرویس ها، عملکرد آن و پارامترهای مورد استفاده شرح داده می شوند. توضیحات مربوط به پیاده سازی سرویس ها در فصل 4 بیان شده اند.

|  |
| --- |
| POST: http://79.175.133.194/SmartCattle/APIs/EnvSensors/List/{FarmId}/{FreeStallid} |
| از این API برای دریافت لیست سنسورهای یک بهاربند خاص استفاده می شود. |
| FarmId  شناسه فارم جاری |
| FreeStalId  شناسه بهاربندی که می خواهیم سنسورهای محیطی آن را استخراج کنیم. |

|  |
| --- |
| POST: http://localhost:5763/EnvSensors/THIValue |
| دریافت داده های مربوط به نمودار THI و رطوبت و دما با استفاده از وب سرویس میسر می شود. خروجی این سرویس سه پارامتر فیزیکی شامل دما، رطوبت و THI را بر می گرداند. |
| {  "FarmId": 2,  "FreeStallId": 32,  "StartTime": "2017-12-08 11:40",  "EndTime":"2017-12-08 11:41",  "MACs":  [  {  "Address":"d1:13:a3:c5:d1:be",  "Step": 2  },  {  "Address":"e3:ad:eb:31:e0:ef",  "Step": 20  }  ]    } |
| FarmId  شناسه فارم |
| FreeStalId  شناسه بهاربند |
| StartTime  زمان آغاز داده های جمع آوری شده |
| EndTime  زمان پایان داده های جمع آوری شده |
| MACs  پارامتر فوق یک لیست از مک آدرس ها است. بدین معنی که می توان به عنوان مثال دو مک را در یک ریکوئست ارسال کرد و به ازای هر یک داده ها را در یک Response به صورت جداگانه و json دریافت کرد. |
| Address  شماره مکی که قصد دریافت داده از آن را داریم. |
| Step  توالی زمانی داده ها را مشخص می کند. |

|  |
| --- |
| POST: http://79.175.133.194/APIs/CattleSensors/Activity |
| دریافت داده های مربوط به اکتیویتی دام |
| {  "FarmId": 3,  "CattleId": 2033,  "StartTime": "2017-12-20 09:07",  "EndTime": "2017-12-25 21:07",  "Step": 2  } |
| FarmId  شناسه فارم |
| CattleId  شناسه دامی که می خواهیم اکتیویتی آن را استخراج کنیم. |
| StartTime  زمان آغاز داده های جمع آوری شده |
| EndTime  زمان پایان داده های جمع آوری شده |
| Step  توالی زمانی داده ها را مشخص می کند. |

|  |
| --- |
| POST: http://79.175.133.194/APIs/EnvSensors/List |
| با استفاده از این API می شه متوجه شد که برای یک بهاربند چندتا سنسور محیطی نصب شده است. |
| {  "FarmId" : 3,  "FreeStallId": 5  } |
| FarmId  شناسه فارم |
| FreeStalId  شناسه بهاربند |

فصل سه

ساختار دیتابیس

مقدمه

در این فصل در مورد ساختار دیتابیس توضیح داده خواهد شد. شیوه توضیح به این صورت است که ابتدا نام تمامی جداولی که در پیاده سازی این پروژه مورد استفاده قرار گرفته بیان می شود و در مورد هر یک به صورت مختصر توضیحی درج می شود. سپس در ادامه فیلدهای هر یک از جداول به همراه کوئری که برای ساخت آن نیاز است بیان می گردد.

لیست جدول های دیتابیس

در جدول زیر لیست جدول های مورد استفاده برای پروژه اسمارت کتل را مشاهده می کنید.

|  |  |  |
| --- | --- | --- |
| ID | Name | Comment |
| 1 | EnvSensors | لیست سنسورهای محیطی برای اندازه گیری دما و رطوبت |
| 2 | ActivityStateTbl | فعالیت دام ها نظیر نشستن، راه رفتن، نشخوار کردن و غیره |
| 3 | CattleScoreTbl | ثبت امتیاز های هر دام |
| 4 | TempretureTbl | دمای بدن دام ها |
| 5 | SensorTbl | سنسورهای وصل شده به دام برای تشخیص فعالیت و دمای بدن |
| 6 | CattlePositionTbl | موقعیت مکانی دام ها |
| 7 | CattleTbl | لیست دام هایی که برای یک دام ثبت می شوند. |
| 8 | CattleGroupTbl | گروه بندی دام ها |
| 9 | EnvTHITbl | داده های سنسور محیطی شامل دما و رطوبت و میزان THI در این جدول ذخیره می شوند. |
| 10 | FarmTbl | لیست فارم هایی که درBackOffice ساخته شده اند. این لیست به Email هایی تخصیص داده می شوند. همچنین شناسه فارم ها برای لینک با سایر جدول ها استفاده می شود. |
| 11 | FreeStallTbl | لیست بهاربندها |
| 12 | UserInfo | داده های مربوط به کاربرا در این بخش ذخیره می شود. |
| 13 | UserPermissions | دسترسی های تعریف شده در برنامه |
| 14 | UserRoles | تعریف نقش ها |
| 15 | Notifications | نوتیفیکیشن هایی که تولید می شوند در این جا ذخیره می شوند |
| 16 | FreeStallNotificationsSetting | تنظیمات مربوط به نوتیفیکیشن بهاربندها |
| 17 | CattleNotificationsSetting | تنظیمات مربوط به نوتیفیکیشن دام ها |

در ادامه تمامی کدهای مورد نیاز برای تولید جداول درج شده اند. همانطور که می دانید برای ساختن این جداول نیاز هست تا تک تک فیلدها در SQL تعریف شوند. ولی از این پس فقط با دانستن کدهای زیر و اجرای کردن هر کدام می توان، تمامی جدول ها را خیلی سریع ساخت.

جدول EnvSensors

|  |
| --- |
| USE [smartCattle]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [SmartCattle].[EnvSensors](  [id] [int] IDENTITY(1,1) NOT NULL,  [FreeStallId] [int] NULL,  [FarmId] [int] NULL,  [Lat] [float] NULL,  [Lng] [float] NULL,  [MAC] [nvarchar](50) NULL  ) ON [PRIMARY]  GO |

