Guida per sviluppatori Xamarin

AWS Mobile SDK

AWS Mobile SDK: Guida per sviluppatori Xamarin

Table of Contents

L'SDK AWS mobile per Xamarin è ora incluso in. AWS SDK per .NET Questa guida fa riferimento alla versione archiviata di Mobile SDK per Xamarin.

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.

Cos'è l'SDK AWS Mobile per.NET and Xamarin?

L'SDK AWS mobile per Xamarin è incluso in. SDK per .NET Per ulteriori informazioni, consulta la Guida per gli sviluppatori di AWS SDK per .NET .

Questa guida non è più aggiornata: fa riferimento alla versione archiviata di Mobile SDK per Xamarin.

Guide e argomenti correlati

- Per lo sviluppo di app front-end e mobili, consigliamo di utilizzare. AWS Amplify
- Per considerazioni speciali sull'utilizzo di per le tue app Xamarin, consulta Considerazioni speciali AWS SDK per .NET per il supporto di Xamarin nella Guida per gli sviluppatori.AWS SDK per .NET
- A scopo di riferimento, puoi trovare la versione archiviata di Mobile SDK per Xamarin su.AWS GitHub

Contenuto di riferimento archiviato

L'SDK AWS Mobile archiviato per .NET e Xamarin fornisce un set di librerie.NET, esempi di codice e documentazione per aiutare gli sviluppatori a creare applicazioni mobili connesse per:

- Xamarin iOS
- Xamarin Android
- Windows Phone Silverlight
- Windows RT 8.1
- Windows Phone 8.1

Le app mobili scritte utilizzando l'SDK AWS Mobile for .NET e Xamarin utilizzano la piattaforma nativa APIs , quindi hanno l'aspetto delle applicazioni native. Le librerie.NET dell'SDK forniscono wrapper C# per AWS REST. APIs

Cosa è incluso nell'SDK AWS Mobile per .NET e Xamarin?

I servizi AWS supportati attualmente includono, a titolo esemplificativo ma non esaustivo:

- Amazon Cognito

- Amazon S3
- Amazon DynamoDB
- Amazon Mobile Analytics
- Amazon Simple Notification Service

Questi servizi consentono di autenticare gli utenti, salvare dati di giocatori e giochi, salvare oggetti nel cloud, ricevere notifiche push e raccogliere e analizzare i dati di utilizzo.

L'SDK AWS Mobile per .NET e Xamarin consente inoltre di utilizzare la maggior parte dei servizi AWS supportati dall'SDK AWS per .NET. I servizi AWS specifici per lo sviluppo mobile sono spiegati in questa guida per sviluppatori. Per ulteriori informazioni sull'SDK AWS per .NET, consulta:

- Guida introduttiva a AWS SDK for .NET
- Guida per gli sviluppatori dell'SDK AWS per .NET
- Riferimento all'API AWS SDK for .NET

Compatibilità

L'SDK AWS Mobile per .NET e Xamarin viene fornito come Portable Class Library (PCL). PCL Support è stato aggiunto in Xamarin.Android 4.10.1 e Xamarin.iOS 7.0.4. I progetti Portable Library sono integrati in Visual Studio.

IDEs

Per ulteriori informazioni sull'utilizzo IDEs con la versione archiviata di Xamarin SDK, vedere. Configurazione dell'SDK AWS Mobile per.NET and Xamarin

Come posso ottenere l'SDK AWS Mobile per.NET and Xamarin?

Per ottenere l'SDK AWS Mobile per .NET e Xamarin, consulta Configurazione dell'SDK AWS Mobile per .NET e Xamarin. L'SDK AWS Mobile per .NET e NuGet Xamarin è distribuito come pacchetti. Puoi trovare un elenco completo dei pacchetti di servizi AWS nei pacchetti SDK AWS NuGet o nell'SDK AWS for GitHub .NET Repository.

Informazioni su AWS Mobile Services

Amazon Cognito Identity

Tutte le chiamate effettuate verso AWS richiedono credenziali AWS. Invece di codificare le tue credenziali nelle tue app, ti consigliamo di utilizzare Amazon Cognito Identity per fornire le credenziali AWS alla tua applicazione. Segui le istruzioni in Configurazione dell'SDK AWS Mobile per.NET and Xamarin per ottenere le credenziali AWS tramite Amazon Cognito.

Cognito consente inoltre di autenticare gli utenti utilizzando provider di accesso pubblici come Amazon, Facebook, Twitter e Google, nonché provider che supportano OpenID Connect. Cognito funziona anche con utenti non autenticati. Cognito fornisce credenziali temporanee con diritti di accesso limitati specificate con un ruolo Identity and Access Management (IAM). Cognito è configurato creando un pool di identità associato a un ruolo IAM. Il ruolo IAM specifica a cui l' resources/services app può accedere.

Per iniziare a usare Cognito Identity, consulta Configurazione dell'SDK AWS Mobile per.NET and Xamarin.

Per ulteriori informazioni su Cognito Identity, consulta Amazon Cognito Identity.

Amazon Cognito Sync

Cognito Sync è un servizio AWS e una libreria client che consente la sincronizzazione tra dispositivi dei dati utente relativi alle applicazioni. Puoi utilizzare l'API Cognito Sync per sincronizzare i dati del profilo utente tra i dispositivi e tra i provider di accesso: Amazon, Facebook, Google e il tuo provider di identità personalizzato.

Per iniziare a usare Cognito Sync, consulta Sincronizzazione dei dati utente con Cognito Sync.

Per ulteriori informazioni su Cognito Sync, consulta Amazon Cognito Sync.

Mobile Analytics

Amazon Mobile Analytics ti consente di raccogliere, visualizzare e comprendere l'utilizzo delle app per le tue app mobili. I report sono disponibili per le metriche su utenti attivi, sessioni, fidelizzazione, entrate in-app ed eventi personalizzati e possono essere filtrati per piattaforma e intervallo di date. Amazon Mobile Analytics è progettato per adattarsi alla tua attività e può raccogliere ed elaborare miliardi di eventi da molti milioni di endpoint.

Per iniziare a utilizzare Mobile Analytics, consulta Tracciamento dei dati di utilizzo delle app con Amazon Mobile Analytics.

Per ulteriori informazioni su Mobile Analytics, consulta Amazon Mobile Analytics.

Dynamo DB

Amazon DynamoDB è un servizio di database non relazionale, conveniente, veloce e altamente scalabile e disponibile. DynamoDB rimuove le tradizionali limitazioni di scalabilità sullo storage dei dati mantenendo una bassa latenza e prestazioni prevedibili.

Per iniziare a utilizzare Dynamo DB, consulta Archiviare e recuperare dati con DynamoDB.

Per ulteriori informazioni su Dynamo DB, consulta Amazon DynamoDB.

Amazon Simple Notification Service

Amazon Simple Notification Service (SNS) è un servizio di notifica push veloce, flessibile e completamente gestito che ti consente di inviare messaggi singoli o di inviare messaggi a un numero elevato di destinatari. Amazon Simple Notification Service semplifica ed economica l'invio di notifiche push a utenti di dispositivi mobili, destinatari di e-mail o persino l'invio di messaggi ad altri servizi distribuiti.

Per iniziare a utilizzare SNS per Xamarin iOS, consulta Ricevere notifiche push tramite SNS (Xamarin iOS).

Per iniziare a utilizzare SNS per Xamarin Android, vedi Ricevere notifiche push utilizzando SNS (Xamarin Android).

Per ulteriori informazioni su SNS, consulta Amazon Simple Notification Service (SNS).

Configurazione dell'SDK AWS Mobile per.NET and Xamarin

Puoi configurare l'SDK AWS Mobile per .NET e Xamarin e iniziare a creare un nuovo progetto oppure puoi integrare l'SDK con un progetto esistente. Puoi anche clonare ed eseguire gli esempi per avere un'idea di come funziona l'SDK. Segui questi passaggi per configurare e iniziare a utilizzare l'SDK AWS Mobile per.NET and Xamarin.

Prerequisiti

Prima di poter utilizzare l'SDK AWS Mobile per .NET e Xamarin, devi fare quanto segue:

- Crea un account AWS.
- Installa Xamarin.

Dopo aver completato i prerequisiti:

1. Ottieni le credenziali AWS utilizzando Amazon Cognito.
2. Imposta le autorizzazioni richieste per ogni servizio AWS che utilizzerai nella tua applicazione.
3. Crea un nuovo progetto nel tuo IDE.
4. Installa l'SDK AWS Mobile per.NET and Xamarin.
5. Configura l'SDK AWS Mobile per.NET and Xamarin.

Fase 1: Ottenere le credenziali AWS

Per effettuare chiamate ad AWS nella tua applicazione, devi prima ottenere le credenziali AWS. Puoi farlo utilizzando Amazon Cognito, un servizio AWS che consente all'applicazione di accedere ai servizi dell'SDK senza dover incorporare le tue credenziali AWS private nell'applicazione.

Per iniziare a usare Amazon Cognito, devi creare un pool di identità. Un pool di identità è un archivio di informazioni specifico per il tuo account e identificato da un ID di pool di identità univoco simile al seguente. :

```
"us-east-1:00000000-0000-0000-0000-000000000000"
```

1. Accedi alla console Amazon Cognito, scegli Gestisci identità federate, quindi scegli Crea nuovo pool di identità.

2. Inserisci un nome per il tuo pool di identità e seleziona la casella di controllo per abilitare l'accesso alle identità non autenticate. Scegli Crea pool per creare il tuo pool di identità.
3. Scegli Consenti per creare i due ruoli predefiniti associati al tuo pool di identità, uno per gli utenti non autenticati e uno per gli utenti autenticati. Questi ruoli predefiniti forniscono al tuo pool di identità l'accesso ad Amazon Cognito Sync e Amazon Mobile Analytics.

In genere, utilizzerai solo un pool di identità per applicazione.

Dopo aver creato il pool di identità, ottieni le credenziali AWS creando un `CognitoAWSCredentials` oggetto (passandogli l'ID del pool di identità) e poi passandolo al costruttore di un client AWS come segue. :

```
CognitoAWSCredentials credentials = new CognitoAWSCredentials (
    "us-east-1:00000000-0000-0000-0000-000000000000", // Your identity pool ID
    RegionEndpoint.USEast1 // Region
);

// Example for |MA|
analyticsManager = MobileAnalyticsManager.GetOrCreateInstance(
  credentials,
  RegionEndpoint.USEast1, // Region
  APP_ID // app id
);
```

Fase 2: Impostare le autorizzazioni

Devi impostare le autorizzazioni per ogni servizio AWS che desideri utilizzare nella tua applicazione. Innanzitutto, devi capire come AWS vede gli utenti della tua applicazione.

Quando qualcuno usa la tua applicazione ed effettua chiamate verso AWS, AWS assegna a quell'utente un'identità. Il pool di identità creato nella fase 1 è il luogo in cui AWS archivia queste identità. Esistono due tipi di identità: autenticate e non autenticate. Le identità autenticate appartengono agli utenti autenticati da un provider di accesso pubblico (ad esempio Facebook, Amazon, Google). Le identità non autenticate appartengono agli utenti ospiti.

Ogni identità è associata a un ruolo. AWS Identity and Access Management Nella fase 1, hai creato due ruoli IAM, uno per gli utenti autenticati e uno per gli utenti non autenticati. A ogni ruolo IAM sono associate una o più policy che specificano a quali servizi AWS possono accedere le identità

assegnate a quel ruolo. Ad esempio, la seguente policy di esempio concede l'accesso a un bucket Amazon S3. :

```
{
  "Statement": [
    {
      "Action": [
        "s3:AbortMultipartUpload",
        "s3:DeleteObject",
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::MYBUCKETNAME/*",
      "Principal": "*"
    }
  ]
}
```

Per impostare le autorizzazioni per i servizi AWS che desideri utilizzare nella tua applicazione, modifica la policy allegata ai ruoli.

1. Vai alla console IAM e scegli Ruoli. Digitare il nome del pool di identità nella casella di ricerca. Scegli il ruolo IAM che desideri configurare. Se la tua applicazione consente sia utenti autenticati che non autenticati, devi concedere le autorizzazioni per entrambi i ruoli.
2. Fai clic su Allega criterio, seleziona il criterio desiderato, quindi fai clic su Allega criterio. Le policy predefinite per i ruoli IAM che hai creato forniscono l'accesso ad Amazon Cognito Sync e Mobile Analytics.

Per ulteriori informazioni sulla creazione di policy o per scegliere da un elenco di policy esistenti, consulta IAM Policies.

Fase 3: creazione di un nuovo progetto

Windows

Puoi usare Visual Studio per sviluppare la tua applicazione.

OS X

Puoi usare Visual Studio per sviluppare le tue applicazioni. Lo sviluppo iOS con Xamarin richiede l'accesso a un Mac per eseguire l'app. Per ulteriori informazioni, vedere Installazione di Xamarin.iOS su Windows.

> **Note**
>
> L'IDE Rider commerciale multipiattaforma JetBrains include il supporto per Xamarin su piattaforme Windows e Mac.

Passaggio 4: installare l'SDK AWS Mobile per.NET and Xamarin

Windows

Opzione 1: Installazione tramite la console Package Manager

L'SDK AWS Mobile per .NET e Xamarin è costituito da un set di assembli.NET. Per installare l'SDK AWS Mobile per .NET e Xamarin, esegui il comando install-package per ogni pacchetto nella console Package Manager. Ad esempio, per installare Cognito Identity, esegui quanto segue. :

```
Install-Package AWSSDK.CognitoIdentity
```

I pacchetti AWS Core Runtime e Amazon Cognito Identity sono necessari per tutti i progetti. Di seguito è riportato un elenco completo dei nomi dei pacchetti per ogni servizio.

Servizio	Nome pacchetto
Runtime AWS Core	AWSSDK.Core
Amazon Cognito Sync	AWSSDK.CognitoSync
Amazon Cognito Identity	AWSSDK.CognitoIdentity
Amazon DynamoDB	AWSSDK.Dinamo DBv2
Amazon Mobile Analytics	AWSSDK.MobileAnalytics

Servizio	Nome pacchetto
Simple Storage Service (Amazon S3)	AWSSDK.S3
Amazon SNS	AWSSDK.SimpleNotificationService

Per includere un pacchetto non definitivo, includete l'argomento della riga di `-Pre` comando durante l'installazione del pacchetto come segue. :

```
Install-Package AWSSDK.CognitoSync -Pre
```

Puoi trovare un elenco completo dei pacchetti di servizi AWS nei pacchetti SDK AWS NuGet o nell'SDK AWS for GitHub .NET Repository.

Opzione 2: installa utilizzando il tuo IDE

In Visual Studio

1. Fai clic con il pulsante destro del mouse sul progetto, quindi scegli Gestisci NuGet pacchetti.
2. Cerca il nome del pacchetto che desideri aggiungere al progetto. Per includere i NuGet pacchetti di prelease, scegli Includi Prelease. Puoi trovare un elenco completo dei pacchetti di servizi AWS nei pacchetti SDK AWS su NuGet.
3. Scegliere il pacchetto e quindi scegliere Install (Installa).

Mac (OS X)

In Visual Studio

1. Fai clic con il pulsante destro del mouse sulla cartella dei pacchetti, quindi scegli Aggiungi pacchetti.
2. Cerca il nome del pacchetto che desideri aggiungere al progetto. Per includere i pacchetti di prelease, scegli Mostra NuGet pacchetti di versione non definitiva. Puoi trovare un elenco completo dei pacchetti di servizi AWS nei pacchetti SDK AWS su NuGet.
3. Seleziona la casella di controllo accanto al pacchetto desiderato, quindi scegli Aggiungi pacchetto.

> **Important**
>
> Se state sviluppando utilizzando una Portable Class Library, dovete anche aggiungere il AWSSDK NuGet pacchetto.Core a tutti i progetti derivanti dalla Portable Class Library.