جدول ActivityStateTbl

|  |
| --- |
| USE [smartCattle]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [SmartCattle].[ActivityStateTbl](  [ID] [int] IDENTITY(1,1) NOT NULL,  [jsonedActivities] [nvarchar](max) NULL,  [Sitting] [decimal](18, 2) NOT NULL,  [Standing] [decimal](18, 2) NOT NULL,  [Walking] [decimal](18, 2) NOT NULL,  [Eating] [decimal](18, 2) NOT NULL,  [Rumination] [decimal](18, 2) NOT NULL,  [Drinking] [decimal](18, 2) NOT NULL,  [cattleId] [int] NOT NULL,  [date] [datetime2](7) NOT NULL,  [FarmID] [int] NOT NULL,  [LastRecievedId] [bigint] NOT NULL,  [UserId] [nvarchar](max) NULL,  CONSTRAINT [PK\_SmartCattle.ActivityStateTbl] PRIMARY KEY CLUSTERED  (  [ID] ASC  )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  ) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]  GO  ALTER TABLE [SmartCattle].[ActivityStateTbl] WITH CHECK ADD CONSTRAINT [FK\_SmartCattle.ActivityStateTbl\_SmartCattle.CattleTbl\_cattleId] FOREIGN KEY([cattleId])  REFERENCES [SmartCattle].[CattleTbl] ([ID])  ON DELETE CASCADE  GO  ALTER TABLE [SmartCattle].[ActivityStateTbl] CHECK CONSTRAINT [FK\_SmartCattle.ActivityStateTbl\_SmartCattle.CattleTbl\_cattleId]  GO  ALTER TABLE [SmartCattle].[ActivityStateTbl] WITH CHECK ADD CONSTRAINT [FK\_SmartCattle.ActivityStateTbl\_SmartCattle.FarmTbl\_FarmID] FOREIGN KEY([FarmID])  REFERENCES [SmartCattle].[FarmTbl] ([ID])  GO  ALTER TABLE [SmartCattle].[ActivityStateTbl] CHECK CONSTRAINT [FK\_SmartCattle.ActivityStateTbl\_SmartCattle.FarmTbl\_FarmID]  GO |

جدول CattleScoreTbl

|  |
| --- |
| USE [smartCattle]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [SmartCattle].[CattleScoreTbl](  [ID] [int] IDENTITY(1,1) NOT NULL,  [item] [int] NOT NULL,  [value] [float] NOT NULL,  [CattleId] [int] NOT NULL,  [Date] [datetime2](7) NOT NULL,  [UserId] [nvarchar](max) NULL,  CONSTRAINT [PK\_SmartCattle.CattleScoreTbl] PRIMARY KEY CLUSTERED  (  [ID] ASC  )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  ) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]  GO  ALTER TABLE [SmartCattle].[CattleScoreTbl] WITH CHECK ADD CONSTRAINT [FK\_SmartCattle.CattleScoreTbl\_SmartCattle.CattleTbl\_CattleId] FOREIGN KEY([CattleId])  REFERENCES [SmartCattle].[CattleTbl] ([ID])  ON DELETE CASCADE  GO  ALTER TABLE [SmartCattle].[CattleScoreTbl] CHECK CONSTRAINT [FK\_SmartCattle.CattleScoreTbl\_SmartCattle.CattleTbl\_CattleId]  GO |

جدول TempretureTbl

|  |
| --- |
| USE [smartCattle]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [SmartCattle].[TempretureTbl](  [ID] [int] IDENTITY(1,1) NOT NULL,  [value] [float] NOT NULL,  [point] [nvarchar](max) NULL,  [cattleId] [int] NOT NULL,  [date] [datetime2](7) NOT NULL,  [LastRecievedId] [bigint] NOT NULL,  [FarmID] [int] NOT NULL,  [UserId] [nvarchar](128) NULL,  [FreeStall] [int] NULL,  [dateStr] [nvarchar](max) NULL,  CONSTRAINT [PK\_SmartCattle.TempretureTbl] PRIMARY KEY CLUSTERED  (  [ID] ASC  )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  ) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]  GO |

جدول SensorTbl

|  |
| --- |
| USE [smartCattle]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [SmartCattle].[SensorTbl](  [ID] [int] IDENTITY(1,1) NOT NULL,  [MacAddress] [nvarchar](max) NULL,  [cattleId] [int] NULL,  [lastTransmitDate] [datetime2](7) NOT NULL,  [activationDate] [datetime2](7) NOT NULL,  [linkingDate] [datetime2](7) NOT NULL,  [lastSyncDate] [datetime2](7) NOT NULL,  [antennaId] [int] NOT NULL,  [antennaName] [nvarchar](max) NULL,  [status] [int] NOT NULL,  [softwareVersion] [nvarchar](max) NULL,  [FarmID] [int] NOT NULL,  [UserId] [nvarchar](max) NULL,  CONSTRAINT [PK\_SmartCattle.SensorTbl] PRIMARY KEY CLUSTERED  (  [ID] ASC  )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  ) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]  GO  ALTER TABLE [SmartCattle].[SensorTbl] WITH CHECK ADD CONSTRAINT [FK\_SmartCattle.SensorTbl\_SmartCattle.FarmTbl\_FarmID] FOREIGN KEY([FarmID])  REFERENCES [SmartCattle].[FarmTbl] ([ID])  GO  ALTER TABLE [SmartCattle].[SensorTbl] CHECK CONSTRAINT [FK\_SmartCattle.SensorTbl\_SmartCattle.FarmTbl\_FarmID]  GO |

جدول CattlePositionTbl

|  |
| --- |
| USE [smartCattle]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [SmartCattle].[CattlePositionTbl](  [ID] [int] IDENTITY(1,1) NOT NULL,  [Latitude] [float] NOT NULL,  [Longitude] [float] NOT NULL,  [cattleId] [int] NOT NULL,  [date] [datetime2](7) NOT NULL,  [LastRecievedId] [bigint] NOT NULL,  [FarmID] [int] NOT NULL,  [FreeStall] [int] NOT NULL,  CONSTRAINT [PK\_SmartCattle.CattlePositionTbl] PRIMARY KEY CLUSTERED  (  [ID] ASC  )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  ) ON [PRIMARY]  GO  ALTER TABLE [SmartCattle].[CattlePositionTbl] WITH CHECK ADD CONSTRAINT [FK\_SmartCattle.CattlePositionTbl\_SmartCattle.CattleTbl\_cattleId] FOREIGN KEY([cattleId])  REFERENCES [SmartCattle].[CattleTbl] ([ID])  ON DELETE CASCADE  GO  ALTER TABLE [SmartCattle].[CattlePositionTbl] CHECK CONSTRAINT [FK\_SmartCattle.CattlePositionTbl\_SmartCattle.CattleTbl\_cattleId]  GO  ALTER TABLE [SmartCattle].[CattlePositionTbl] WITH CHECK ADD CONSTRAINT [FK\_SmartCattle.CattlePositionTbl\_SmartCattle.FarmTbl\_FarmID] FOREIGN KEY([FarmID])  REFERENCES [SmartCattle].[FarmTbl] ([ID])  GO  ALTER TABLE [SmartCattle].[CattlePositionTbl] CHECK CONSTRAINT [FK\_SmartCattle.CattlePositionTbl\_SmartCattle.FarmTbl\_FarmID]  GO |

جدول CattleTbl

|  |
| --- |
| USE [smartCattle]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [SmartCattle].[CattleTbl](  [ID] [int] IDENTITY(1,1) NOT NULL,  [age] [int] NOT NULL,  [preg] [int] NOT NULL,  [milkAvg] [float] NOT NULL,  [healthStatus] [int] NOT NULL,  [animalNumber] [int] NOT NULL,  [heatStatus] [int] NOT NULL,  [birthDate] [datetime2](7) NOT NULL,  [Dim] [int] NOT NULL,  [fertilityStatus] [int] NOT NULL,  [lactationNumber] [int] NOT NULL,  [InseminationCount] [int] NOT NULL,  [lastInseminationDate] [datetime2](7) NULL,  [lastCalvingDate] [datetime2](7) NULL,  [calvingCount] [int] NOT NULL,  [CattleGroupId] [int] NULL,  [FreeStallId] [int] NULL,  [FarmID] [int] NOT NULL,  [UserId] [nvarchar](128) NULL,  [CattleHerd\_ID] [int] NULL,  CONSTRAINT [PK\_SmartCattle.CattleTbl] PRIMARY KEY CLUSTERED  (  [ID] ASC  )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  ) ON [PRIMARY]  GO |

جدول CattleGroupTbl

|  |
| --- |
| USE [smartCattle]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [SmartCattle].[CattleGroupTbl](  [ID] [int] IDENTITY(1,1) NOT NULL,  [name] [nvarchar](max) NULL,  [Description] [nvarchar](max) NULL,  [order] [nvarchar](max) NULL,  [code] [nvarchar](max) NULL,  [FarmID] [int] NOT NULL,  [UserId] [nvarchar](128) NULL,  CONSTRAINT [PK\_SmartCattle.CattleGroupTbl] PRIMARY KEY CLUSTERED  (  [ID] ASC  )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  ) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]  GO  ALTER TABLE [SmartCattle].[CattleGroupTbl] WITH CHECK ADD CONSTRAINT [FK\_SmartCattle.CattleGroupTbl\_SmartCattle.AspNetUsers\_UserId] FOREIGN KEY([UserId])  REFERENCES [SmartCattle].[AspNetUsers] ([Id])  GO  ALTER TABLE [SmartCattle].[CattleGroupTbl] CHECK CONSTRAINT [FK\_SmartCattle.CattleGroupTbl\_SmartCattle.AspNetUsers\_UserId]  GO  ALTER TABLE [SmartCattle].[CattleGroupTbl] WITH CHECK ADD CONSTRAINT [FK\_SmartCattle.CattleGroupTbl\_SmartCattle.FarmTbl\_FarmID] FOREIGN KEY([FarmID])  REFERENCES [SmartCattle].[FarmTbl] ([ID])  GO  ALTER TABLE [SmartCattle].[CattleGroupTbl] CHECK CONSTRAINT [FK\_SmartCattle.CattleGroupTbl\_SmartCattle.FarmTbl\_FarmID]  GO |

جدول EnvSensors

|  |
| --- |
| USE [smartCattle]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [SmartCattle].[EnvSensors](  [id] [int] IDENTITY(1,1) NOT NULL,  [FreeStallId] [int] NULL,  [FarmId] [int] NULL,  [Lat] [float] NULL,  [Lng] [float] NULL,  [MAC] [nvarchar](50) NULL  ) ON [PRIMARY]  GO |

جدول FarmTbl

|  |
| --- |
| USE [smartCattle]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [SmartCattle].[FarmTbl](  [ID] [int] IDENTITY(1,1) NOT NULL,  [name] [nvarchar](max) NULL,  [LogoUrl] [nvarchar](max) NULL,  [email] [nvarchar](max) NULL,  [website] [nvarchar](max) NULL,  [Latitude] [nvarchar](max) NULL,  [Longitude] [nvarchar](max) NULL,  [UserId] [nvarchar](max) NULL,  [MapGeoJson] [nvarchar](max) NULL,  [MapFilePath] [nvarchar](max) NULL,  [UerIdentify] [int] NULL,  CONSTRAINT [PK\_SmartCattle.FarmTbl] PRIMARY KEY CLUSTERED  (  [ID] ASC  )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  ) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]  GO  ALTER TABLE [SmartCattle].[FarmTbl] WITH CHECK ADD CONSTRAINT [FK\_FarmTbl\_UserInfo] FOREIGN KEY([UerIdentify])  REFERENCES [SmartCattle].[UserInfo] ([ID])  GO  ALTER TABLE [SmartCattle].[FarmTbl] CHECK CONSTRAINT [FK\_FarmTbl\_UserInfo]  GO |

جدول FreeStallTbl

|  |
| --- |
| USE [smartCattle]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [SmartCattle].[FreeStallTbl](  [ID] [int] IDENTITY(1,1) NOT NULL,  [name] [nvarchar](max) NULL,  [Description] [nvarchar](max) NULL,  [code] [int] NOT NULL,  [location] [geography] NULL,  [FarmID] [int] NOT NULL,  [GroupID] [int] NULL,  [UserId] [nvarchar](128) NULL,  CONSTRAINT [PK\_SmartCattle.FreeStallTbl] PRIMARY KEY CLUSTERED  (  [ID] ASC  )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  ) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]  GO  ALTER TABLE [SmartCattle].[FreeStallTbl] WITH CHECK ADD CONSTRAINT [FK\_SmartCattle.FreeStallTbl\_SmartCattle.AspNetUsers\_UserId] FOREIGN KEY([UserId])  REFERENCES [SmartCattle].[AspNetUsers] ([Id])  GO  ALTER TABLE [SmartCattle].[FreeStallTbl] CHECK CONSTRAINT [FK\_SmartCattle.FreeStallTbl\_SmartCattle.AspNetUsers\_UserId]  GO  ALTER TABLE [SmartCattle].[FreeStallTbl] WITH CHECK ADD CONSTRAINT [FK\_SmartCattle.FreeStallTbl\_SmartCattle.CattleGroupTbl\_GroupID] FOREIGN KEY([GroupID])  REFERENCES [SmartCattle].[CattleGroupTbl] ([ID])  GO  ALTER TABLE [SmartCattle].[FreeStallTbl] CHECK CONSTRAINT [FK\_SmartCattle.FreeStallTbl\_SmartCattle.CattleGroupTbl\_GroupID]  GO  ALTER TABLE [SmartCattle].[FreeStallTbl] WITH CHECK ADD CONSTRAINT [FK\_SmartCattle.FreeStallTbl\_SmartCattle.FarmTbl\_FarmID] FOREIGN KEY([FarmID])  REFERENCES [SmartCattle].[FarmTbl] ([ID])  ON DELETE CASCADE  GO  ALTER TABLE [SmartCattle].[FreeStallTbl] CHECK CONSTRAINT [FK\_SmartCattle.FreeStallTbl\_SmartCattle.FarmTbl\_FarmID]  GO |

جدول UserInfo

|  |
| --- |
| USE [smartCattle]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [SmartCattle].[UserInfo](  [ID] [int] IDENTITY(1,1) NOT NULL,  [Name] [nvarchar](max) NULL,  [Family] [nvarchar](max) NULL,  [Email] [nvarchar](max) NULL,  [Password] [nvarchar](max) NULL,  [Role] [nvarchar](max) NULL,  [Permissions] [nvarchar](max) NULL,  [FarmId] [int] NULL,  CONSTRAINT [PK\_UserInfo] PRIMARY KEY CLUSTERED  (  [ID] ASC  )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  ) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]  GO |