Passaggio 5: configurare l'SDK AWS Mobile per.NET and Xamarin

Imposta la registrazione

Le impostazioni di registrazione vengono impostate utilizzando la `Amazon.AWSConfigs` classe e la `Amazon.Util.LoggingConfig` classe. Puoi trovarli nell'`AWSSdk.Core`assembly, disponibile tramite Nuget Package Manager in Visual Studio. Puoi inserire il codice delle impostazioni di registrazione nel `OnCreate` metodo nel `MainActivity.cs` file per le app Android o nel `AppDelegate.cs` file per le app iOS. È inoltre necessario aggiungere `using Amazon.Util` istruzioni `using Amazon` and ai file.cs.

Configurare le impostazioni di registrazione come segue. :

```
var loggingConfig = AWSConfigs.LoggingConfig;
loggingConfig.LogMetrics = true;
loggingConfig.LogResponses = ResponseLoggingOption.Always;
loggingConfig.LogMetricsFormat = LogMetricsFormatOption.JSON;
loggingConfig.LogTo = LoggingOptions.SystemDiagnostics;
```

Quando si accede SystemDiagnostics, il framework stampa internamente l'output su System.Console. Se vuoi registrare le risposte HTTP, imposta il flag. `LogResponses` I valori possono essere Always, Never o OnError.

È inoltre possibile registrare le metriche delle prestazioni per le richieste HTTP utilizzando la `LogMetrics` proprietà. Il formato del registro può essere specificato utilizzando la `LogMetricsFormat` proprietà. I valori validi sono JSON o standard.

Imposta l'endpoint della regione

Configura la regione predefinita per tutti i client di servizio come segue. :

```
AWSConfigs.AWSRegion="us-east-1";
```

Questo imposta la regione predefinita per tutti i client di servizio nell'SDK. Puoi ignorare questa impostazione specificando esplicitamente la regione al momento della creazione di un'istanza del client di servizio, come segue. :

```
IAmazonS3 s3Client = new AmazonS3Client(credentials,RegionEndpoint.USEast1);
```

Configurare le impostazioni del proxy HTTP

Se la rete è protetta da un proxy, è possibile configurare le impostazioni proxy per le richieste HTTP come segue.

```
var proxyConfig = AWSConfigs.ProxyConfig;
proxyConfig.Host = "localhost";
proxyConfig.Port = 80;
proxyConfig.Username = "<username>";
proxyConfig.Password = "<password>";
```

Corretto per Clock Skew

Questa proprietà determina se l'SDK deve correggere l'inclinazione dell'orologio del client determinando l'ora corretta del server e riemettendo la richiesta con l'ora corretta.

```
AWSConfigs.CorrectForClockSkew = true;
```

Questo campo viene impostato se una chiamata di servizio ha prodotto un'eccezione e l'SDK ha stabilito che esiste una differenza tra l'ora locale e quella del server.

```
var offset = AWSConfigs.ClockOffset;
```

Per ulteriori informazioni sull'inclinazione dell'orologio, consulta Clock-skew Correction sul blog di AWS.

Fasi successive

Ora che hai configurato l'SDK AWS Mobile per.NET and Xamarin, puoi:

- Inizia. Leggi Getting Started with the AWS Mobile SDK for .NET and Xamarin per istruzioni rapide su come usare e configurare i servizi nell'SDK AWS Mobile per .NET e Xamarin.

- Esplora gli argomenti del servizio. Scopri ogni servizio e come funziona nell'SDK AWS Mobile per .NET and Xamarin.
- Esegui le demo. Visualizza i nostri esempi di applicazioni Xamarin che illustrano casi d'uso comuni. Per eseguire le app di esempio, configura l'SDK AWS Mobile per .NET e Xamarin come descritto in precedenza, quindi segui le istruzioni contenute nei file README dei singoli esempi.
- Impara il. APIs Visualizza il | sdk-xamarin-ref |_.
- Fai domande: pubblica domande sui forum dell'SDK AWS Mobile o apri un problema su GitHub.

Guida introduttiva all'SDK AWS Mobile per.NET and Xamarin

L'SDK AWS Mobile per .NET and Xamarin fornisce le librerie, gli esempi e la documentazione necessari per richiamare i servizi AWS dalle applicazioni Xamarin.

È necessario completare tutte le istruzioni in Configurazione dell'SDK AWS Mobile per.NET and Xamarin prima di iniziare a utilizzare i servizi seguenti.

Questi argomenti introduttivi ti illustreranno quanto segue:

Argomenti

- Archivia e recupera file con Amazon S3
- Sincronizzazione dei dati utente con Cognito Sync
- Archivia e recupera dati con DynamoDB
- Monitoraggio dei dati di utilizzo delle app con Amazon Mobile Analytics
- Ricevi notifiche push tramite SNS (Xamarin iOS)
- Ricevi notifiche push tramite SNS (Xamarin Android)

Per informazioni su altri AWS Mobile SDKs, consulta AWS Mobile SDK.

Archivia e recupera file con Amazon S3

Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) fornisce agli sviluppatori mobili uno storage di oggetti sicuro, durevole e altamente scalabile. Amazon S3 è facile da usare, con una semplice interfaccia di servizi Web per archiviare e recuperare qualsiasi quantità di dati da qualsiasi punto del Web.

Il tutorial seguente spiega come integrare S3 TransferUtility, un'utilità di alto livello per l'utilizzo di S3 con la tua app. Per ulteriori informazioni sull'uso di S3 dalle applicazioni Xamarin, consulta Amazon Simple Storage Service (S3).

Configurazione del progetto

Prerequisiti

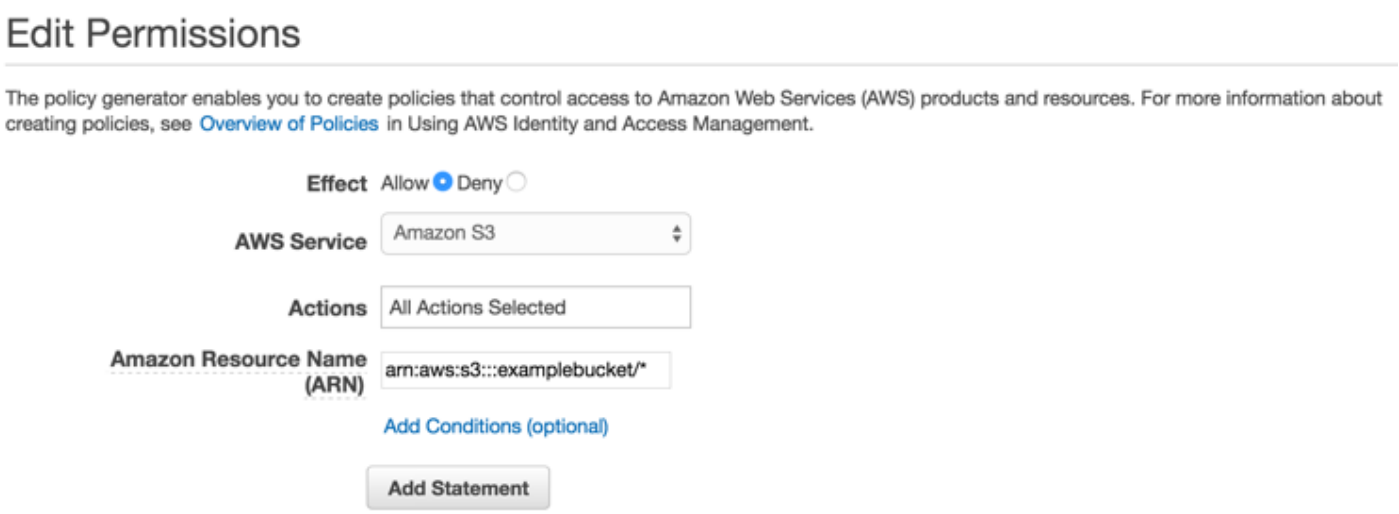
È necessario completare tutte le istruzioni sulla configurazione dell'SDK AWS Mobile per.NET and Xamarin prima di iniziare questo tutorial.

Questo tutorial presuppone inoltre che tu abbia già creato un bucket S3. Per creare un bucket S3, visita la console S3 AWS.

Imposta le autorizzazioni per S3

La policy di ruolo IAM predefinita consente all'applicazione di accedere ad Amazon Mobile Analytics e Amazon Cognito Sync. Affinché il pool di identità di Cognito possa accedere ad Amazon S3, è necessario modificare i ruoli del pool di identità.

1. Vai alla Identity and Access Management Console e fai clic su Ruoli nel riquadro a sinistra.
2. Digitare il nome del pool di identità nella casella di ricerca. Vengono elencati due ruoli, relativi, rispettivamente, agli utenti autenticati e non.
3. Fai clic sul ruolo per gli utenti non autenticati (verrà aggiunto unauth al nome del tuo pool di identità).
4. Fai clic su Crea politica sul ruolo, seleziona Policy Generator, quindi fai clic su Seleziona.
5. Nella pagina Modifica autorizzazioni, inserisci le impostazioni mostrate nell'immagine seguente, sostituendo l'Amazon Resource Name (ARN) con il tuo. L'ARN di un bucket S3 ha l'aspetto `arn:aws:s3:::examplebucket/*` ed è composto dalla regione in cui si trova il bucket e dal nome del bucket. Le impostazioni mostrate di seguito consentiranno al pool di identità di accedere a tutte le azioni per il bucket specificato.



1. Fai clic sul pulsante Aggiungi dichiarazione, quindi su Passaggio successivo.
2. La procedura guidata ti mostrerà la configurazione che hai generato. Fai clic su Applica politica.

Per ulteriori informazioni sulla concessione dell'accesso a S3, consulta Garantire l'accesso a un bucket Amazon S3.

Aggiungi NuGet Package for S3 al tuo progetto

Segui il passaggio 4 delle istruzioni in Configurazione dell'SDK AWS Mobile per.NET and Xamarin per NuGet aggiungere il pacchetto S3 al tuo progetto.

(opzionale) Configura la versione di firma per le richieste S3

Ogni interazione con Amazon S3 è autenticata o anonima. AWS utilizza gli algoritmi Signature Version 4 o Signature Version 2 per autenticare le chiamate al servizio.

Tutte le nuove regioni AWS create dopo gennaio 2014 supportano solo la versione Signature 4. Tuttavia, molte regioni precedenti supportano ancora le richieste Signature Version 4 e Signature Version 2.

Se il tuo bucket si trova in una delle regioni che non supportano le richieste Signature Version 2 elencate in questa pagina, devi impostare AWSConfigs S3. UseSignatureVersion4 proprietà su «true» in questo modo:

```
AWSConfigsS3.UseSignatureVersion4 = true;
```

Per ulteriori informazioni sulle versioni di AWS Signature, consulta Authenticating Requests (AWS Signature versione 4).

Inizializza il client S3 TransferUtility

Crea un client S3, passandogli l'oggetto delle credenziali AWS, quindi passa il client S3 all'utilità di trasferimento, in questo modo:

```
var s3Client = new AmazonS3Client(credentials,region);
var transferUtility = new TransferUtility(s3Client);
```

Caricare un file su Amazon S3

Per caricare un file `Upload` su S3, richiama l'oggetto Transfer Utility, passando i seguenti parametri:

- `file`- Nome stringa del file che vuoi caricare
- `bucketName`- Nome stringa del bucket S3 per archiviare il file

```
transferUtility.Upload(
  Path.Combine(Environment.SpecialFolder.ApplicationData,"file"),
  "bucketName"
);
```

Il codice precedente presuppone che ci sia un file nella directory Environment. SpecialFolder. ApplicationData. I caricamenti utilizzano automaticamente la funzionalità di caricamento in più parti di S3 su file di grandi dimensioni per migliorare la velocità di trasmissione.

Scarica un file da Amazon S3

Per scaricare un file da S3, `Download` richiamate l'oggetto Transfer Utility, passando i seguenti parametri:

- `file`- Nome stringa del file che vuoi scaricare
- `bucketName`- Nome stringa del bucket S3 da cui si desidera scaricare il file
- `key`- Una stringa che rappresenta il nome dell'oggetto S3 (un file in questo caso) da scaricare

```
transferUtility.Download(
```

```
  Path.Combine(Environment.SpecialFolder.ApplicationData,"file"),
  "bucketName",
  "key"
);
```

Per ulteriori informazioni sull'accesso ad Amazon S3 da un'applicazione Xamarin, consulta Amazon Simple Storage Service (S3).

Sincronizzazione dei dati utente con Cognito Sync

Amazon Cognito Sync semplifica il salvataggio dei dati degli utenti mobili, come le preferenze delle app o lo stato del gioco, nel cloud AWS senza scrivere alcun codice di backend o gestire alcuna infrastruttura. Puoi salvare i dati localmente sui dispositivi degli utenti, permettendo alle applicazioni di funzionare anche quando i dispositivi sono offline. Puoi anche sincronizzare i dati tra i dispositivi di un utente in modo che la sua esperienza con l'app sia coerente indipendentemente dal dispositivo che utilizza.

Il tutorial seguente spiega come integrare Sync con la tua app.

Configurazione del progetto

Prerequisiti

È necessario completare tutte le istruzioni sulla configurazione dell'SDK AWS Mobile per.NET and Xamarin prima di iniziare questo tutorial.

Concedi l'accesso alle tue risorse di Cognito Sync

La politica predefinita associata ai ruoli non autenticati e di autenticazione che hai creato durante la configurazione concede all'applicazione l'accesso a Cognito Sync. Non è richiesta alcuna configurazione aggiuntiva.

Aggiungi un NuGet pacchetto per Cognito Sync al tuo progetto

Segui il passaggio 4 delle istruzioni in Configurazione dell'SDK AWS Mobile per.NET and Xamarin per SyncManager NuGet aggiungere il pacchetto Cognito al tuo progetto.

Inizializza il CognitoSyncManager

Bisogna passare il provider di credenziali Amazon Cognito inizializzato al costruttore `CognitoSyncManager`:

```
CognitoSyncManager syncManager = new CognitoSyncManager (
    credentials,
    new AmazonCognitoSyncConfig {
        RegionEndpoint = RegionEndpoint.USEast1 // Region
    }
);
```

Sincronizzazione dei dati utente

Per sincronizzare i dati utente non autenticati:

1. Crea un set di dati.
2. Aggiungi i dati utente al set di dati.
3. Sincronizza il set di dati con il cloud.

Creare un set di dati

Creare un'istanza di `Dataset`. Il metodo openOrCreate Dataset viene utilizzato per creare un nuovo set di dati o aprire un'istanza esistente di un set di dati archiviato localmente sul dispositivo:

```
Dataset dataset = syncManager.OpenOrCreateDataset("myDataset");
```

Aggiungi dati utente al set di dati

I dati utente vengono aggiunti sotto forma di key/value coppie:

```
dataset.OnSyncSuccess += SyncSuccessCallback;
dataset.Put("myKey", "myValue");
```

I set di dati Cognito funzionano come dizionari, con valori accessibili tramite chiave:

```
string myValue = dataset.Get("myKey");
```

Sincronizza set di dati

Per sincronizzare un set di dati, chiama il relativo metodo synchronize:

```
dataset.SynchronizeAsync();
```

```
void SyncSuccessCallback(object sender, SyncSuccessEventArgs e) {
  // Your handler code here
}
```

Tutti i dati scritti nei set di dati verranno archiviati localmente fino alla sincronizzazione del set di dati. Il codice in questa sezione presuppone che tu stia utilizzando un'identità Cognito non autenticata, quindi quando i dati dell'utente vengono sincronizzati con il cloud verranno archiviati per dispositivo. Al dispositivo è associato un ID del dispositivo. Quando i dati dell'utente vengono sincronizzati con il cloud, verranno associati all'ID del dispositivo.

Per ulteriori informazioni su Cognito Sync, consulta Amazon Cognito Sync.