جدول UserPermissions

|  |
| --- |
| USE [smartCattle]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [SmartCattle].[UserPermissions](  [ID] [int] IDENTITY(1,1) NOT NULL,  [Controller] [nvarchar](max) NULL,  [Action] [nvarchar](max) NULL,  [Comment] [nvarchar](max) NULL,  [UniqueId] [nvarchar](max) NULL  ) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]  GO |

جدول UserRoles

|  |
| --- |
| USE [smartCattle]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [SmartCattle].[UserRoles](  [ID] [int] IDENTITY(1,1) NOT NULL,  [Name] [nvarchar](max) NULL,  [Permissions] [nvarchar](max) NULL,  [Comment] [nvarchar](max) NULL  ) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]  GO |

جدول Notifications

|  |
| --- |
| USE [smartCattle]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [SmartCattle].[Notifications](  [ID] [int] IDENTITY(1,1) NOT NULL,  [Topic] [nvarchar](max) NULL,  [Comment] [nvarchar](max) NULL,  [FarmID] [int] NULL,  [RoleName] [nvarchar](max) NULL,  [CreatedDate] [nvarchar](max) NULL,  [Status] [nvarchar](max) NULL,  [NotificationType] [nvarchar](max) NULL,  [Snooze] [int] NULL,  [TagName] [nvarchar](max) NULL,  [SnoozeMsg] [nvarchar](max) NULL  ) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]  GO |

جدول FreeStallNotificationsSetting

|  |
| --- |
| USE [smartCattle]  GO  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [SmartCattle].[FreeStallNotificationsSetting](  [ID] [int] IDENTITY(1,1) NOT NULL,  [FarmId] [int] NULL,  [GroupName] [nvarchar](max) NULL,  [Topic] [nvarchar](max) NULL,  [Roles] [nvarchar](max) NULL,  [Comment] [nvarchar](max) NULL,  [PeroidTime] [int] NULL,  [WindowTime] [int] NULL,  [TempMin] [float] NULL,  [TempMax] [float] NULL,  [HumMin] [float] NULL,  [HumMax] [float] NULL,  [THIMin] [float] NULL,  [THIMax] [float] NULL,  [CreateDate] [datetime2](7) NULL,  [ActivationState] [nvarchar](max) NULL,  [SnoozeTime] [nvarchar](max) NULL  ) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]  GO |

جدول CattleNotificationsSetting

|  |
| --- |
| USE [smartCattle]  GO  /\*\*\*\*\*\* Object: Table [SmartCattle].[CattleNotificationsSetting] Script Date: 4/12/2018 10:05:39 PM \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [SmartCattle].[CattleNotificationsSetting](  [ID] [int] IDENTITY(1,1) NOT NULL,  [FarmId] [int] NULL,  [GroupName] [nvarchar](max) NULL,  [Topic] [nvarchar](max) NULL,  [Roles] [nvarchar](max) NULL,  [Comment] [nvarchar](max) NULL,  [PeroidTime] [int] NULL,  [WindowTime] [int] NULL,  [CattleTempMin] [float] NULL,  [CattleTempMax] [float] NULL,  [SittingMin] [float] NULL,  [SittingMax] [float] NULL,  [WalkingMin] [float] NULL,  [WalkingMax] [float] NULL,  [RuminationMin] [float] NULL,  [RuminationMax] [float] NULL,  [DrinkingMin] [float] NULL,  [DrinkingMax] [float] NULL,  [EatingMin] [float] NULL,  [EatingMax] [float] NULL,  [StandingMin] [float] NULL,  [StandingMax] [float] NULL,  [CreateDate] [datetime2](7) NULL,  [ActivationState] [nvarchar](max) NULL,  [SnoozeTime] [nvarchar](max) NULL  ) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]  GO |

فصل چهار

توضیح اکشن های برنامه

مقدمه

در این فصل به توضیح خط به خط کدها پرداخته می شود. ساختاری که طبق آن این فصل شکل خواهد گرفت بدین شرح است. از mvc برای طراحی اسمارت کتل استفاده شده است. mvc مخفف کلمات Model View Controller هست. در واقع MVC بر روی معماری های چند لایه ای جهت جداسازی قسمت های مختلف برنامه است. هر یک از بخش های معماری MVC یعنی Model و View و controller را به شکل زیر تعریف می کنیم.

Model

در mvc مدل ها همان کلاس های برنامه هستند که شبیه به دیتابیس هستند ولی می توانند متفاوت و بیشتر باشند. همچنین Model وظیفه تایید داده ها را برعهده دارد تا اطلاعات درست درون آنها قرار بگیرد و برای دیتابیس ارسال شود.

View

این بخش اطلاعات را به کاربر نشان می دهد، ویو کاری به منطق برنامه ندارد و فقط یک model دریافت می کند و آن را به هر شکلی که نیاز باشد به کاربر نشان می دهد، این model هم توسط کنترلر ایجاد شده است و برای view رسال شده است. نکته مهمی که در بخش View باید مد نظر داشت این است که این لایه مسئول کنترل صحت داده های وارد شده از طریق کاربر و همچنین مسئول صحت داده های نشان داده شده به کاربر نیست . در واقع این بخش با داده های خام کار میکند و مسولیت صحت داده با model است.

Controller

این بخش همانگونه که از اسم آن مشخص است، یک بخش کنترل کننده است، ساده بگویم controller بیشتر مواقع دو جور کار می کند :

controller اطلاعات را از دیتابیس در قالب model هایی می گیرد و بعد پردازش خود را روی آن انجام می دهد و اطلاعات را در قالب model جدیدی به view می فرستد تا ویو هر طور خواست این اطلاعات را به کاربر نشان دهد.مثل فرض کنید که ما برای صفحه پروفایل یک view برای موبایل داریم و یک view برای ویندوز، حال کنترولر بنا به تشخیص که کاربر با موبایل است یا نه یک view را انتخاب می کند تا برای کاربر ارسال شود و می بینید که یک اطلاعات مشخص در دو view متفاوت ظاهر می شود و view (ظاهر) از منطق برنامه جدا است.

controller اطلاعات را از مدلی که از view برای او ارسال کرده است دریافت می کند و پس انجام اقداماتی آن را در دیتابیس ذخیره می کند و view دیگری را به کاربر می فرستد تا ببیند. به زبان ساده Controller ها بخش اصلی از سناریوی یک برنامه هستند. بنابراین کدهای نرم افزار اسمارت کتل بر اساس Controller ها به ترتیب حروف الفبا شرح داده خواهد شد.