Archivia e recupera dati con DynamoDB

Amazon DynamoDB è un servizio di database non relazionale, conveniente, veloce e altamente scalabile e disponibile. DynamoDB rimuove le tradizionali limitazioni di scalabilità sullo storage dei dati mantenendo una bassa latenza e prestazioni prevedibili.

Il tutorial seguente spiega come integrare il DynamoDB Object Persistence Model con l'app, che archivia gli oggetti in DynamoDB.

Configurazione del progetto

Prerequisiti

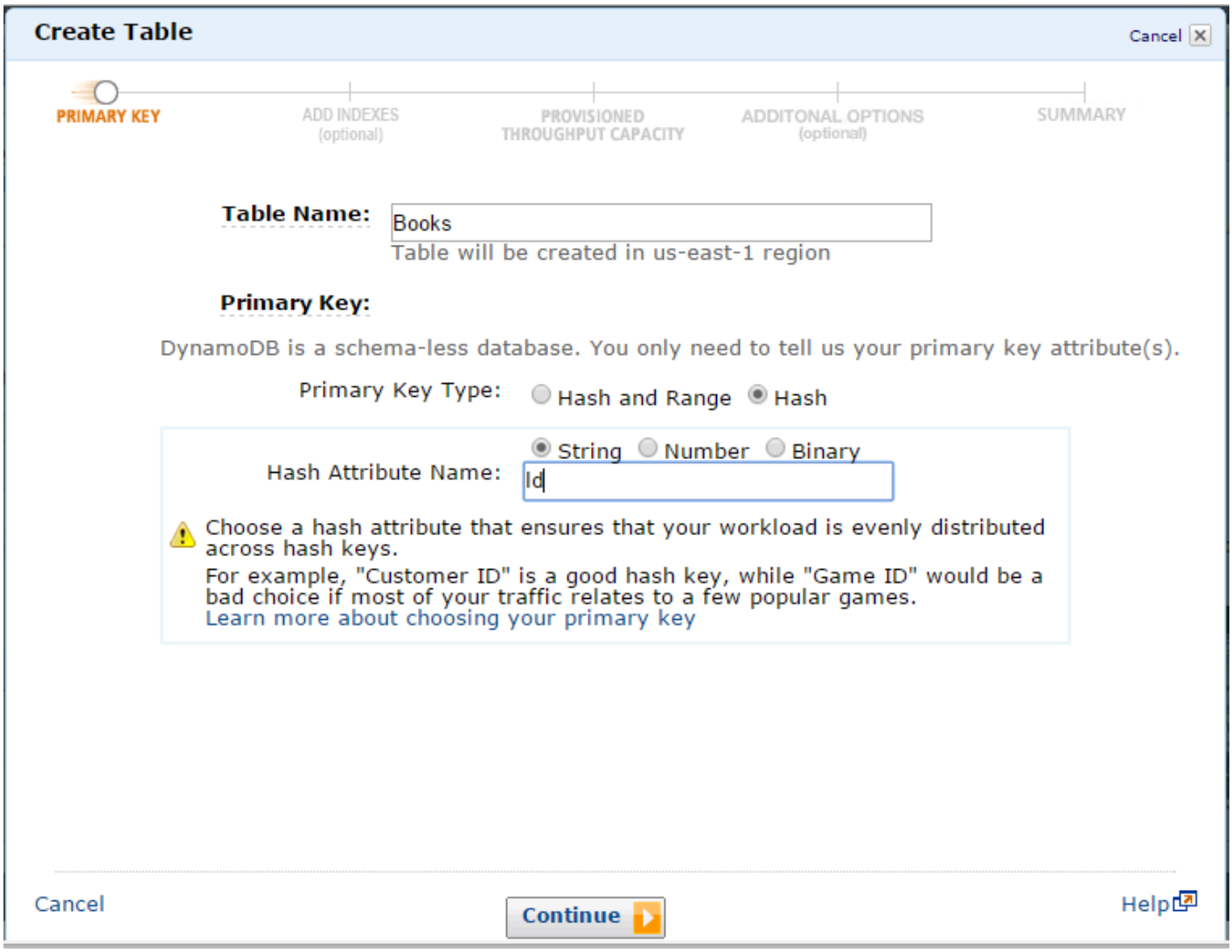
È necessario completare tutte le istruzioni sulla configurazione dell'SDK AWS Mobile per.NET and Xamarin prima di iniziare questo tutorial.

Creare una tabella DynamoDB

Prima di poter leggere e scrivere dati su un database DynamoDB, è necessario creare una tabella. Quando si crea una tabella è necessario specificare la chiave primaria. La chiave primaria è composta da un attributo hash e da un attributo range opzionale. Per ulteriori informazioni su come vengono utilizzati gli attributi primari e di intervallo, vedere Lavorare con le tabelle.

1. Vai alla console DynamoDB e fai clic su Crea tabella. Viene visualizzata la procedura guidata Crea tabella.

2. Specificate il nome della tabella, il tipo di chiave primaria (Hash) e il nome dell'attributo hash («Id») come mostrato di seguito, quindi fate clic su Continua:

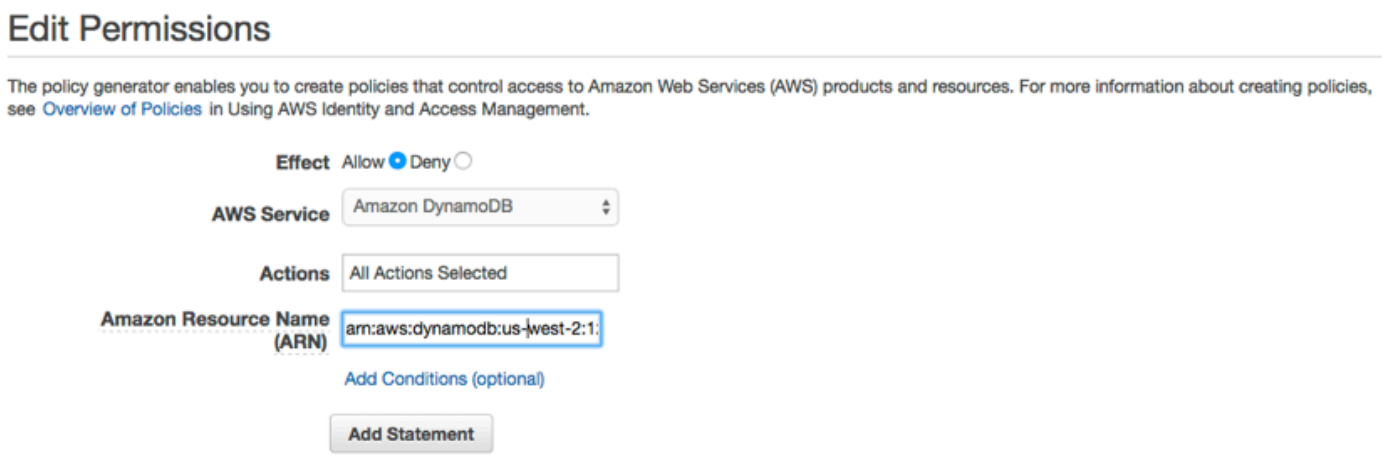


3. Lascia vuoti i campi di modifica nella schermata successiva e fai clic su Continua.
4. Accettate i valori predefiniti per le unità di capacità di lettura e le unità di capacità di scrittura e fate clic su Continua.
5. Nella schermata successiva, inserisci il tuo indirizzo e-mail nella casella di testo Invia notifica a: e fai clic su Continua. Viene visualizzata la schermata di revisione.
6. Fai clic su Create (Crea). La creazione della tabella potrebbe richiedere alcuni minuti.

Impostazione delle autorizzazioni per DynamoDB

Affinché il tuo pool di identità possa accedere ad Amazon DynamoDB, devi modificare i ruoli del pool di identità.

1. Vai alla Identity and Access Management Console e fai clic su Ruoli nel riquadro a sinistra. Cerca il nome del tuo pool di identità: verranno elencati due ruoli, uno per gli utenti non autenticati e uno per gli utenti autenticati.
2. Fai clic sul ruolo per gli utenti non autenticati (al nome del pool di identità verrà aggiunto «unauth») e fai clic su Crea politica di ruolo.
3. Seleziona Policy Generator e fai clic su Seleziona.
4. Nella pagina Modifica autorizzazioni, inserisci le impostazioni mostrate nell'immagine seguente. L'Amazon Resource Name (ARN) di una tabella DynamoDB ha l'aspetto `arn:aws:dynamodb:us-west-2:123456789012:table/Books` ed è composto dalla regione in cui si trova la tabella, dal numero di account AWS del proprietario e dal nome della tabella nel formato. `table/Books` Per ulteriori informazioni sulla specificazione ARNs, consulta Amazon Resource Names for DynamoDB.



5. Fai clic su Aggiungi dichiarazione, quindi su Passaggio successivo. La procedura guidata ti mostrerà la configurazione generata.
6. Fai clic su Applica politica.

Aggiungi un NuGet pacchetto per DynamoDB al tuo progetto

Segui il passaggio 4 delle istruzioni in Configurazione dell'SDK AWS Mobile per.NET and Xamarin per NuGet aggiungere il pacchetto DynamoDB al tuo progetto.

Inizializza AmazonDynamo DBClient

Passa il provider di credenziali Amazon Cognito inizializzato e la tua regione al costruttore, quindi passa `AmazonDynamoDB` il client a Dynamo: DBContext

```
var client = new AmazonDynamoDBClient(credentials,region);
DynamoDBContext context = new DynamoDBContext(client);
```

Crea una classe

Per scrivere una riga nella tabella, definite una classe che contenga i dati della riga. La classe deve contenere anche proprietà che contengono i dati degli attributi per la riga e verranno mappate alla tabella DynamoDB creata nella console. La seguente dichiarazione di classe illustra tale classe:

```
[DynamoDBTable("Books")]
public class Book
{
    [DynamoDBHashKey]    // Hash key.
    public int Id { get; set; }
    public string Title { get; set; }
    public string ISBN { get; set; }
    public int Price { get; set; }
    public string PageCount { get; set; }
    public string Author{ get; set; }
}
```

Salvataggio di una voce

Per salvare un elemento, crea prima un oggetto:

```
Book songOfIceAndFire = new Book()
{
    Id=1,
    Title="Game Of Thrones",
    ISBN="978-0553593716",
    Price=4,
    PageCount="819",
    Author="GRRM"
};
```

Quindi salvalo:

```
context.Save(songOfIceAndFire);
```

Per aggiornare una riga, modifica l'istanza della `DDTableRow` classe e chiama `AWSDynamoObjectMapper.save()` come mostrato sopra.

Recupero di una voce

Recupera un elemento utilizzando una chiave primaria:

```
Book retrievedBook = context.Load<Book>(1);
```

Aggiornamento di una voce

Per aggiornare un elemento:

```
Book retrievedBook = context.Load<Book>(1);
retrievedBook.ISBN = "978-0553593716";
context.Save(retrievedBook);
```

Eliminazione di una voce

Per eliminare un elemento:

```
Book retrievedBook = context.Load<Book>(1);
context.Delete(retrievedBook);
```

Per ulteriori informazioni sull'accesso a DynamoDB da un'applicazione Xamarin, consulta Amazon DynamoDB.

Monitoraggio dei dati di utilizzo delle app con Amazon Mobile Analytics

Amazon Mobile Analytics ti consente di misurare l'utilizzo e i ricavi delle app. Monitorando le tendenze chiave, come gli utenti nuovi e quelli abituali, le entrate delle app, la fidelizzazione degli utenti e gli eventi comportamentali personalizzati all'interno dell'app, puoi prendere decisioni basate sui dati per aumentare il coinvolgimento e la monetizzazione della tua app.

Il tutorial seguente spiega come integrare Mobile Analytics con la tua app.

Configurazione del progetto

Prerequisiti

È necessario completare tutte le istruzioni sulla configurazione dell'SDK AWS Mobile per.NET and Xamarin prima di iniziare questo tutorial.

Creare un'app nella Mobile Analytics Console

Vai alla console Amazon Mobile Analytics e crea un'app. Annota il `appId` valore, poiché ti servirà in seguito. Quando crei un'app nella Mobile Analytics Console, devi specificare l'ID del tuo pool di identità. Per istruzioni sulla creazione di un pool di identità, consulta Configurazione dell'SDK AWS Mobile per.NET and Xamarin.

Per ulteriori informazioni su come lavorare nella console, consulta la Amazon Mobile Analytics User Guide.

Imposta le autorizzazioni per Mobile Analytics

La politica predefinita associata ai ruoli che hai creato durante la configurazione concede all'applicazione l'accesso a Mobile Analytics. Non è richiesta alcuna configurazione aggiuntiva.

Aggiungi NuGet Package for Mobile Analytics al tuo progetto

Segui il passaggio 4 delle istruzioni in Configurazione dell'SDK AWS Mobile per.NET and Xamarin per aggiungere il pacchetto Mobile NuGet Analytics al tuo progetto.

Configurazione delle impostazioni di Mobile Analytics

Mobile Analytics definisce alcune impostazioni che possono essere configurate nel file awsconfig.xml:

```
var config = new MobileAnalyticsManagerConfig();
config.AllowUseDataNetwork = true;
config.DBWarningThreshold = 0.9f;
config.MaxDBSize = 5242880;
config.MaxRequestSize = 102400;
config.SessionTimeout = 5;
```

- AllowUseDataNetwork - Un valore booleano che specifica se gli eventi della sessione vengono inviati sulla rete dati.

- DBWarningSoglia - Questo è il limite alla dimensione del database che, una volta raggiunto, genererà dei log di avviso.
- Max DBSize - Questa è la dimensione del SQLIte database. Quando il database raggiunge la dimensione massima, tutti gli eventi aggiuntivi vengono eliminati.
- MaxRequestSize - Questa è la dimensione massima della richiesta in byte che deve essere trasmessa in una richiesta HTTP al servizio di analisi mobile.
- SessionTimeout - Questo è l'intervallo di tempo dopo che un'applicazione passa in background e quando la sessione può essere terminata.

Le impostazioni mostrate sopra sono i valori predefiniti per ogni elemento di configurazione.

Inizializza MobileAnalyticsManager

Per inizializzare MobileAnalyticsManager, richiama GetOrCreateInstance le tue credenziali AWS`MobileAnalyticsManager`, la tua regione, l'ID dell'applicazione Mobile Analytics e il tuo oggetto di configurazione opzionale:

```
var manager = MobileAnalyticsManager.GetOrCreateInstance(
  "APP_ID",
  "Credentials",
  "RegionEndPoint",
  config
);
```

Tieni traccia degli eventi della sessione

Xamarin Android

Sostituisci le attività `OnPause()` e i `OnResume()` metodi per registrare gli eventi della sessione.

```
protected override void OnResume()
{
    manager.ResumeSession();
    base.OnResume();
}

protected override void OnPause()
{
    manager.PauseSession();
```

```
    base.OnPause();
}
```

Questo deve essere implementato per ogni attività dell'applicazione.

Xamarin iOS

Nel tuo .cs AppDelegate:

```
public override void DidEnterBackground(UIApplication application)
{
    manager.PauseSession();
}

public override void WillEnterForeground(UIApplication application)
{
    manager.ResumeSession();
}
```

Per ulteriori informazioni su Mobile Analytics, consulta Amazon Mobile Analytics.

Ricevi notifiche push tramite SNS (Xamarin iOS)

Questo documento spiega come inviare notifiche push a un'applicazione Xamarin iOS utilizzando Amazon Simple Notification Service (SNS) e l'SDK AWS Mobile per .NET and Xamarin.

Configurazione del progetto

Prerequisiti

È necessario completare tutte le istruzioni sulla configurazione dell'SDK AWS Mobile per.NET and Xamarin prima di iniziare questo tutorial.