AccountController

وظیفه کنترلر Account، بررسی معتبر بودن کاربر و لاگین کردن وی به اسمارت کتل است. این کنترل با استفاده از اکشن Login صلاحیت کاربر را برای ورودی به نرم افزار بررسی می کند. در ادامه به توضیح اکشن های کنترلر AccountController پرداخته می شود.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | [HttpPost]  [AllowAnonymous]  [ValidateAntiForgeryToken]  public async Task<ActionResult> Login(LoginViewModel model, string returnUrl) |  |
|  |  |  |

از این متد برای بررسی صحت یوزر و پسورد وارد شده توسط کاربر استفاده می شود. اگر کاربر یوزر و پسورد اشتباهی را وارد کند زمان و آی کاربر در دیتابیس ذخیره می شود. اگر برای بار دوم پسورد اشتباهی وارد کند، مدت زمان وارد کردن دو پسورد با هم مقایسه می شود. اگر زمان آن ها زیر 3 ثانیه باشد، اون نام کاربری غیر فعال می شود. در غیر این صورت کاربر می تواند تا 5 بار پسورد اشتباه وارد کند. در بار 5ام نام کاربری غیر فعال می شود.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | [HttpPost]  [AllowAnonymous]  [ValidateAntiForgeryToken]  public async Task<ActionResult> Register(RegisterViewModel model) |  |
|  |  |  |

از این اکشن برای ثبت نام کاربر استفاده می شود. ابتدا اطلاعات وارد شده کاربر به یک مدل تبدیل می شود و سپس این مدل از نظر وجود موارد امنیتی بررسی می شود. اگر مشکلی وجود نداشت داده ها در دیتابیس ذخیره می شود. البته در مورد رعایت موارد امنیتی به صورت مفصل در فصل های بعدی توضیح داده شده است.

CattleController

یکی از مهمترین کنترلر های اسمارت کتل CattleController است. در ادامه در مورد اکشن های تشکیل دهنده آن صحبت خواهد شد.

فصل چهار

امنیت

مقدمه

در این فصل در مورد انواع حملات به برنامه های تحت وب و راه های مقابله با آن صحبت خواهد شد. همچنین سعی می شود به طور دقیق در مورد تست های نفوذ توضیح داده شود.

حمله SQL Injection

یک حمله SQL Injection شامل وارد کردن و یا تزریق یک کوئری SQL از طریق اطلاعات ورودی از سوی کاربر به برنامه می باشد.یک exploit موفق که منجر به تزریق SQL بشود می تواند اطلاعات حساس را از دیتابیس بخواهند و یا اطلاعات را حذف و یا ویرایش نماید. کار دیگری که می تواند انجام دهد این است که بتواند DBMS را خاموش کند یا این که محتویات یک فایل بر روی سیستم DBMS را فراخوانی کند و در برخی حالات هم می تواند فرمان هایی را برای سیستم عامل صادر کند.

برای مثال صفحه لاگین اسمارت کتل رو نظر بگیرید. صفحات Login معمولا دارای فیلدهایی هستند که دو مقدار ID و Password را از کاربر گرفته و سپس با استفاده از یک دستور SQL آن را پردازش می کنند. نمونه ساده ای از این دستور به صورت زیر است:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | SELECT COUNT(UserID) FROM tblUsers WHERE UserID=''' & UserID.Text & ''' AND Pass=''' & Password.Text |  |
|  |  |  |

در این مثال UserID و Password دو کنترل TextBox هستند که مقادیر آنها بایستی مورد پردازش قرار گیرد. حال فرض می کنیم کاربر مقادیر را به صورت Pooya و 123 وارد نماید، در این صورت جمله SQL به صورت زیر تولید می شود:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | SELECT COUNT(UserID) FROM tblUsers WHERE UserID='Pooya' AND Pass=’123’ |  |
|  |  |  |

خوب تا اینحا مشکلی وجود ندارد. حال فرض کنید که هکر به جای کلمه کاربری خود عبارت زیر را وارد نماید:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | ' OR 1=-1 |  |
|  |  |  |

تکه کدی که در بالا درج شده دقیقا باید به همین شکل باشد. یعنی وجود تک کوتیشن و مساوی عبارت نامعادل یک مساوی یک منفی می بایست به همین ترتیب درج شود. در این صورت عبارت SQL زیر تولید خواهد شد:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | SELECT COUNT(UserID) FROM tblUsers WHERE UserID='' OR 1=1 – AND PASS='' |  |
|  |  |  |

کاراکتر - در SQL علامت توضیحات است و عبارت پس از آن دیگر پردازش نمی شود لذا هنگام پردازش دستور چون همواره 1=1 است و با توجه به OR به کار رفته، بدون توجه به خالی بودن مقدار مقابل UserID ، کوئری از نظر دیتابیس کافی تلقی شده و کلیه رکوردهای بانک اطلاعاتی برگردانده می شود و Attacker اصطلاحا Authenticate می شود. از این نوع دستورات که به دستورات SQL Injection معروف هستند زیاد هست و راه های مختلفی هم برای مقابله با آن ها وجود دارد.

ساختار شی گرای ASP.NET که اسمارت کتل بر اساس آن نوشته شده است و امکانات این ساختار به برنامه نویسان امکان مانور بیشتری را داده است. حال ما کد کامل اصلاح شده را در زیر آورده و سپس توضیحات آن را ذکر می کنیم:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | Dim strSQL As String = ''SELECT COUNT(UserID) FROM tblUsers WHERE UserID=@UserID AND Password=@Password''  Dim cmndCheck As OleDbCommand = New OleDbCommand(strSQL, \_Connection) cmndCheck.Parameters.Add(''@UserID'', UserID.Text);  cmndCheck.Parameters.Add(''@Password'', Password.Text);  cmndCheck.Connection.Open()  Dim IsValid As Integer = cmndCheck.ExecuteScalar()  If IsValid > 0  '... Some Code here... User is authenticated  Else  '... Some Code here...  User is not aututorized to view the page End If |  |
|  |  |  |

تغییر در همان خط اول یعنی دستور SQL مشخص است این بار به جای اینکه مقادیر دریافتی از فیلدها با Single Quote به خود پردازشگر دستور داده شود مقادیر با پارامترهای تولید شده توسط آبجکت OleDbCommand جایگزین می شود و در آخر cmndCheck پردازش شده چنانچه مقدار تولید شده توسط ExecuteScalar بزرگتر از صفر باشد بدین معنی است که حداقل یک رکورد با شرایط مورد نظر ما پیدا شده است. (متغیر Connection\_ که باید برابر ConnectionString اسمارت کتل باشد) تفاوت اصلی این روش نسبت به روش ناامن قبلی این است که در اینجا مقادیر به صورت کاملا پارامتری شده پردازش می شوند نه به صورت قطعاتی از یک رشته حرفی که در واقع دام اصلی SQL Injection به شمار می آید. در واقع ما در اینجا از تکنیکهای رشته سازی با استفاده از & یا + اجتناب کرده ایم.