Imposta le autorizzazioni per SNS

Segui il passaggio 2 in Configurazione dell'SDK AWS Mobile per.NET and Xamarin per allegare la policy menzionata di seguito ai ruoli dell'applicazione. Questo darà alla tua applicazione le autorizzazioni appropriate per accedere a SNS:

1. Vai alla console IAM e seleziona il ruolo IAM che desideri configurare.

2. Fai clic su Allega politica, seleziona la politica di Amazon SNSFull Access e fai clic su Allega politica.

> **Warning**
>
> L'uso di Amazon SNSFull Access non è consigliato in un ambiente di produzione. Lo utilizziamo qui per consentirti di essere subito operativo. Per ulteriori informazioni sulla specificazione delle autorizzazioni per un ruolo IAM, consulta Panoramica delle autorizzazioni dei ruoli IAM.

Ottieni l'iscrizione all'Apple iOS Developer Program

Dovrai eseguire l'app su un dispositivo fisico per ricevere notifiche push. Per eseguire l'app su un dispositivo, devi essere iscritto all'Apple iOS Developer Program Membership. Una volta sottoscritta l'iscrizione, puoi usare Xcode per generare un'identità di firma. Per ulteriori informazioni, consulta la documentazione Quick Start di App Distribution di Apple.

Crea un certificato iOS

Innanzitutto, devi creare un certificato iOS. Quindi, è necessario creare un profilo di provisioning configurato per le notifiche push. A tale scopo:

1. Vai all'Apple Developer Member Center, fai clic su Certificati, identificatori e profili.
2. Fai clic su Identificatori in App iOS, fai clic sul pulsante più nell'angolo in alto a destra della pagina Web per aggiungere un nuovo ID app iOS e inserisci una descrizione dell'ID app.
3. Scorri verso il basso fino alla sezione Aggiungi suffisso ID, seleziona Explicit App ID e inserisci l'identificatore del pacchetto.
4. Scorri verso il basso fino alla sezione Servizi app e seleziona Notifiche push.
5. Fai clic su Continue (Continua).
6. Fare clic su Submit (Invia).
7. Fai clic su Fine.
8. Seleziona l'ID app che hai appena creato, quindi fai clic su Modifica.
9. Scorri verso il basso fino alla sezione Notifiche push. Fai clic su Crea certificato in Certificato SSL di sviluppo.

10. Segui le istruzioni per creare una richiesta di firma del certificato (CSR), carica la richiesta e scarica un certificato SSL che verrà utilizzato per comunicare con Apple Notification Service (APNS).
11. Torna alla pagina Certificati, identificatori e profili. Fai clic su Tutto in Provisioning Profiles.
12. Fai clic sul pulsante più nell'angolo in alto a destra per aggiungere un nuovo profilo di provisioning.
13. Seleziona Sviluppo app iOS, quindi fai clic su Continua.
14. Seleziona l'ID dell'app, quindi fai clic su Continua.
15. Seleziona il certificato per sviluppatori, quindi fai clic su Continua.
16. Seleziona il tuo dispositivo, quindi fai clic su Continua.
17. Inserisci un nome di profilo, quindi fai clic su Genera.
18. Scaricate e fate doppio clic sul file di provisioning per installare il profilo di provisioning.

Per ulteriori informazioni sul provisioning di un profilo configurato per le notifiche push, consulta la documentazione sulla configurazione delle notifiche push di Apple.

Usa il certificato per creare l'ARN della piattaforma nella console SNS

1. Esegui l'app di KeyChain accesso, seleziona I miei certificati nella parte inferiore sinistra dello schermo, quindi fai clic con il pulsante destro del mouse sul certificato SSL che hai generato per la connessione ad APNS e seleziona Esporta. Ti verrà richiesto di specificare un nome per il file e una password per proteggere il certificato. Il certificato verrà salvato in un file P12.
2. Vai alla console SNS e fai clic su Applicazioni sul lato sinistro dello schermo.
3. Fai clic su Crea applicazione di piattaforma per creare una nuova applicazione di piattaforma SNS.
4. Inserisci un nome per l'applicazione.
5. Seleziona Apple Development per la piattaforma di notifica push.
6. Fai clic su Scegli file e seleziona il file P12 che hai creato quando hai esportato il certificato SSL.
7. Inserisci la password che hai specificato quando hai esportato il certificato SSL e fai clic su Carica credenziali dal file.
8. Fai clic su Crea applicazione di piattaforma.
9. Seleziona l'applicazione della piattaforma che hai appena creato e copia l'ARN dell'applicazione. Ne avrai bisogno nei prossimi passaggi.

Aggiungi un NuGet pacchetto per SNS al tuo progetto

Segui il passaggio 4 delle istruzioni in Configurazione dell'SDK AWS Mobile per.NET and Xamarin per aggiungere il pacchetto NuGet Amazon Simple Notification Service al tuo progetto.

Crea un client SNS

```
var snsClient = new AmazonSimpleNotificationServiceClient(credentials, region);
```

Registra la tua applicazione per le notifiche remote

Per registrare un'applicazione, richiamate RegisterForRemoteNotifications il vostro UIApplication oggetto, come illustrato di seguito. Inserisci il codice seguente in AppDelegate .cs, inserendo l'ARN dell'applicazione della piattaforma dove richiesto di seguito:

```
public override bool FinishedLaunching(UIApplication app, NSDictionary options) {
// do something
var pushSettings = UIUserNotificationSettings.GetSettingsForTypes (
  UIUserNotificationType.Alert |
  UIUserNotificationType.Badge |
  UIUserNotificationType.Sound,
  null
);
app.RegisterUserNotifications(pushSettings);
app.RegisterForRemoteNotifications();
// do something
  return true;
}

public override void RegisteredForRemoteNotifications(UIApplication application, NSData
 token) {
  var deviceToken = token.Description.Replace("<", "").Replace(">", "").Replace(" ",
 "");
  if (!string.IsNullOrEmpty(deviceToken)) {
    //register with SNS to create an endpoint ARN
    var response = await SnsClient.CreatePlatformEndpointAsync(
    new CreatePlatformEndpointRequest {
      Token = deviceToken,
      PlatformApplicationArn = "YourPlatformArn" /* insert your platform application
 ARN here */
    });
```

```
    }
}
```

Invia un messaggio dalla console SNS al tuo endpoint

1. Vai alla console SNS > Applicazioni.
2. Seleziona l'applicazione della piattaforma, seleziona un endpoint e fai clic su Pubblica sull'endpoint.
3. Digita un messaggio di testo nella casella di testo e fai clic su Pubblica messaggio per pubblicare un messaggio.

Ricevi notifiche push tramite SNS (Xamarin Android)

Il tutorial spiega come inviare notifiche push a un'applicazione Xamarin Android utilizzando Amazon Simple Notification Service (SNS) e l'SDK AWS Mobile per .NET and Xamarin.

Configurazione del progetto

Prerequisiti

È necessario completare tutte le istruzioni sulla configurazione dell'SDK AWS Mobile per.NET and Xamarin prima di iniziare questo tutorial.

Imposta le autorizzazioni per SNS

Segui il passaggio 2 in Configurazione dell'SDK AWS Mobile per.NET and Xamarin per allegare la policy menzionata di seguito ai ruoli dell'applicazione. Ciò fornirà all'applicazione le autorizzazioni appropriate per accedere a SNS:

1. Vai alla console IAM e seleziona il ruolo IAM che desideri configurare.
2. Fai clic su Allega politica, seleziona la politica di Amazon SNSFull Access e fai clic su Allega politica.

> **Warning**
>
> L'uso di Amazon SNSFull Access non è consigliato in un ambiente di produzione. Lo utilizziamo qui per consentirti di essere subito operativo. Per ulteriori informazioni sulla

specificazione delle autorizzazioni per un ruolo IAM, consulta Panoramica delle autorizzazioni dei ruoli IAM.

Abilita le notifiche push su Google Cloud

Innanzitutto, aggiungi un nuovo progetto API di Google:

1. Vai alla Google Developers Console.
2. Fai clic su Crea progetto.
3. Nella casella Nuovo progetto, inserisci il nome del progetto, prendi nota dell'ID del progetto (ti servirà in seguito) e fai clic su Crea.

Quindi, abilita il servizio Google Cloud Messaging (GCM) per il tuo progetto:

1. Nella Google Developers Console, il tuo nuovo progetto dovrebbe essere già selezionato. In caso contrario, selezionalo nel menu a discesa nella parte superiore della pagina.
2. Seleziona APIs & auth dalla barra laterale sul lato sinistro della pagina.
3. Nella casella di ricerca, digita «Google Cloud Messaging per Android» e fai clic sul link Google Cloud Messaging per Android.
4. Fai clic su Abilita API.

Infine, ottieni una chiave API:

1. Nella Google Developers Console, seleziona APIs & auth > Credentials.
2. In Accesso pubblico all'API, fai clic su Crea nuova chiave.
3. Nella finestra di dialogo Crea una nuova chiave, fai clic su Chiave server.
4. Nella finestra di dialogo visualizzata, fai clic su Crea e copia la chiave API visualizzata. Utilizzerai questa chiave API per eseguire l'autenticazione in un secondo momento.

Usa l'ID del progetto per creare un ARN di piattaforma nella console SNS

1. Vai alla console SNS.
2. Fai clic su Applicazioni sul lato sinistro dello schermo.
3. Fai clic su Crea applicazione di piattaforma per creare una nuova applicazione di piattaforma SNS.

4. Inserisci un nome per l'applicazione.
5. Seleziona Google Cloud Messaging (GCM) per la piattaforma di notifica push.
6. Incolla la chiave API nella casella di testo denominata Chiave API.
7. Fai clic su Crea applicazione della piattaforma.
8. Seleziona l'applicazione della piattaforma che hai appena creato e copia l'ARN dell'applicazione.

Aggiungi un NuGet pacchetto per SNS al tuo progetto

Segui il passaggio 4 delle istruzioni in Configurazione dell'SDK AWS Mobile per.NET and Xamarin per aggiungere il pacchetto NuGet Amazon Simple Notification Service al tuo progetto.

Crea un client SNS

```
var snsClient = new AmazonSimpleNotificationServiceClient(credentials, region);
```

Registra la tua applicazione per le notifiche remote

Per registrarti alle notifiche remote su Android, dovrai crearne uno BroadcastReceiver che possa ricevere messaggi di Google Cloud. Cambia il nome del pacchetto qui sotto dove richiesto:

```
[BroadcastReceiver(Permission = "com.google.android.c2dm.permission.SEND")]
[IntentFilter(new string[] {
      "com.google.android.c2dm.intent.RECEIVE"
}, Categories = new string[] {
      "com.amazonaws.sns" /* change to match your package */
})]
[IntentFilter(new string[] {
      "com.google.android.c2dm.intent.REGISTRATION"
}, Categories = new string[] {
      "com.amazonaws.sns" /* change to match your package */
})]
[IntentFilter(new string[] {
      "com.google.android.gcm.intent.RETRY"
}, Categories = new string[] {
      "com.amazonaws.sns" /* change to match your package */
})]
public class GCMBroadcastReceiver: BroadcastReceiver {
      const string TAG = "PushHandlerBroadcastReceiver";
      public override void OnReceive(Context context, Intent intent) {
```

```
                GCMIntentService.RunIntentInService(context, intent);
                SetResult(Result.Ok, null, null);
        }
}

[BroadcastReceiver]
[IntentFilter(new[] {
        Android.Content.Intent.ActionBootCompleted
})]
public class GCMBootReceiver: BroadcastReceiver {
        public override void OnReceive(Context context, Intent intent) {
                GCMIntentService.RunIntentInService(context, intent);
                SetResult(Result.Ok, null, null);
        }
}
```

Di seguito è riportato il servizio che riceve la notifica push da BroadcastReceiver e la visualizza sulla barra delle notifiche del dispositivo:

```
[Service]
 public class GCMIntentService: IntentService {
  static PowerManager.WakeLock sWakeLock;
  static object LOCK = new object();

  public static void RunIntentInService(Context context, Intent intent) {
    lock(LOCK) {
      if (sWakeLock == null) {
        // This is called from BroadcastReceiver, there is no init.
        var pm = PowerManager.FromContext(context);
        sWakeLock = pm.NewWakeLock(
        WakeLockFlags.Partial, "My WakeLock Tag");
      }
    }

    sWakeLock.Acquire();
    intent.SetClass(context, typeof(GCMIntentService));
    context.StartService(intent);
  }

  protected override void OnHandleIntent(Intent intent) {
    try {
      Context context = this.ApplicationContext;
      string action = intent.Action;
```

```
      if (action.Equals("com.google.android.c2dm.intent.REGISTRATION")) {
        HandleRegistration(intent);
      } else if (action.Equals("com.google.android.c2dm.intent.RECEIVE")) {
        HandleMessage(intent);
      }
    } finally {
      lock(LOCK) {
        //Sanity check for null as this is a public method
        if (sWakeLock != null) sWakeLock.Release();
      }
    }
  }

  private void HandleRegistration(Intent intent) {
    string registrationId = intent.GetStringExtra("registration_id");
    string error = intent.GetStringExtra("error");
    string unregistration = intent.GetStringExtra("unregistered");

    if (string.IsNullOrEmpty(error)) {
      var response = await SnsClient.CreatePlatformEndpointAsync(new
CreatePlatformEndpointRequest {
        Token = registrationId,
        PlatformApplicationArn = "YourPlatformArn" /* insert your platform application
ARN here */
      });
    }
  }

  private void HandleMessage(Intent intent) {
    string message = string.Empty;
    Bundle extras = intent.Extras;
    if (!string.IsNullOrEmpty(extras.GetString("message"))) {
      message = extras.GetString("message");
    } else {
      message = extras.GetString("default");
    }

    Log.Info("Messages", "message received = " + message);
    ShowNotification(this, "SNS Push", message);
    //show the message

  }
```

```
  public void ShowNotification(string contentTitle,
  string contentText) {
    // Intent
    Notification.Builder builder = new Notification.Builder(this)
      .SetContentTitle(contentTitle)
      .SetContentText(contentText)
      .SetDefaults(NotificationDefaults.Sound | NotificationDefaults.Vibrate)
      .SetSmallIcon(Resource.Drawable.Icon)
      .SetSound(RingtoneManager.GetDefaultUri(RingtoneType.Notification));

    // Get the notification manager:
    NotificationManager notificationManager =
 this.GetSystemService(Context.NotificationService) as NotificationManager;

    notificationManager.Notify(1001, builder.Build());
  }
}
```

Invia un messaggio dalla console SNS al tuo endpoint

1. Vai alla console SNS > Applicazioni.
2. Seleziona l'applicazione della piattaforma, seleziona un endpoint e fai clic su Pubblica sull'endpoint.
3. Digita un messaggio di testo nella casella di testo e fai clic su Pubblica messaggio per pubblicare un messaggio.

Identità Amazon Cognito

Cos'è Amazon Cognito Identity?

Amazon Cognito Identity ti consente di creare identità uniche per i tuoi utenti e di autenticarli con provider di identità. Con un'identità, puoi ottenere credenziali AWS temporanee con privilegi limitati per sincronizzare i dati con Amazon Cognito Sync o accedere direttamente ad altri servizi AWS. Amazon Cognito Identity supporta i provider di identità pubblici (Amazon, Facebook e Google) e le identità non autenticate. Questa funzione supporta anche identità non autenticate per gli sviluppatori e ti consente di registrare e di autenticare gli utenti tramite il tuo processo di autenticazione di back-end.

Per ulteriori informazioni su Cognito Identity, consulta la Amazon Cognito Developer Guide.

Per informazioni sulla disponibilità della regione di autenticazione Cognito, consulta AWS Service Region Availability.

Utilizzo di un provider pubblico per autenticare gli utenti

Utilizzando Amazon Cognito Identity, puoi creare identità uniche per i tuoi utenti e autenticarli per un accesso sicuro alle tue risorse AWS come Amazon S3 o Amazon DynamoDB. Amazon Cognito Identity supporta provider di identità pubblici (Amazon, Facebook, Twitter/Digits, Google o qualsiasi provider compatibile con OpenID Connect) oltre a identità non autenticate.

Per informazioni sull'utilizzo di provider di identità pubblici come Amazon, Facebook, Twitter/Digits o Google per autenticare gli utenti, consulta i provider esterni nella Amazon Cognito Developer Guide.

Utilizzo di identità autenticate dagli sviluppatori

Amazon Cognito supporta le identità autenticate degli sviluppatori, oltre alla federazione delle identità Web tramite Facebook, Google e Amazon. Con le identità autenticate dagli sviluppatori, puoi registrare e autenticare gli utenti tramite il tuo processo di autenticazione esistente, continuando a utilizzare Amazon Cognito Sync per sincronizzare i dati degli utenti e accedere alle risorse AWS. L'utilizzo di identità autenticate dagli sviluppatori prevede l'interazione tra il dispositivo dell'utente finale, il back-end per l'autenticazione e Amazon Cognito.

Per informazioni sulle identità autenticate degli sviluppatori, consulta la Developer Authenticated Identities nella Amazon Cognito Developer Guide.

Sincronizzazione con Amazon Cognito

Cos'è Amazon Cognito Sync?

Cognito Sync è un servizio AWS e una libreria client che consente la sincronizzazione tra dispositivi dei dati degli utenti (ad esempio punteggi di gioco, preferenze utente, stato del gioco). Puoi utilizzare l'API Cognito Sync per sincronizzare i dati degli utenti tra i dispositivi. Per utilizzare Cognito Sync nella tua app, devi includere | nel tuo progetto.

Per istruzioni su come integrare Amazon Cognito Sync nella tua applicazione, consulta la Amazon Cognito Sync Developer Guide.

Amazon Mobile Analytics

Amazon Mobile Analytics è un servizio per la raccolta, la visualizzazione, la comprensione e l'estrazione di dati sull'utilizzo delle app su larga scala. Mobile Analytics acquisisce facilmente sia i dati standard del dispositivo che gli eventi personalizzati e calcola automaticamente i report per tuo conto. Oltre ai report aggregati elencati di seguito, puoi anche configurare i tuoi dati in modo che vengano esportati automaticamente su Redshift e S3 per ulteriori analisi.

Utilizzando Amazon Mobile Analytics, puoi monitorare i comportamenti dei clienti, aggregare metriche, generare visualizzazioni di dati e identificare modelli significativi.

Concetti chiave

Tipi di report

Per impostazione predefinita, Mobile Analytics fornisce i seguenti report nella Mobile Analytics Console:

- Daily Active Users (DAU, Utenti attivi giornalieri), Monthly Active Users (MAU, Utenti attivi mensili) e New Users (Nuovi utenti)
- Sticky Factor (Fattore sticky) (DAU diviso MAU)
- Session Count (Conteggio sessioni) e Average Sessions per Daily Active User (Sessioni medie per utente attivo giornaliero)
- Entrate medie per utente attivo giornaliero (ARPDAU) e entrate medie per utente attivo pagante giornaliero (ARPPDAU)
- Day 1, 3, and 7 Retention (Conservazione giorno 1, 3 e 7) e Week 1, 2, and 3 Retention (Conservazione settimana 1, 2 e 3)
- Eventi personalizzati

Questi report vengono forniti tramite sei schede di reporting nella console:

- Panoramica: tieni traccia di nove report preselezionati in una simple-to-review dashboard per farti un'idea rapida del coinvolgimento: MAU, DAU, New Users, Daily Sessions, Sticky Factor, 1-Day Retention, ARPDAU, Daily Paying Users, ARPPDAU.
- Utenti attivi: monitora quanti utenti interagiscono con la tua app ogni giorno e ogni mese e monitora lo sticky factor per valutare il coinvolgimento, l'appeal e la monetizzazione.

- Sessioni: tieni traccia della frequenza di utilizzo dell'app in un determinato giorno e della frequenza con cui ogni utente la apre durante un giorno.
- Fidelizzazione: monitora la frequenza con cui i clienti tornano alla tua app su base giornaliera e settimanale.
- Entrate: monitora le tendenze delle entrate in-app per identificare le aree di miglioramento della monetizzazione.
- Eventi personalizzati: monitora le azioni utente definite su misura e specifiche per la tua app.

Per saperne di più sui report di Mobile Analytics e sull'utilizzo della console Mobile Analytics, consulta la Panoramica dei report della Mobile Analytics Console nella Mobile Analytics Developer Guide.

Configurazione del progetto

Prerequisiti

Per utilizzare Mobile Analytics nella tua applicazione, devi aggiungere l'SDK al tuo progetto. A tale scopo, segui le istruzioni in Configurazione dell'SDK AWS Mobile per.NET and Xamarin.

Configurazione delle impostazioni di Mobile Analytics

Mobile Analytics definisce alcune impostazioni che possono essere configurate nel file awsconfig.xml:

```
var config = new MobileAnalyticsManagerConfig();
config.AllowUseDataNetwork = true;
config.DBWarningThreshold = 0.9f;
config.MaxDBSize = 5242880;
config.MaxRequestSize = 102400;
config.SessionTimeout = 5;
```

- SessionTimeout- Se l'app rimane in background per un periodo di tempo superiore a quello del SessionTimeout client Mobile Analytics del momento termina la sessione corrente e viene creata una nuova sessione quando l'app torna in primo piano. Consigliamo di utilizzare valori compresi tra 5 e 10. Il valore predefinito è 5.
- Max DBSize: la dimensione massima del database (in byte) utilizzata per l'archiviazione locale degli eventi. Se la dimensione del database supera questo valore, gli eventi aggiuntivi verranno ignorati. Si consiglia di utilizzare valori compresi tra 1 MB e 10 MB. Il valore predefinito è 5242880 (5 MB).

- DBWarningThreashold: la soglia di avviso. I valori validi sono compresi tra 0 e 1. Se i valori superano la soglia, verranno generati dei log di avviso. Il valore predefinito è 0,9.
- MaxRequestSize- La dimensione massima di una richiesta HTTP effettuata al servizio Mobile Analytics. Il valore è specificato in byte e può variare tra 1 e 512 KB. Il valore predefinito è 102400 (100 KB). Non utilizzare valori superiori a 512 KB, in quanto ciò potrebbe causare il rifiuto della richiesta HTTP da parte del servizio.
- AllowUseDataNetwork- Un valore che indica se la chiamata di servizio è consentita su una rete dati cellulare. Utilizzate questa opzione con cautela poiché ciò potrebbe aumentare l'utilizzo dei dati da parte del cliente.

Le impostazioni mostrate sopra sono i valori predefiniti per ogni elemento di configurazione.

Integrazione della Mobile Analytics con la tua applicazione

Le sezioni seguenti spiegano come integrare Mobile Analytics con la tua app.

Creare un'app nella Mobile Analytics Console

Vai alla console Amazon Mobile Analytics e crea un'app. Annota il `appId` valore, poiché ti servirà in seguito. Quando crei un'app nella Mobile Analytics Console, devi specificare l'ID del tuo pool di identità. Per istruzioni sulla creazione di un pool di identità, consulta Configurazione dell'SDK AWS Mobile per.NET and Xamarin.

Per ulteriori informazioni sull'utilizzo della Mobile Analytics Console, consulta la Panoramica dei report della Mobile Analytics Console nella Mobile Analytics Developer Guide.

Crea un MobileAnalyticsManager cliente

Per inizializzare MobileAnalyticsManager, richiama GetOrCreateInstance le tue credenziali AWS`MobileAnalyticsManager`, la tua regione, l'ID dell'applicazione Mobile Analytics e il tuo oggetto di configurazione opzionale:

```
// Initialize the MobileAnalyticsManager
analyticsManager = MobileAnalyticsManager.GetOrCreateInstance(
  cognitoCredentials,
  RegionEndpoint.USEast1,
  APP_ID,
  config
```

```
);
```

`APP_ID`Viene generato automaticamente durante la procedura guidata di creazione dell'app. Entrambi questi valori devono corrispondere a quelli della Mobile Analytics Console. `APP_ID`Viene utilizzato per raggruppare i dati nella console Mobile Analytics. Per trovare l'ID dell'app dopo averla creata nella console Mobile Analytics, accedi alla Mobile Analytics Console e fai clic sull'icona a forma di ingranaggio nell'angolo in alto a destra dello schermo. Verrà visualizzata la pagina di gestione delle app che elenca tutte le app registrate e le relative app IDs.

Registra gli eventi di monetizzazione

L'SDK AWS Mobile per .NET and `MonetizationEvent` Xamarin fornisce la classe che consente di generare eventi di monetizzazione per tenere traccia degli acquisti effettuati all'interno di applicazioni mobili. Il seguente frammento di codice mostra come creare un evento di monetizzazione:

```
// Create the monetization event object
MonetizationEvent monetizationEvent = new MonetizationEvent();

// Set the details of the monetization event
monetizationEvent.Quantity = 3.0;
monetizationEvent.ItemPrice = 1.99;
monetizationEvent.ProductId = "ProductId123";
monetizationEvent.ItemPriceFormatted = "$1.99";
monetizationEvent.Store = "Your-App-Store";
monetizationEvent.TransactionId = "TransactionId123";
monetizationEvent.Currency = "USD";

// Record the monetiziation event
analyticsManager.RecordEvent(monetizationEvent);
```

Registra eventi personalizzati

Mobile Analytics consente di definire eventi personalizzati. Gli eventi personalizzati sono definiti interamente da te; ti aiutano a tenere traccia delle azioni degli utenti specifiche per la tua app o il tuo gioco. Per ulteriori informazioni sugli eventi personalizzati, consulta Custom-Events.

Per questo esempio, diremo che la nostra app è un gioco e che vogliamo registrare un evento quando un utente completa un livello. Crea un evento «LevelComplete» creando una nuova `AmazonMobileAnalyticsEvent` istanza:

```
CustomEvent customEvent = new CustomEvent("LevelComplete");
```

```
// Add attributes
customEvent.AddAttribute("LevelName","Level1");
customEvent.AddAttribute("CharacterClass","Warrior");
customEvent.AddAttribute("Successful","True");

// Add metrics
customEvent.AddMetric("Score",12345);
customEvent.AddMetric("TimeInLevel",64);

// Record the event
analyticsManager.RecordEvent(customEvent);
```

Sessioni di registrazione

Xamarin iOS

Quando l'applicazione perde il focus, puoi mettere in pausa la sessione.
Per le app iOS, nel AppDelegate file.cs, sostituisci `DidEnterBackground` `MobileAnalyticsManager.PauseSesion` e `WillEnterForeground` chiama, `MobileAnalyticsManager.ResumeSession` come mostrato nel seguente frammento:

```
public override void DidEnterBackground(UIApplication application)
{
  // ...
  _manager.PauseSesson();
  // ...
}

public override void WillEnterForeground(UIApplication application)
{
  // ...
  _manager.ResumeSession();
  // ...
}
```

Xamarin Android

Per le app Android, chiamate `MobileAnalyticsManager.PauseSesion` il metodo OnPause () e `MobileAnalyticsManager.ResumeSession` il metodo OnResume (), come illustrato nel seguente frammento di codice:

```
protected override void OnResume()
{
  _manager.ResumeSession();
  base.OnResume();
}

protected override void OnPause()
{
  _manager.PauseSession();
  base.OnPause();
}
```

Per impostazione predefinita, se l'utente distoglie l'attenzione dall'app per meno di 5 secondi e torna all'app, la sessione verrà ripresa. Se l'utente distoglie l'attenzione dall'app per 5 secondi o più, verrà creata una nuova sessione. Questa impostazione è configurabile nel file di configurazione aws_mobile_analytics.json impostando la proprietà «SESSION_DELTA» sul numero di secondi di attesa prima di creare una nuova sessione.

Servizio Amazon Simple Storage (S3)

Che cos'è S3?

Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3), offre agli sviluppatori uno storage di oggetti sicuro, durevole e altamente scalabile. Amazon S3 è facile da usare, con una semplice interfaccia di servizi Web per archiviare e recuperare qualsiasi quantità di dati da qualsiasi punto del Web. Con Amazon S3, paghi solo per lo spazio di archiviazione effettivamente utilizzato. Non è prevista una tariffa minima e non viene applicato alcun costo di configurazione.

Amazon S3 offre uno storage di oggetti conveniente per un'ampia varietà di casi d'uso, tra cui applicazioni cloud, distribuzione di contenuti, backup e archiviazione, disaster recovery e analisi dei big data.

Per informazioni sulla disponibilità della regione AWS S3, consulta AWS Service Region Availability.

Concetti chiave

Bucket

Ogni oggetto archiviato in Amazon S3 risiede in un bucket. I bucket consentono di raggruppare oggetti correlati nello stesso modo in cui si utilizza una directory per raggruppare i file in un file system. I bucket hanno proprietà, come le autorizzazioni di accesso e lo stato del controllo delle versioni, e puoi specificare la regione in cui desideri che risiedano.

Per ulteriori informazioni sui bucket S3, consulta Working with Bucket nella S3 Developer Guide.

Oggetti

Gli oggetti sono i dati archiviati in Amazon S3. Ogni oggetto risiede in un bucket creato in una regione AWS specifica.

Gli oggetti archiviati in una regione non la lasciano mai a meno che non vengano trasferiti esplicitamente in un'altra regione. Ad esempio, gli oggetti archiviati nella regione UE (Irlanda) non la escono mai. Gli oggetti archiviati in una regione Amazon S3 rimangono fisicamente in quella regione. Amazon S3 non conserva copie né le sposta in altre regioni. Sarà tuttavia possibile accedere agli oggetti da qualsiasi posizione, purché si disponga delle autorizzazioni necessarie.

Gli oggetti possono essere di qualsiasi tipo di file: immagini, dati di backup, filmati, ecc. La dimensione massima per un oggetto è di 5 TB. Un bucket può avere un numero illimitato di oggetti.

Prima di poter caricare un oggetto in Amazon S3, è necessario disporre delle autorizzazioni di scrittura in un bucket. Per ulteriori informazioni sull'impostazione delle autorizzazioni per i bucket, consulta Modifica delle autorizzazioni per i bucket nella S3 Developer Guide.

Per saperne di più sugli oggetti S3, consulta Working with Objects nella S3 Developer Guide.

Metadata degli oggetti

Ogni oggetto in Amazon S3 ha un set di coppie chiave-valore che ne rappresentano i metadati. Esistono due tipi di metadata:

- Metadati di sistema: a volte elaborati da Amazon S3, ad esempio Content-Type e Content-Length.
- Metadati utente: mai elaborati da Amazon S3. I metadata degli utenti vengono archiviati e restituiti con l'oggetto. La dimensione massima dei metadata degli utenti è di 2 KB. Sia le chiavi, sia i relativi valori devono inoltre essere conformi agli standard US-ASCII.

Per ulteriori informazioni sui metadati degli oggetti S3, consulta Modifica dei metadati degli oggetti.

Configurazione del progetto

Prerequisiti

Per utilizzare Amazon S3 nella tua applicazione, devi aggiungere l'SDK al tuo progetto. A tale scopo, segui le istruzioni in Configurazione dell'SDK AWS Mobile per.NET and Xamarin.