نکته مهم دیگری که باید حتما مورد توجه قرار گیرد این است که یکی از راههایی که هکرها برای حمله به نرم افزار استفاده می کنند خطاسازی صوری در زمان اجرای Application و مطالعه اطلاعات خطای دریافت شده است. برای جلوگیری از این اتفاق ساختار بعضی از قسمتهای کد در Application از روال خطایابی Customize شده استفاده می شود. به عنوان مثال وقتی کاربری پسورد خود را اشتباه وارد می کند، به جای آن که به وی گفته شود که پسورد اشتباه است، پیامی مبتنی بر عدم دسترسی کاربر برای وی به نمایش در میاید. به این ترتیب هکر نمی تواند متوجه شود که آیا نام کاربری را اشتباه وارد کرده است یا کلمه عبور.

مثال ها و راه حل های مختلفی برای حملات Injection وجود دارد. برای بررسی بیشتر مثال ها می توان به سایت های زیر مراجعه کرد.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | [www.sqlsecurity.com](http://www.sqlsecurity.com)  [www.nextgenss.com](http://www.nextgenss.com) |  |
|  |  |  |

چنین سیستمی در ابتدای پیاده سازی اسمارت کتل در نظر گرفته شده بود. این سیستم خوبی ها و بدی های مختلف دارد. یکی از مزایای آن سادگی در فهم و پیاده سازی کوئری های ساخته شده است ولی از بدی های آن می توان به عدم پشتیبانی از رنج وسیعی از تزریق ها نام برد. برای بهبود حفاظت اسمارت کتل در مقابل حمله های مبتنی بر Injection، از First Code Entity و landa Expression ها استفاده شده. به این ترتیب که کوئری های مورد نیاز به صورت Landa به ORM داده می شود و سپس توسط ORM کار مورد نظر بر روی دیتابیس انجام می شود.

حملاتی از نوع Variable Manipulation و Buffer Overflow

علاوه برحملات SQL Injection عليه برنامه هاي تحت وب، دو دسته دیگر از حملات با نام های Variable Manipulation و Buffer Overflow وجود دارد. در اين تكنيك، مهاجم سعي بر تغيير متغيرهاي موجود در برنامه دارد. در نتيجه اين تغييرات، منطق برنامه دچار مشكل مي شود. مثال كلاسيك اين نوع حمله eShoplifting نام دارد كه در آن مهاجم با دسترسي به يك يا چند متغير قيمت در وب سايت فروش آنلاين و تغيير مقدار آنها، منطق برنامه را فريب داده و منجر به محاسبه اشتباه قيمت محصول توسط برنامه مي شود. در نتيجه مهاجم مي تواند اقدام به خريد يك آيتم گران قيمت با قيمت بسيار پايين كند. در اكثر موارد برنامه قادر به تشخيص تغيير قيمت نيست و روند كار در حالت عادي انجام خواهد شد. براي بررسي وجود چنين رخنه پذيري در اسمارت کتل به این صورت عمل شد:

كد HTML صفحات اسمارت کتل بررسی شد و متغیرهایی که موارد مهم نظیر نام جداول و یا آی دی برخی سطرها را در خود نگه داری می کردند رمزگذاری شدند. شیوه رمزگذاری به این صورت است که هربار که از سمت View درخواستی به یکی از کنترلرها ارسال می شود، زمان جاری از کنترلر Utile دریافت می شود. سپس این زمان با یک عدد رندم و یک delimiter با دیتای مورد نظر ترکیب و در نهایت نتیجه انکریپت می شود. داده انکریپت شده به کنترلر مورد نظر ارسال می شود و در آن جا ابتدا دکریپت می شود و سپس با delimiter بخش های مختلف داده از هم مجزا می شود. ابتدا زمان چک می شود. اگر اختلاف زمان داخل پکت از زمان جاری بیش از ده ثانیه باشد نشان می دهد که هکر قصد دارد با ارسال یک پکت Sniff شده به دفعات مختلف، به سیستم نفوذ کند. این یعنی هکر تونسته یک پکت سالم رو از سیستم پیدا کنه ولی نتونسته الگوریتم انکریپشن آن را مورد هدف قرار دهد. با توجه به این که Threshold ده ثانیه ای برای معتبر بودن پکت ها در نظر گرفته شده، هر پکت ساخته شده پس از سپری شدن زمان اعتبارش دیگر هیچ ارزشی ندارد.

یکی دیگر از حملاتی که در اسمارت کتل بررسی شد Buffer overflow نام دارد. همانطور كه مي دانيد، يك برنامه حجم مشخصي از حافظه را اشغال مي كند. درصورتي كه اطلاعاتي بيشتر از فضاي اختصاص يافته به برنامه (حتي باندازه يك بايت)، وارد حافظه شود، منجر به سرريزي خواهد شد. در زير به بيان دو نوع رايج سرريز به نام هاي" سرريزي پشته" و "سرريزي هيپ" پرداخته می شود:

Stack Overflow پشته، بخشي از حافظه است. سیستم اطلاعاتي را كه نمي تواند در رجيسترهاي خود ذخيره كند، در پشته ذخيره مي كند. در اين تكنيك مهاجم با آگاهي از عدم كنترل نوع و سايز مقادير ورودي يك تابع توسط برنامه، اقدام به ارسال مقادير بيشتر از فضاي حافظه و نهايتا بازنويسي آدرس تابع مخرب خود به جاي آدرس بازگشت تابع اصلي مي كند. تابع مخرب عبارتست از هر آن که چه منجر به خطر پذيري برنامه و سرور شود، مثلا در محيط لينوكس دستور ساده EXEC("sh") منجر به باز شدن پنجره دستورات و خطرپذيري بسيار براي سرور مي شود، حال وقتي برنامه قفل مي شود، اقدام به بازيابي توسط آدرس بازگشت مي كند و چون اين آدرس به دستور مخرب مشخص شده از سوي مهاجم تغيير كرده است، منجر به اجراي دستور مخرب خواهد شد.

در حمله از نوع Heap Overflow وقتي يك برنامه با حجم بزرگي از اطلاعات نيازمند پردازش مواجه مي شود، بخشي از حافظه به نام هيپ به منظور مديريت اطلاعات فوق در نظر گرفته مي شود. در زبان های سطح پایین، برای هندل کردن چنین موضوعی، لازم است تا برنامه نویس میزان حافظه مورد نیاز را خودش تعیین کند ولی در زبان های برنامه نویسی سطح بالا نظیر سی شارپ که اسمارت کتل با آن نوشته شده است، برنامه نویس نیازی به تنظیم میزان حافظه ندارد مگر آن که ضرورتی در کار تشخیص داده شود. زیرا چنین زبان های برنامه نویسی به صورت خودکار و با توجه به برنامه نوشته شده، میزانی از حافظه را به کد تخیصص می دهند. در نرم افزار نوشته شده اسمارت کتل تمامی کدهایی که به نحوی به صورت دستی حافظه ای را به کد تخصیص می دادند حذف شدند و کد جدیدی به جای آن ها درج شد.