Creare un bucket S3

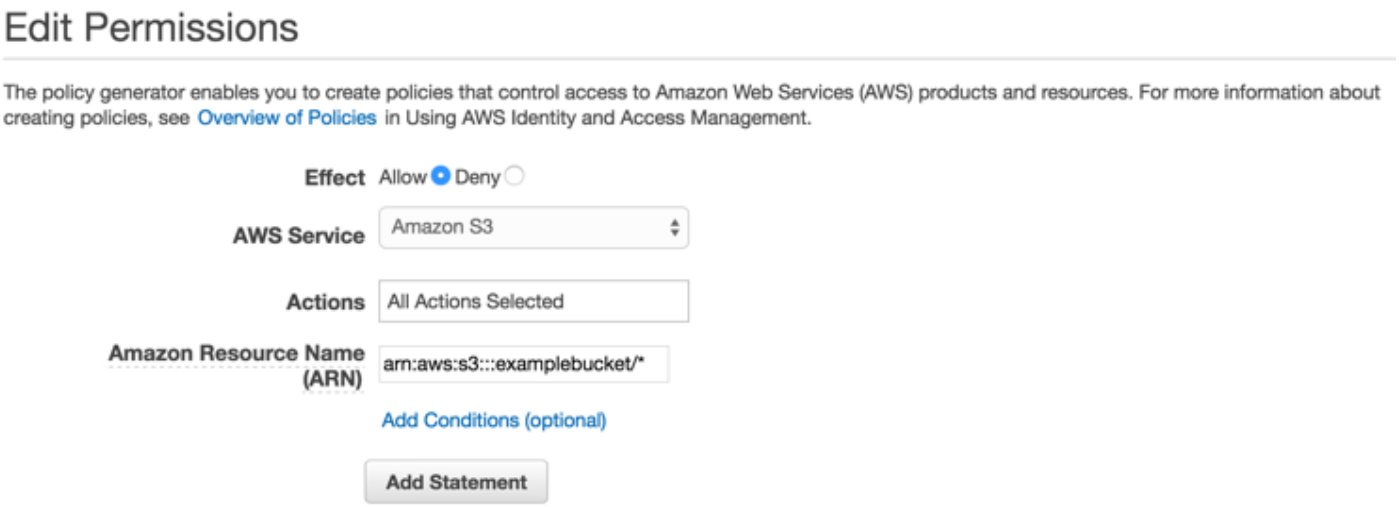
Amazon S3 archivia le risorse dell'applicazione in bucket Amazon S3, contenitori di archiviazione cloud che risiedono in una regione specifica. Ogni bucket Amazon S3 deve avere un nome univoco globale. Puoi usare la console Amazon S3 per creare un bucket.

1. Accedi alla console Amazon S3 e fai clic su Create Bucket.
2. Inserisci il nome del bucket, seleziona una regione e fai clic su Crea.

Imposta le autorizzazioni per S3

La policy di ruolo IAM predefinita consente all'applicazione di accedere ad Amazon Mobile Analytics e Amazon Cognito Sync. Affinché il pool di identità di Cognito possa accedere ad Amazon S3, è necessario modificare i ruoli del pool di identità.

1. Vai alla Identity and Access Management Console e fai clic su Ruoli nel riquadro a sinistra.
2. Digitare il nome del pool di identità nella casella di ricerca. Vengono elencati due ruoli, relativi, rispettivamente, agli utenti autenticati e non.
3. Fai clic sul ruolo per gli utenti non autenticati (verrà aggiunto unauth al nome del tuo pool di identità).
4. Fai clic su Crea politica sul ruolo, seleziona Policy Generator, quindi fai clic su Seleziona.
5. Nella pagina Modifica autorizzazioni, inserisci le impostazioni mostrate nell'immagine seguente, sostituendo l'Amazon Resource Name (ARN) con il tuo. L'ARN di un bucket S3 ha l'aspetto `arn:aws:s3:::examplebucket/*` ed è composto dalla regione in cui si trova il bucket e dal nome del bucket. Le impostazioni mostrate di seguito consentiranno al pool di identità di accedere a tutte le azioni per il bucket specificato.



1. Fai clic sul pulsante Aggiungi dichiarazione, quindi su Passaggio successivo.
2. La procedura guidata ti mostrerà la configurazione che hai generato. Fai clic su Applica politica.

Per ulteriori informazioni sulla concessione dell'accesso a S3, consulta Garantire l'accesso a un bucket Amazon S3.

(opzionale) Configura la versione di firma per le richieste S3

Ogni interazione con Amazon S3 è autenticata o anonima. AWS utilizza gli algoritmi Signature Version 4 o Signature Version 2 per autenticare le chiamate al servizio.

Tutte le nuove regioni AWS create dopo gennaio 2014 supportano solo la versione Signature 4. Tuttavia, molte regioni precedenti supportano ancora le richieste Signature Version 4 e Signature Version 2.

Se il tuo bucket si trova in una delle regioni che non supportano le richieste Signature Version 2 elencate in questa pagina, devi impostare AWSConfigs S3. UseSignatureVersion4 proprietà su «true» in questo modo:

```
AWSConfigsS3.UseSignatureVersion4 = true;
```

Per ulteriori informazioni sulle versioni di AWS Signature, consulta Authenticating Requests (AWS Signature versione 4).

Integrazione di S3 con la tua applicazione

Esistono due modi per interagire con S3 nell'applicazione Xamarin. I due metodi vengono esplorati in modo approfondito nei seguenti argomenti:

Utilizzo della S3 Transfer Utility

La S3 Transfer Utility semplifica il caricamento e il download di file su S3 dall'applicazione Xamarin.

Inizializza il TransferUtility

Crea un client S3, passandogli l'oggetto delle credenziali AWS, quindi passa il client S3 all'utilità di trasferimento, in questo modo:

```
var s3Client = new AmazonS3Client(credentials,region);
var transferUtility = new TransferUtility(s3Client);
```

(opzionale) Configura il TransferUtility

È possibile configurare tre proprietà opzionali:

- ConcurrentServiceRequests- Determina quanti thread attivi o il numero di richieste Web asincrone simultanee verranno utilizzati nel file. upload/download Il valore predefinito è 10.
- MinSizeBeforePartUpload- Ottiene o imposta la dimensione minima della parte per il caricamento delle parti in byte. Il valore predefinito è 16 MB. Se si riduce la dimensione minima delle parti, i caricamenti composti da più parti vengono suddivisi in un numero maggiore di parti più piccole. L'impostazione di questo valore su un valore troppo basso ha un effetto negativo sulla velocità di trasferimento, causando latenza e comunicazione di rete aggiuntive per ogni parte.
- NumberOfUploadThreads- Ottiene o imposta il numero di thread in esecuzione. Questa proprietà determina quanti thread attivi verranno utilizzati per caricare il file. Il valore predefinito è 10 thread.

Per configurare il TransferUtility client S3, crea un oggetto di configurazione, imposta le proprietà e passa l'oggetto al TransferUtility costruttore in questo modo:

```
var config = new TransferUtilityConfig();

config.ConcurrentServiceRequests = 10;
config.MinSizeBeforePartUpload=16*1024*1024;
config.NumberOfUploadThreads=10;

var s3Client = new AmazonS3Client(credentials);
var utility = new TransferUtility(s3Client,config);
```

Download di un file

Per scaricare un file da S3, `Download` richiamate l'oggetto Transfer Utility, passando i seguenti parametri:

- `file`- Nome stringa del file che vuoi scaricare
- `bucketName`- Nome stringa del bucket S3 da cui si desidera scaricare il file
- `key`- Una stringa che rappresenta il nome dell'oggetto S3 (un file in questo caso) da scaricare

```
transferUtility.Download(
  Path.Combine(Environment.SpecialFolder.ApplicationData,"file"),
  "bucketName",
  "key"
);
```

Caricamento di un file

Per caricare un file `Upload` su S3, richiamate l'oggetto Transfer Utility, passando i seguenti parametri:

- `file`- Nome stringa del file che vuoi caricare
- `bucketName`- Nome stringa del bucket S3 per archiviare il file

```
transferUtility.Upload(
  Path.Combine(Environment.SpecialFolder.ApplicationData,"file"),
  "bucketName"
);
```

Il codice precedente presuppone che ci sia un file nella directory Environment. SpecialFolder. ApplicationData. I caricamenti utilizzano automaticamente la funzionalità di caricamento in più parti di S3 su file di grandi dimensioni per migliorare la velocità di trasmissione.

Utilizzo del livello di servizio S3 APIs

Oltre a utilizzare S3 TransferUtility, puoi anche interagire con S3 utilizzando l'S3 di basso livello. APIs

Inizializza il client Amazon S3

Per utilizzare Amazon S3, dobbiamo prima creare un'istanza AmazonS3Client che faccia riferimento all'istanza Cognito AWSCredentials che hai creato in precedenza e alla tua regione:

```
AmazonS3Client S3Client = new AmazonS3Client (credentials,region);
```

Download di un file

Per scaricare un file da S3:

```
// Create a GetObject request
GetObjectRequest request = new GetObjectRequest
{
    BucketName = "SampleBucket",
    Key = "Item1"
};
```

```
// Issue request and remember to dispose of the response
using (GetObjectResponse response = client.GetObject(request))
{
    using (StreamReader reader = new StreamReader(response.ResponseStream))
    {
        string contents = reader.ReadToEnd();
        Console.WriteLine("Object - " + response.Key);
        Console.WriteLine(" Version Id - " + response.VersionId);
        Console.WriteLine(" Contents - " + contents);
    }
}
```

Caricamento di un file

Per caricare un file su S3:

```
// Create a client
AmazonS3Client client = new AmazonS3Client();

// Create a PutObject request
PutObjectRequest request = new PutObjectRequest
{
    BucketName = "SampleBucket",
    Key = "Item1",
    FilePath = "contents.txt"
};

// Put object
PutObjectResponse response = client.PutObject(request);
```

Eliminazione di una voce

Per eliminare un elemento in S3:

```
// Create a client
AmazonS3Client client = new AmazonS3Client();

// Create a DeleteObject request
DeleteObjectRequest request = new DeleteObjectRequest
{
    BucketName = "SampleBucket",
```

```
    Key = "Item1"
};

// Issue request
client.DeleteObject(request);
```

Eliminare più elementi

Per eliminare più oggetti da un bucket utilizzando una singola richiesta HTTP:

```
// Create a client
AmazonS3Client client = new AmazonS3Client();

// Create a DeleteObject request
DeleteObjectsRequest request = new DeleteObjectsRequest
{
    BucketName = "SampleBucket",
    Objects = new List<KeyVersion>
    {
        new KeyVersion() {Key = "Item1"},
        // Versioned item
        new KeyVersion() { Key = "Item2", VersionId =
 "Rej8CiBxcZKVK81cLr39j27Y5FVXghDK", },
        // Item in subdirectory
        new KeyVersion() { Key = "Logs/error.txt"}
    }
};

try
{
    // Issue request
    DeleteObjectsResponse response = client.DeleteObjects(request);
}
catch (DeleteObjectsException doe)
{
    // Catch error and list error details
    DeleteObjectsResponse errorResponse = doe.Response;

    foreach (DeletedObject deletedObject in errorResponse.DeletedObjects)
    {
        Console.WriteLine("Deleted item " + deletedObject.Key);
    }
    foreach (DeleteError deleteError in errorResponse.DeleteErrors)
```

```
    {
        Console.WriteLine("Error deleting item " + deleteError.Key);
        Console.WriteLine(" Code - " + deleteError.Code);
        Console.WriteLine(" Message - " + deleteError.Message);
    }
}
```

È possibile specificare fino a 1000 chiavi.

Creazione di un elenco di bucket

Per restituire un elenco di tutti i bucket di proprietà del mittente autenticato della richiesta:

```
// Create a client
AmazonS3Client client = new AmazonS3Client();

// Issue call
ListBucketsResponse response = client.ListBuckets();

// View response data
Console.WriteLine("Buckets owner - {0}", response.Owner.DisplayName);
foreach (S3Bucket bucket in response.Buckets)
{
    Console.WriteLine("Bucket {0}, Created on {1}", bucket.BucketName,
 bucket.CreationDate);
}
```

Elenco di oggetti

Puoi restituire alcuni o tutti (fino a 1000) gli oggetti archiviati nel tuo bucket S3. A tale scopo, è necessario disporre dell'accesso in lettura al bucket.

```
// Create a GetObject request
GetObjectRequest request = new GetObjectRequest
{
  BucketName = "SampleBucket",
  Key = "Item1"
};

// Issue request and remember to dispose of the response
using (GetObjectResponse response = client.GetObject(request))
{
```

```
  using (StreamReader reader = new StreamReader(response.ResponseStream))
  {
      string contents = reader.ReadToEnd();
      Console.WriteLine("Object - " + response.Key);
      Console.WriteLine(" Version Id - " + response.VersionId);
      Console.WriteLine(" Contents - " + contents);
  }
}
```

Ottieni una regione di Bucket

Per ottenere la regione in cui risiede un bucket:

```
// Create a client
AmazonS3Client client = new AmazonS3Client();

// Construct request
GetBucketLocationRequest request = new GetBucketLocationRequest
{
    BucketName = "SampleBucket"
};

// Issue call
GetBucketLocationResponse response = client.GetBucketLocation(request);

// View response data
Console.WriteLine("Bucket location - {0}", response.Location);
```

Ottieni una politica di Bucket

Per ottenere una politica di Bucket:

```
// Create a client
AmazonS3Client client = new AmazonS3Client();

// Construct request
GetBucketPolicyRequest getRequest = new GetBucketPolicyRequest
{
   BucketName = "SampleBucket"
};
string policy = client.GetBucketPolicy(getRequest).Policy;
```

```
Console.WriteLine(policy);
Debug.Assert(policy.Contains("BasicPerms"));
```

Amazon DynamoDB

Che cos'è Amazon DynamoDB?

Amazon DynamoDB è un servizio di database non relazionale veloce e altamente scalabile. DynamoDB rimuove le tradizionali limitazioni di scalabilità sullo storage dei dati mantenendo una bassa latenza e prestazioni prevedibili.

Concetti chiave

I concetti del modello di dati DynamoDB includono tabelle, elementi e attributi.

Tabelle

In Amazon DynamoDB, un database è una raccolta di tabelle. Una tabella è una raccolta di elementi e ogni elemento è un insieme di attributi.

In un database relazionale, una tabella ha uno schema predefinito come il nome della tabella, la chiave primaria, l'elenco dei nomi delle colonne e i relativi tipi di dati. Tutti i record memorizzati nella tabella devono avere lo stesso set di colonne. Al contrario, DynamoDB richiede solo che una tabella abbia una chiave primaria, ma non richiede di definire in anticipo tutti i nomi degli attributi e i tipi di dati.

Per ulteriori informazioni sull'utilizzo delle tabelle, consulta Lavorare con le tabelle in DynamoDB.

Elementi e attributi

I singoli elementi in una tabella DynamoDB possono avere un numero qualsiasi di attributi, sebbene esista un limite di 400 KB per la dimensione dell'elemento. La dimensione di un elemento è la somma delle lunghezze dei nomi e dei valori degli attributi (lunghezze binarie e UTF-8).

Ogni attributo di un elemento è una coppia nome-valore. Un attributo può essere un insieme a valore singolo o multivalore. Ad esempio, un elemento del libro può avere gli attributi del titolo e dell'autore. Ogni libro ha un titolo ma può avere molti autori. L'attributo multivalore è un set; non sono consentiti valori duplicati.

Ad esempio, prendete in considerazione l'archiviazione di un catalogo di prodotti in DynamoDB. È possibile creare una tabella con ProductCatalog l'attributo Id come chiave primaria. La chiave

primaria identifica in modo univoco ogni articolo, in modo che due prodotti nella tabella non possano avere lo stesso ID.

Per ulteriori informazioni sull'utilizzo degli elementi, consulta Lavorare con gli elementi in DynamoDB.

Tipi di dati

Amazon DynamoDB supporta i seguenti tipi di dati:

- Tipi scalari: Number, String, Binary, Boolean e Null.
- Tipi multivalore: set di stringhe, set di numeri e set binario.
- Tipi di documenti: elenco e mappa.

Per ulteriori informazioni sui tipi di dati scalari, i tipi di dati multivalore e i tipi di dati del documento, vedere Tipi di dati DynamoDB.