حملات از نوع CSRF

نوع دیگری از حملات که باید در برنامه‌های وب به آن‌ها دقت داشت به نام CSRF یا Cross site request forgery معروف هستند. برای مثال فرض کنید کاربری قبل از اینکه بتواند در نرم افزار کار خاصی را انجام دهد، نیاز به اعتبار سنجی داشته باشد. پس از لاگین شخص و ایجاد کوکی و سشن معتبر، همین شخص به سایت دیگری مراجعه می‌کند که در آن مهاجمی بر اساس وضعیت جاری اعتبار سنجی او مثلا لینک حذف کاربری یا افزودن اطلاعات جدیدی را به برنامه ارائه می‌دهد. چون سشن شخص و کوکی مرتبط به سایت اول هنوز معتبر هستند و شخص سایت را نبسته است، «احتمال» اجرا شدن درخواست مهاجم بالا است. خصوصا اگر از مرورگرهای قدیمی استفاده کند. بنابراین نیاز است بررسی شود آیا درخواست رسیده واقعا از طریق فرم‌های برنامه ما صادر شده است یا اینکه شخصی از طریق سایت دیگری اقدام به جعل درخواست‌ها کرده است.

CSRF یا Cross Site Request Forgery به صورت خلاصه به این معنا است که شخص مهاجم اعمالی را توسط کاربر و با سطح دسترسی وی بر روی سایت انجام دهد و اطلاعات مورد نظر خود را استخراج کرده و به هر سایتی که تمایل دارد ارسال کند. این‌کار عموما با تزریق کد در صفحه صورت می‌گیرد.

برای مقابله با این نوع خطاها ابتدا داخل برنامه اسمارت کتل از متد Html.AntiForgeryToken استفاده شد. کار این متد ایجاد یک فیلد مخفی با مقداری منحصربفرد بر اساس اطلاعات سشن جاری کاربر است، به علاوه ارسال یک کوکی خودکار تا بتوان از تطابق اطلاعات اطمینان حاصل کرد. در مرحله بعد فیلتر ValidateAntiForgeryToken جهت بررسی مقدار token دریافتی به متد ثبت اطلاعات اضافه شد. سپس مقدار دریافتی از فیلد مخفی از قبل درج شده در فرم با مقدار موجود در کوکی سایت بررسی و تطابق داده شد. اگر این مقادیر تطابق نداشته باشند، یک پیام عدم دسترسی صادر شده و از پردازش اطلاعات رسیده جلوگیری می‌شود.

علاوه بر این‌ها حین استفاده از متد و فیلتر یاد شده، از یک salt مجزا نیز به ازای هر فرم، استفاده می شود. به این ترتیب tokenهای تولید شده در فرم‌های مختلف سایت یکسان نخواهند بود.

به علاوه باید دقت داشت که ValidateAntiForgeryToken فقط با فعال بودن کوکی‌ها در مرورگر کاربر کار می‌کند و اگر کاربری پذیرش کوکی‌ها را غیرفعال کرده باشد، قادر به ارسال اطلاعاتی به برنامه نخواهد بود. همچنین این فیلتر تنها در حالت HttpPost قابل استفاده است. در جاهایی که با HttpGet ساده کار می‌کند،‌ تبدیل به HttpPost شدند تا میزان امنیت برنامه بهبود یابد. از HttpGet فقط برای گزارشگیری و خواندن اطلاعات از برنامه استفاده می شود و نه ثبت اطلاعات.

حمله XSS

در این نوع حمله هکر به نرم افزارهای تحت وب از طریق اسکریپت نویسی حمله می‌کند و به دنبال فیلد های ورود اطلاعاتی می‌گردد و در صورت عدم اعتبارسنجی یا Validation فیلدهای ورودی می‌تواند به آن نرم افزار حمله کند. هدف اصلی این نوع حمله بدست آودن اطلاعات از کاربران و کلاینت هایی است که به سرور متصل می‌شوند. بیشتر اطلاعاتی که از طریق XSS بدست می آید از طریق Cookie هایی است که کلاینت بر روی مرورگر خود بر جای می گذارد. در حملات XSS ، کوکی مرورگر قربانی به سرقت میره. برای آن که در اسمارت کتل بتوان با چنین حمله هایی مقابله کرد چند کار صورت گرفت. در ابتدا فیلدهای ورود اطلاعات regular expression اعتبارسنجی می شود و به کاربران اجازه داده نمی شود که کد مخرب وارد کنند. همچنین در قدم بعدی با استفاده از متدهای جلوگیری از تزریق کدهای HTML و JavaScript، جلوی Injection گرفته می شود. یکی دیگر از کارهایی که انجام ی شود استفاده از سشن ها به جای کوکی هاست. در هر جای برنامه که از کوکی استفاده شد، تمام تبدیل به سشن شدند و همچنین با استفاده از یک Salt تبدیل به یک کد یونیک شدند.

فصل چهار

تنظیمات مربوط به سرور

ایجاد حساب کاربری FTP

برای آپلود فایل های پروژه بر روی سرور راه های بسیاری وجود دارد. یکی از راه های استاندارد و اصولی استفاده از پورتکل FTP است. برای ایجاد پروتکل طبق روش زیر عمل می کنیم.

۱- فعال کردن رول‌های WebServer (IIS) و FTP Server

1. Server Manager را اجرا کنید
2. به Manage > Add Roles and Features بروید.
3. Role-based or feature-based installation را انتخاب کنید.
4. سرور خود را انتخاب کنید
5. تیک Web Server (IIS) را بزنید
6. یک پاپ آپ باز می‌شود که باید از بین گزینه‌های موجود تیک Include management tools (if applicable) را بزنید.
7. دکمه Add Features را بزنید
8. دوباره پایین‌تر آمده و تیک‌های FTP server ، FTP Service و FTP Extensibility را هم بزنید.
9. در این بین Next ها را زده و در نهایت Install را بزنید و پس از اتمام پنجره را ببندید.

۲- ساخت کاربر FTP

1. Server Manager را اجرا کنید
2. در زیر مجموعه Tools گزینه Computer Management را بزنید
3. روی Local Users and Groups کلیک کنید. در صورتی که این گزینه را مشاهده نمی‌کنید باید کاربران خود را روی دامین تعریف کنید (از طریق Active Directory Computer and users در منوی Tools در Server Manager)
4. روی Users کلیک کنید
5. وسط صفحه کلیک راست کرده و New User… را بزنید.
6. اطلاعات کاربر را وارد کرده و Create را بزنید
7. هر تعداد کاربر می‌خواهید ایجاد کنید.

۳- تنظیمات کلی FTP روی IIS

یک سری تنظیمات کلی باید روی IIS خود اعمال کنید تا بتوانید از FTP استفاده کنید. این تنظیمات ساده‌است. کافیست مراحل زیر را انجام دهید:

1. در Server Manager به Tools بروید.
2. روی Internet Information Services (IIS) manager کلیک کنید
3. از پنل سمت چپ روی سرور دابل‌کلیک کنید (ریشه درخت در زیر Start Page)
4. اگر پاپ‌آپی باز شد در باره Microsoft Web Platform، تیک Do not show this message را بزنید و No رو بزنید.
5. در پنل وسط FTP Authentication را دابل‌کلیک کنید
6. اگر می‌خواهید همه بتوانند به FTP دسترسی داشته‌باشند (بدون نیاز به نام کاربری و کلمه عبور) روی Anonymous Authentication کلیک‌راست کرده و Enable را بزنید.
7. اگر می‌خواهید به کاربرانی که در مرحله ۲ بالا ایجاد کردید دسترسی دهید روی Basic Authentication کلیک‌راست کرده و Enable را بزنید.
8. دوباره از پنل سمت چپ روی سرور دابل‌کلیک کنید
9. در پنل وسط FTP Authorization Rules را دابل‌کلیک کنید.
10. تمام قوانین موجود را حذف کنید (در صورت وجود و در صورت نیاز)
11. بعد از حذف تمام قوانید وسط صفحه کلیک‌راست کرده و Add Allow Rule… را بزنید.
12. گزینه Specified users را انتخاب کنید.
13. در کادر مربوطه کاربرانی که در مرحله قبل تعریف کردی را به صورتی که با کاما از هم جدا شده‌اند وارد کنید.
14. سطح دسترسی‌های Read و Write مد نظر خود را به کاربران انتخاب شده بدهید.
15. روی OK کلیک کنید
16. مراحل ۸ تا ۱۵ بالا را به تعداد دلخواه خود تکرار کنید و کاربران بیشتری با سطح دسترسی‌های دلخواهد اضافه کنید.

۴- ساخت وب‌سایت FTP

1. یک فولدر در کامپیوتر خود ایجاد کنید
2. آدرس آن را کپی کنید. یا اینکه یک فولدر که می‌خواهید روی آن دسترسی FTP ایجاد کنید را باز کرده و آدرس آن را کپی کنید.
3. Server Manager را باز کنید
4. به Tools بروید
5. روی Internet Information Services (IIS) Manager کلیک کنید
6. از پنل سمت چپ روی سرور کلیک‌راست کنید (ریشه درخت در زیر Start Page)
7. Add FTP Site را بزنید
8. یک نام برای سایت خودوارد کنید مثلا Armandar FTP Site
9. آدرسی که در بالا (مرحله ۲) کپی کرده بودید در بخش Physical path وارد کنید
10. Next کنید
11. در لیست IP ها IP سرور خود را انتخاب کنید
12. پورت ۲۱ به صورت پیشفرض انتخاب شده می‌توانید پورت دیگری را انتخاب کنید
13. مطمئن شوید تیک Start FTP site automatically خورده شده است.
14. گزینه No SSL را انتخاب کنید. (اگر گواهینامه SSL ندارید)
15. Next کنید.
16. تیک Basic را بزنید تا به کاربرای که مشخص می‌کنید دسترسی به این وب‌سایت FTP داده‌شود.
17. در بخش Allow access گزینه Specified Users را انتخاب کنید
18. در محل مربوطه نامهای کاربری مدنظر خود را وارد کنید.
19. تیک‌های Read و Write را متناسب به نیاز خود انتخاب کنید.
20. Finish را بزنید.

۵- تنظیمات Firewall در IIS

1. در Server Manager به Tools بروید
2. روی Internet Information Services (IIS) manager کلیک کنید
3. از پنل سمت چپ روی سرور دابل‌کلیک کنید (ریشه درخت در زیر Start Page)
4. در پنل وسط FTP Firewall Support را دابل‌کلیک کنید
5. در بخش Data Channel Port Range از 0-0 برای استفاده جهت پورت‌های پیشفرض استفاده کنید
6. یا آن را در صورت نیاز خود تغییر دهید.
7. Apply کنید
8. IIS را ببندید

۶- تنظیمات Firewall در ویندوز

به صورت پیشفرض قواعد مربوط جهت دسترسی به FTP در زمانی که رول مربوط به آن را نصب می‌کنید به صورت اتوماتیک اضافه می‌شود. با این وجود من برای اینکه در صورت پیدا کردن مشکل بتوانید راهی جهت رفع آن پیدا کنید راهکار مربوط به افزودن تنظیمات مربوط به FTP در فایروال ویندوز را به شما آموزش می‌دهم.

1. Server Manager را باز کنید
2. در پنل سمت چپ روی Local Server کلیک کنید
3. در پنل سمت راست لینک مقابل Windows Firewall را کلیک کنید.
4. وقتی فایروال باز شد در پنل سمت چپ روی Advanced Settings کلیک کنید
5. وقتی باز شد Inbound Rules را کلیک کنید.
6. در پنل سمت راست چک کنید ببینید یک قاعده با نام FTP Server (FTP Traffic-In) وجود دارد
7. روی آن دابل کلیک کنید
8. در تب General چک کنید تیک Enabled خورده است.
9. به تب Protocols and Ports بروید
10. پروتوکل باید روی TCP و پورت باید روی ۲۱ باشد. (یا پورتی که شما انتخاب کرده بودید)
11. به تب Advanced بروید
12. مطمئن شوید پروفایل‌های Domain ، Private و Public تیک خورده‌اند.
13. Ok کنید.
14. مراحل ۷ تا ۱۳ را برای قاعده با نام FTP Server Passive (FTP Passive Traffic-In) هم بررسی کنید. با این تفاوت که پورت برای این قاعده باید 1024-65535 باشد.
15. مراحل ۷ تا ۱۳ را برای قاعده با نام FTP Server Secure (FTP SSL Traffic-In) هم بررسی کنید. با این تفاوت که پورت برای این قاعده باید 990 باشد.
16. در پنل سمت چپ Outbound Rules را کلیک کنید.
17. مراحل ۷ تا ۱۳ را برای قاعده با نام FTP Server (FTP Traffic-Out) هم بررسی کنید. با این تفاوت که پورت برای این قاعده باید 20 باشد.
18. مراحل ۷ تا ۱۳ را برای قاعده با نام FTP Server Secure (FTP SSL Traffic-Out) هم بررسی کنید. با این تفاوت که پورت برای این قاعده باید 989 باشد.
19. تمام پنجره‌ها را ببندید.

بعد از آن که مراحل بالا طی شدند امکان دارد که هنوز برای وصل شدن با FTP مشکلی وجود داشته باشد. دو کار لازم هست که انجام شود. اولی ایجاد یک Virtual Directory برای وب سایتی که ایجاد شده و تخصیص FTPUSER ایجاد شده به آن. کار دوم که به مراتب مهمتر هست Share پوشه ایجاد شده و دادن دسترسی به FTPUSER ایجاد شده. برای این کار روی پوشه کلیک راست کنید. در پنجره باز شده روی Edit کلیک کنید. در پنجره Permission for Folder Name بر روی Add کلیک شود. در پنجره Select Users بر روی Advance کلیک شود و یوزری که برای FTP ساختید رو پیدا کنید. سپس آن را انتخاب کنید و تمام گزینه ها را اوکی کنید. به یاد داشته باشید که پرمیژن ها برای Read, Read & Execute, and List Folder Contents می بایست فعال باشند.