Chiave primaria

Quando crei una tabella, oltre al nome della tabella, è necessario specificare la chiave primaria della tabella. La chiave primaria identifica in modo univoco ogni elemento della tabella, in modo che due elementi non possano avere la stessa chiave. DynamoDB supporta i seguenti due tipi di chiavi primarie:

- Chiave hash: La chiave primaria è composta da un attributo, un attributo hash. DynamoDB crea un indice hash non ordinato su questo attributo chiave primario. Ogni elemento della tabella è identificato in modo univoco dal relativo valore di chiave hash.
- Hash e Range Key: la chiave primaria è composta da due attributi. Il primo attributo è l'attributo hash e il secondo è l'attributo range. DynamoDB crea un indice hash non ordinato sull'attributo della chiave primaria hash e un indice di intervallo ordinato sull'attributo della chiave primaria range. Ogni elemento della tabella è identificato in modo univoco dalla combinazione dei valori delle chiavi hash e range. È possibile che due elementi abbiano lo stesso valore di chiave hash, ma tali due elementi devono avere valori di chiave di intervallo diversi.

Indici secondari

Quando crei una tabella con un hash e una chiave di intervallo, puoi facoltativamente definire uno o più indici secondari su quella tabella. Un indice secondario consente di eseguire query sui dati nella tabella utilizzando una chiave alternativa, oltre alle query sulla chiave primaria.

DynamoDB supporta due tipi di indici secondari: indici secondari locali e indici secondari globali.

- Indice secondario locale: un indice che ha la stessa chiave hash della tabella, ma una chiave di intervallo diversa.
- Indice secondario globale: un indice con un hash e una chiave di intervallo che può essere diverso da quelli della tabella.

È possibile definire fino a 5 indici secondari globali e 5 indici secondari locali per tabella. Per ulteriori informazioni, consulta Improving Data Access with Secondary Indexes in DynamoDB nella DynamoDB Developer Guide.

Interrogazione e scansione

Oltre a utilizzare le chiavi primarie per accedere agli elementi, Amazon DynamoDB ne fornisce anche APIs due per la ricerca dei dati: Query e Scan. Ti consigliamo di leggere le Guidelines for Query and Scan nella DynamoDB Developer Guide per acquisire familiarità con alcune best practice.

Query

Un'operazione di Query trova gli elementi in una tabella o in un indice secondario utilizzando solo i valori degli attributi chiave primari. È necessario fornire un nome di attributo chiave hash e un valore distinto da cercare. Facoltativamente, è possibile fornire un nome e un valore dell'attributo chiave di intervallo e utilizzare un operatore di confronto per affinare i risultati della ricerca.

Per domande di esempio, consulta:

- Utilizzo del modello di documento
- Utilizzo del modello di persistenza degli oggetti
- Utilizzo del livello di servizio DynamoDB APIs

Per ulteriori informazioni su Query, consulta Query nella DynamoDB Developer Guide.

Scan

Un'operazione di scansione legge ogni elemento di una tabella o di un indice secondario. Per impostazione predefinita, un'operazione di scansione restituisce tutti gli attributi dei dati per ogni elemento della tabella o dell'indice. È possibile utilizzare il ProjectionExpression parametro in modo che Scan restituisca solo alcuni attributi, anziché tutti.

Per esempi di scansioni, vedi:

- Utilizzo del modello di documento
- Utilizzo del modello di persistenza degli oggetti
- Utilizzo del livello di servizio DynamoDB APIs

Per ulteriori informazioni su Scan, consulta Scan nella DynamoDB Developer Guide.

Configurazione del progetto

Prerequisiti

Per utilizzare DynamoDB nella tua applicazione, devi aggiungere l'SDK al tuo progetto. A tale scopo, segui le istruzioni in Configurazione dell'SDK AWS Mobile per.NET and Xamarin.

Creare una tabella DynamoDB

Per creare una tabella, accedi alla console DynamoDB e segui questi passaggi:

1. Fare clic su Create Table (Crea tabella).
2. Inserisci il nome della tabella.
3. Seleziona Hash come tipo di chiave principale.
4. Seleziona un tipo e inserisci un valore per il nome dell'attributo hash. Fai clic su Continue (Continua).
5. Nella pagina Aggiungi indici, se prevedi di utilizzare indici secondari globali, imposta il tipo di indice su «Indice secondario globale» e in Index Hash Key, inserisci un valore per l'indice secondario. Ciò ti consentirà di eseguire query e scansioni utilizzando sia l'indice primario che l'indice secondario. Fai clic su Aggiungi indice alla tabella, quindi su Continua. Per non utilizzare gli indici secondari globali, fai clic su Continua.
6. Imposta la capacità di lettura e scrittura ai livelli desiderati. Per ulteriori informazioni sulla configurazione della capacità, consulta Provisioned Throughput in Amazon DynamoDB. Fai clic su Continue (Continua).
7. Nella schermata successiva, inserisci un'e-mail di notifica per creare allarmi di throughput, se lo desideri. Fai clic su Continue (Continua).
8. Nella pagina di riepilogo, fai clic su Crea. DynamoDB creerà il tuo database.

Impostazione delle autorizzazioni per DynamoDB

Per utilizzare DynamoDB in un'applicazione, è necessario impostare le autorizzazioni corrette. La seguente policy IAM consente all'utente di eliminare, ottenere, inserire, interrogare, scansionare e aggiornare gli elementi in una tabella DynamoDB specifica, identificata da ARN:

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DeleteItem",
        "dynamodb:GetItem",
        "dynamodb:PutItem",
        "dynamodb:Query",
        "dynamodb:Scan",
        "dynamodb:UpdateItem"
      ],
      "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/MyTable"
    }
  ]
}
```

È possibile modificare le policy nella console IAM. Dovresti aggiungere o rimuovere le azioni consentite in base alle esigenze della tua app.

Per ulteriori informazioni sulle policy IAM, consulta la sezione relativa all'utilizzo di IAM.

Per ulteriori informazioni sulle policy specifiche di DynamoDB, consulta Using IAM to Control Access to DynamoDB Resources nella DynamoDB Developer Guide.

Integrazione di DynamoDB con la tua applicazione

L'SDK AWS Mobile per .NET e Xamarin fornisce una libreria di alto livello per lavorare con DynamoDB. Puoi anche effettuare richieste direttamente all'API DynamoDB di basso livello, ma per la maggior parte dei casi d'uso è consigliata la libreria di alto livello. AmazonDynamoDBClient È una parte particolarmente utile della libreria di alto livello. Utilizzando questa classe, è possibile eseguire varie operazioni di creazione, lettura, aggiornamento ed eliminazione (CRUD) ed eseguire interrogazioni.

L'SDK AWS Mobile per .NET e Xamarin ti consente di APIs effettuare chiamate utilizzando l'SDK AWS for .NET per lavorare con DynamoDB. Tutti sono disponibili nel file dll. APIs AWSSDK Per informazioni sul download dell'SDK AWS per.NET, consulta SDK AWS per .NET.

Esistono tre modi per interagire con DynamoDB nell'applicazione Xamarin:

- Modello di documento: questa API fornisce classi wrapper attorno all'API DyanModB di basso livello per semplificare ulteriormente le attività di programmazione. Table e Document sono le classi wrapper chiave. È possibile utilizzare il modello di documento per le operazioni sui dati come creare, recuperare, aggiornare ed eliminare elementi. L'API è disponibile in Amazon.DynamoDB. DocumentModel namespace.
- Modello di persistenza degli oggetti: l'API Object Persistence consente di mappare le classi lato client alle tabelle DynamoDB. Ogni istanza dell'oggetto viene quindi mappata a un elemento nelle tabelle corrispondenti. La DBContext classe Dynamo di questa API fornisce metodi per salvare oggetti lato client in una tabella, recuperare elementi come oggetti ed eseguire query e scansioni. È possibile utilizzare il modello Object Persistence per le operazioni sui dati come la creazione, il recupero, l'aggiornamento e l'eliminazione di elementi. È necessario innanzitutto creare le tabelle utilizzando l'API Service Client e quindi utilizzare il modello di persistenza degli oggetti per mappare le classi alle tabelle. L'API è disponibile in Amazon.DynamoDB. DataModel namespace.
- API Service Client: questa è l'API a livello di protocollo che si collega strettamente all'API DynamoDB. È possibile utilizzare questa API di basso livello per tutte le operazioni relative a tabelle ed elementi, come la creazione, l'aggiornamento e l'eliminazione di tabelle ed elementi. Puoi anche interrogare e scansionare le tue tabelle. Questa API è disponibile nello spazio dei nomi Amazon.DynamoDB.

Questi tre modelli vengono approfonditi nei seguenti argomenti:

Utilizzo del modello di documento

Il Document Model fornisce classi wrapper basate sull'API.NET di basso livello. Table e Document sono le classi wrapper chiave. È possibile utilizzare il modello di documento per creare, recuperare, aggiornare ed eliminare elementi. Per creare, aggiornare ed eliminare tabelle, è necessario utilizzare l'API di basso livello. Per istruzioni su come utilizzare l'API di basso livello, consulta Utilizzo del livello di servizio DynamoDB. APIs L'API di basso livello è disponibile in Amazon.DynamoDB. DocumentModel namespace.

Per ulteriori informazioni sul Document Model, vedi.NET Document Model.

Creare un client DynamoDB

Per creare un client DynamoDB:

```
var client = new AmazonDynamoDBClient(credentials,region);
DynamoDBContext context = new DynamoDBContext(client);
```

Operazioni CRUD

Salvataggio di una voce

Crea un elemento:

```
Table table = Table.LoadTable(client, "Books");
id = Guid.NewGuid().ToString();
var books = new Document();
books["Id"] = id;
books["Author"] = "Mark Twain";
books["Title"] = "Adventures of Huckleberry Finn";
books["ISBN"] = "112-111111";
books["Price"] = "10";
```

Salva un elemento in una tabella DynamoDB:

```
var book = await table.PutItemAsync(books);
```

Recupero di una voce

Per recuperare un elemento:

```
public async Task GetItemAsync(AWSCredentials credentials, RegionEndpoint region)
{
      var client = new AmazonDynamoDBClient(credentials, region);
      Table books = Table.LoadTable(client, "Books");
      var book = await books.GetItemAsync(id);
}
```

Aggiornamento di una voce

Per aggiornare un articolo:

```
public async Task UpdateItemAttributesAsync(AWSCredentials credentials, RegionEndpoint
 region)
{
      var book = new Document();
      book["Id"] = id;
      book["PageCount"] = "200";
      var client = new AmazonDynamoDBClient(credentials, region);
      Table books = Table.LoadTable(client, "Books");
      Document updatedBook = await books.UpdateItemAsync(book);
}
```

Per aggiornare un articolo in modo condizionale:

```
public async Task UpdateItemConditionallyAsync(AWSCredentials credentials,
 RegionEndpoint region) {
      var book = new Document();
      book["Id"] = id;
      book["Price"] = "30";

      // For conditional price update, creating a condition expression.
      Expression expr = new Expression();
      expr.ExpressionStatement = "Price = :val";
      expr.ExpressionAttributeValues[":val"] = 10.00;

      var client = new AmazonDynamoDBClient(credentials, region);
      Table books = Table.LoadTable(client, "Books");

      Document updatedBook = await books.UpdateItemAsync(book);
}
```

Eliminazione di una voce

Per eliminare un elemento:

```
public async Task DeleteItemAsync(AWSCredentials credentials, RegionEndpoint region)
{
  var client = new AmazonDynamoDBClient(credentials, region);
  Table books = Table.LoadTable(client, "Books");
  await books.DeleteItemAsync(id);
}
```

Interrogazione e scansione

Per interrogare e recuperare tutti i libri il cui autore è «Mark Twain»:

```
public async Task QueryAsync(AWSCredentials credentials, RegionEndpoint region) {
  var client = new AmazonDynamoDBClient(credentials, region);
  Table books = Table.LoadTable(client, "Books");
  var search = books.Query(new QueryOperationConfig() {
   IndexName = "Author-Title-index",
   Filter = new QueryFilter("Author", QueryOperator.Equal, "Mark Twain")
  });
  Console.WriteLine("ScanAsync: printing query response");
  var documents = await search.GetRemainingAsync();
  documents.ForEach((d) = > {
   PrintDocument(d);
  });
}
```

Il codice di esempio di scansione riportato di seguito restituisce tutti i libri della nostra tabella:

```
public async Task ScanAsync(AWSCredentials credentials, RegionEndpoint region) {
      var client = new AmazonDynamoDBClient(credentials, region);
      Table books = Table.LoadTable(client, "Books");
      var search = books.Scan(new ScanOperationConfig() {
              ConsistentRead = true
      });
      Console.WriteLine("ScanAsync: printing scan response");
      var documents = await search.GetRemainingAsync();
      documents.ForEach((d) = > {
              PrintDocument(d);
      });
}
```

Utilizzo del modello di persistenza degli oggetti

L'SDK AWS Mobile per .NET and Xamarin fornisce un modello di Object Persistence che consente di mappare le classi lato client su una tabella DynamoDB. Ogni istanza di oggetto viene quindi mappata su un elemento nella tabella corrispondente. Per salvare gli oggetti sul lato client in una tabella, il modello Object Persistence fornisce la DBContext classe Dynamo, un punto di ingresso a DynamoDB. Questa classe fornisce una connessione a DynamoDB e consente di accedere alle tabelle, eseguire varie operazioni CRUD ed eseguire query.

Il modello Object Persistence non fornisce un'API per creare, aggiornare o eliminare tabelle. Esso fornisce solo operazioni di dati. Per creare, aggiornare ed eliminare tabelle, è necessario utilizzare l'API di basso livello. Per istruzioni su come utilizzare l'API di basso livello, consulta Utilizzo del livello di servizio DynamoDB. APIs

Panoramica di

Il modello Object Persistence fornisce un set di attributi per mappare le classi lato client alle tabelle e agli attributi delle tabelle. properties/fields Il modello Object Persistence supporta sia la mappatura esplicita che quella predefinita tra le proprietà delle classi e gli attributi delle tabelle.

- Mappatura esplicita: per mappare una proprietà a una chiave primaria, è necessario utilizzare gli attributi del modello Dynamo Key e Dynamo DBHash Key Object Persistence. DBRange Inoltre, per gli attributi della chiave non primaria, se il nome di una proprietà nella classe e l'attributo di tabella corrispondente a cui si desidera mapparla non sono gli stessi, è necessario definire la mappatura aggiungendo esplicitamente l'attributo Dynamo. DBProperty
- Mappatura predefinita: per impostazione predefinita, il modello Object Persistence mappa le proprietà della classe agli attributi con lo stesso nome nella tabella.

Non è necessario mappare ogni singola proprietà della classe. È possibile identificare queste proprietà aggiungendo l'attributo DynamoDBIgnore . Il salvataggio e il recupero di un'istanza di un oggetto comporterebbero l'omissione di qualsiasi proprietà contrassegnata con questo attributo.

Tipi di dati supportati

Il modello Object Persistence supporta un insieme di tipi di dati.NET primitivi, raccolte e tipi di dati arbitrari. Il modello supporta i seguenti tipi di dati primitivi:

- bool
- byte
- char
- DateTime
- decimale, doppio, float
- Int16, Int32, Int64
- SByte
- stringa

- UInt16, UInt32, UInt64

Il modello Object Persistence supporta anche i tipi di raccolte.NET con le seguenti limitazioni:

- Il tipo di raccolta deve implementare l' ICollection interfaccia.
- Il tipo di raccolta deve essere composto dai tipi primitivi supportati. <bool>Ad esempio ICollection<string>, ICollection.
- Il tipo di raccolta deve fornire un costruttore senza parametri.

Per ulteriori informazioni sul modello Object Persistence, vedi.NET Object Persistence Model.

Creare un client DynamoDB

Per creare un client DynamoDB:

```
var client = new AmazonDynamoDBClient(credentials,region);
DynamoDBContext context = new DynamoDBContext(client);
```

Operazioni CRUD

Salvare un oggetto

Crea un oggetto:

```
[DynamoDBTable("Books")]
public class Book {
  [DynamoDBHashKey] // Hash key.
  public string Id {
    get;
    set;
  }

  [DynamoDBGlobalSecondaryIndexHashKey]
  public string Author {
    get;
    set;
  }

  [DynamoDBGlobalSecondaryIndexRangeKey]
  public string Title {
```

```
    get;
    set;
  }
  public string ISBN {
    get;
    set;
  }
  public int Price {
    get;
    set;
  }
  public string PageCount {
    get;
    set;
  }
}

Book myBook = new Book
{
    Id = id,
    Author = "Charles Dickens",
    Title = "Oliver Twist",
    ISBN = "111-1111111001",
    Price = 10,
    PageCount = 300
};
```

Salva un oggetto in una tabella DynamoDB:

```
context.Save(myBook);
```

Recupera un oggetto

Per recuperare un oggetto:

```
Book retrievedBook = context.Load<Book>(1);
```

Aggiornare un oggetto

Per aggiornare un oggetto:

```
Book retrievedBook = context.Load<Book>(1);
```

```
retrievedBook.ISBN = "111-1111111001";
context.Save(retrievedBook);
```

Eliminazione di un oggetto

Per eliminare un oggetto:

```
Book retrievedBook = context.Load<Book>(1);
context.Delete(retrievedBook);
```

Interrogazione e scansione

Per interrogare e recuperare tutti i libri il cui autore è «Charles Dickens»:

```
public async Task QueryAsync(AWSCredentials credentials, RegionEndpoint region) {
  var client = new AmazonDynamoDBClient(credentials, region);
  DynamoDBContext context = new DynamoDBContext(client);

  var search = context.FromQueryAsync < Book > (new
 Amazon.DynamoDBv2.DocumentModel.QueryOperationConfig() {
    IndexName = "Author-Title-index",
    Filter = new Amazon.DynamoDBv2.DocumentModel.QueryFilter("Author",
 Amazon.DynamoDBv2.DocumentModel.QueryOperator.Equal, "Charles Dickens")
  });

  Console.WriteLine("items retrieved");

  var searchResponse = await search.GetRemainingAsync();
  searchResponse.ForEach((s) = > {
    Console.WriteLine(s.ToString());
  });
}
```

Il codice di esempio di scansione riportato di seguito restituisce tutti i libri della nostra tabella:

```
public async Task ScanAsync(AWSCredentials credentials, RegionEndpoint region) {
  var client = new AmazonDynamoDBClient(credentials, region);
  DynamoDBContext context = new DynamoDBContext(client);

  var search = context.FromScanAsync < Book > (new
 Amazon.DynamoDBv2.DocumentModel.ScanOperationConfig() {
```

```
    ConsistentRead = true
  });

  Console.WriteLine("items retrieved");

  var searchResponse = await search.GetRemainingAsync();
  searchResponse.ForEach((s) = > {
   Console.WriteLine(s.ToString());
  });
}
```

Utilizzo del livello di servizio DynamoDB APIs

Il livello di servizio Dynamo APIs consente di creare, aggiornare ed eliminare tabelle. È inoltre possibile eseguire operazioni tipiche di creazione, lettura, aggiornamento ed eliminazione (CRUD) sugli elementi di una tabella utilizzando questa API.

Creare un client DynamoDB

Per creare un client DynamoDB:

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient(credentials,region);
```

Operazioni CRUD

Salvataggio di una voce

Per salvare un elemento in una tabella DynamoDB:

```
// Create a client
AmazonDynamoDBClient client = new AmazonDynamoDBClient(credentials,region);

// Define item attributes
Dictionary<string, AttributeValue> attributes = new Dictionary<string,
 AttributeValue>();

// Author is hash-key
attributes["Author"] = new AttributeValue { S = "Mark Twain" };
attributes["Title"] = new AttributeValue { S = "The Adventures of Tom Sawyer" };
attributes["PageCount"] = new AttributeValue { N = "275" };
```

```
attributes["Price"] = new AttributeValue{N = "10.00"};
attributes["Id"] = new AttributeValue{N="10"};
attributes["ISBN"] = new AttributeValue{S="111-1111111"};

// Create PutItem request
PutItemRequest request = new PutItemRequest
{
    TableName = "Books",
    Item = attributes
};

// Issue PutItem request
var response = await client.PutItemAsync(request);
```

Recupero di una voce

Per recuperare un elemento:

```
// Create a client
AmazonDynamoDBClient client = new AmazonDynamoDBClient(credentials,region);

Dictionary<string, AttributeValue> key = new Dictionary<string, AttributeValue>
{
    { "Id", new AttributeValue { N = "10" } }
};

// Create GetItem request
GetItemRequest request = new GetItemRequest
{
    TableName = "Books",
    Key = key,
};

// Issue request
var result = await client.GetItemAsync(request);

// View response
Console.WriteLine("Item:");
Dictionary<string, AttributeValue> item = result.Item;
foreach (var keyValuePair in item)
{
    Console.WriteLine("Author := {0}", item["Author"]);
    Console.WriteLine("Title := {0}", item["Title"]);
```

```
    Console.WriteLine("Price:= {0}", item["Price"]);
    Console.WriteLine("PageCount := {0}", item["PageCount"]);
}
```

Aggiornamento di una voce

Per aggiornare un elemento:

```
// Create a client
AmazonDynamoDBClient client = new AmazonDynamoDBClient(credentials,region);

Dictionary<string, AttributeValue> key = new Dictionary<string, AttributeValue>
{
    { "Id", new AttributeValue { N = "10" } }
};

// Define attribute updates
Dictionary<string, AttributeValueUpdate> updates = new Dictionary<string,
 AttributeValueUpdate>();
// Add a new string to the item's Genres SS attribute
updates["Genres"] = new AttributeValueUpdate()
{
    Action = AttributeAction.ADD,
    Value = new AttributeValue { SS = new List<string> { "Bildungsroman" } }
};

// Create UpdateItem request
UpdateItemRequest request = new UpdateItemRequest
{
    TableName = "Books",
    Key = key,
    AttributeUpdates = updates
};

// Issue request
var response = await client.UpdateItemAsync(request);
```

Eliminazione di una voce

Per eliminare un elemento:

```
// Create a client
```

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient(credentials,region);

Dictionary<string, AttributeValue> key = new Dictionary<string, AttributeValue>
{
  { "Id", new AttributeValue { N = "10" } }
};

// Create DeleteItem request
DeleteItemRequest request = new DeleteItemRequest
{
  TableName = "Books",
  Key = key
};

// Issue request
var response = await client.DeleteItemAsync(request);
```

Interrogazione e scansione

Per interrogare e recuperare tutti i libri il cui autore è «Mark Twain»:

```
public void Query(AWSCredentials credentials, RegionEndpoint region) {
  using(var client = new AmazonDynamoDBClient(credentials, region)) {
    var queryResponse = await client.QueryAsync(new QueryRequest() {
      TableName = "Books",
      IndexName = "Author-Title-index",
      KeyConditionExpression = "Author = :v_Id",
      ExpressionAttributeValues = new Dictionary < string, AttributeValue > {
        {
          ":v_Id", new AttributeValue {
            S = "Mark Twain"
          }
        }
      }
    });
    queryResponse.Items.ForEach((i) = > {
      Console.WriteLine(i["Title"].S);
    });

  }
}
```

Il codice di esempio di scansione riportato di seguito restituisce tutti i libri della nostra tabella:

```
public void Scan(AWSCredentials credentials, RegionEndpoint region) {
      using(var client = new AmazonDynamoDBClient(credentials, region)) {
              var queryResponse = client.Scan(new ScanRequest() {
                      TableName = "Books"
              });
              queryResponse.Items.ForEach((i) = > {
                      Console.WriteLine(i["Title"].S);
              });
      }
}
```

Amazon Simple Notification Service (SNS)

Utilizzando SNS e l'SDK AWS Mobile per .NET e Xamarin, puoi scrivere applicazioni in grado di ricevere notifiche push mobili. Per informazioni su SNS, consulta Amazon Simple Notification Service.

Concetti chiave

Amazon SNS consente alle applicazioni e agli utenti finali su diversi dispositivi di ricevere notifiche tramite le code di notifica Mobile Push (dispositivi Apple, Google e Kindle Fire), HTTP/HTTPS, Email/Email -JSON, SMS o le code di Amazon Simple Queue Service (SQS) o le funzioni AWS Lambda. SNS ti consente di inviare messaggi singoli o messaggi estesi a un gran numero di destinatari abbonati a un singolo argomento.

Argomenti

Un argomento è un «punto di accesso» che consente ai destinatari di sottoscrivere dinamicamente copie identiche della stessa notifica. Un argomento può supportare le consegne a più tipi di endpoint: ad esempio, puoi raggruppare i destinatari iOS, Android e SMS.

Sottoscrizioni

Per ricevere i messaggi pubblicati in un argomento devi effettuare la sottoscrizione di un endpoint all'argomento specificato. Un endpoint è un'app mobile, un server Web, un indirizzo e-mail o una coda Amazon SQS in grado di ricevere messaggi di notifica da Amazon SNS. Una volta effettuata la sottoscrizione di un endpoint a un argomento, e dopo che la sottoscrizione è stata confermata, l'endpoint riceverà tutti i messaggi pubblicati nell'argomento specificato.

Pubblicazione

Quando pubblichi su un argomento, SNS fornisce copie del messaggio in formato appropriato a ciascun sottoscrittore di quell'argomento. Per le notifiche push mobili, puoi pubblicare direttamente sull'endpoint o sottoscrivere l'endpoint a un argomento.

Configurazione del progetto

Prerequisiti

Per utilizzare SNS nella tua applicazione, devi aggiungere l'SDK al tuo progetto. A tale scopo, segui le istruzioni in Configurazione dell'SDK AWS Mobile per.NET and Xamarin.

Imposta le autorizzazioni per SNS

Per informazioni sull'impostazione delle autorizzazioni per SNS, consulta gli argomenti sulla gestione dell'accesso ai tuoi argomenti su Amazon SNS.

Aggiungi un NuGet pacchetto per SNS al tuo progetto

Segui il passaggio 4 delle istruzioni in Configurazione dell'SDK AWS Mobile per.NET and Xamarin per aggiungere il pacchetto NuGet Amazon Simple Notification Service al tuo progetto.

Integrazione di SNS con la tua applicazione

Esistono molti modi per interagire con SNS nell'applicazione Xamarin:

Invio di notifiche push (Xamarin Android)

Questo documento spiega come inviare notifiche push a un'applicazione Xamarin Android utilizzando Amazon Simple Notification Service (SNS) e l'SDK AWS Mobile per .NET and Xamarin.

Configurazione del progetto

Prerequisiti

È necessario completare tutte le istruzioni sulla configurazione dell'SDK AWS Mobile per.NET and Xamarin prima di iniziare questo tutorial.

Imposta le autorizzazioni per SNS

Segui il passaggio 2 in Configurazione dell'SDK AWS Mobile per.NET and Xamarin per allegare la policy menzionata di seguito ai ruoli dell'applicazione. Questo darà alla tua applicazione le autorizzazioni appropriate per accedere a SNS:

1. Vai alla console IAM e seleziona il ruolo IAM che desideri configurare.
2. Fai clic su Allega politica, seleziona la politica di Amazon SNSFull Access e fai clic su Allega politica.

> **Warning**
>
> L'uso di Amazon SNSFull Access non è consigliato in un ambiente di produzione. Lo utilizziamo qui per consentirti di essere subito operativo. Per ulteriori informazioni sulla specificazione delle autorizzazioni per un ruolo IAM, consulta Panoramica delle autorizzazioni dei ruoli IAM.

Abilita le notifiche push su Google Cloud

Innanzitutto, aggiungi un nuovo progetto API di Google:

1. Vai alla Google Developers Console.
2. Fai clic su Crea progetto.
3. Nella casella Nuovo progetto, inserisci il nome del progetto, prendi nota dell'ID del progetto (ti servirà in seguito) e fai clic su Crea.

Quindi, abilita il servizio Google Cloud Messaging (GCM) per il tuo progetto:

1. Nella Google Developers Console, il tuo nuovo progetto dovrebbe essere già selezionato. In caso contrario, selezionalo nel menu a discesa nella parte superiore della pagina.
2. Seleziona APIs & auth dalla barra laterale sul lato sinistro della pagina.
3. Nella casella di ricerca, digita «Google Cloud Messaging per Android» e fai clic sul link Google Cloud Messaging per Android.
4. Fai clic su Abilita API.

Infine, ottieni una chiave API:

1. Nella Google Developers Console, seleziona APIs & auth > Credentials.
2. In Accesso pubblico all'API, fai clic su Crea nuova chiave.
3. Nella finestra di dialogo Crea una nuova chiave, fai clic su Chiave server.

4. Nella finestra di dialogo visualizzata, fai clic su Crea e copia la chiave API visualizzata. Utilizzerai questa chiave API per eseguire l'autenticazione in un secondo momento.

Usa l'ID del progetto per creare un ARN di piattaforma nella console SNS

1. Vai alla console SNS.
2. Fai clic su Applicazioni sul lato sinistro dello schermo.
3. Fai clic su Crea applicazione di piattaforma per creare una nuova applicazione di piattaforma SNS.
4. Inserisci un nome per l'applicazione.
5. Seleziona Google Cloud Messaging (GCM) per la piattaforma di notifica push.
6. Incolla la chiave API nella casella di testo denominata Chiave API.
7. Fai clic su Crea applicazione della piattaforma.
8. Seleziona l'applicazione della piattaforma che hai appena creato e copia l'ARN dell'applicazione.

Aggiungi un NuGet pacchetto per SNS al tuo progetto

Segui il passaggio 4 delle istruzioni in Configurazione dell'SDK AWS Mobile per.NET and Xamarin per aggiungere il pacchetto NuGet Amazon Simple Notification Service al tuo progetto.

Crea un client SNS

```
var snsClient = new AmazonSimpleNotificationServiceClient(credentials, region);
```

Registra la tua applicazione per le notifiche remote

Per registrarti alle notifiche remote su Android, dovrai crearne uno BroadcastReceiver che possa ricevere messaggi di Google Cloud. Cambia il nome del pacchetto qui sotto dove richiesto:

```
[BroadcastReceiver(Permission = "com.google.android.c2dm.permission.SEND")]
[IntentFilter(new string[] {
      "com.google.android.c2dm.intent.RECEIVE"
}, Categories = new string[] {
      "com.amazonaws.sns" /* change to match your package */
})]
[IntentFilter(new string[] {
      "com.google.android.c2dm.intent.REGISTRATION"
}, Categories = new string[] {
```

```
      "com.amazonaws.sns" /* change to match your package */
})]
[IntentFilter(new string[] {
      "com.google.android.gcm.intent.RETRY"
}, Categories = new string[] {
      "com.amazonaws.sns" /* change to match your package */
})]
public class GCMBroadcastReceiver: BroadcastReceiver {
      const string TAG = "PushHandlerBroadcastReceiver";
      public override void OnReceive(Context context, Intent intent) {
              GCMIntentService.RunIntentInService(context, intent);
              SetResult(Result.Ok, null, null);
      }
}

[BroadcastReceiver]
[IntentFilter(new[] {
      Android.Content.Intent.ActionBootCompleted
})]
public class GCMBootReceiver: BroadcastReceiver {
      public override void OnReceive(Context context, Intent intent) {
              GCMIntentService.RunIntentInService(context, intent);
              SetResult(Result.Ok, null, null);
      }
}
```

Di seguito è riportato il servizio che riceve la notifica push da BroadcastReceiver e la visualizza sulla barra delle notifiche del dispositivo:

```
[Service]
 public class GCMIntentService: IntentService {
  static PowerManager.WakeLock sWakeLock;
  static object LOCK = new object();

  public static void RunIntentInService(Context context, Intent intent) {
    lock(LOCK) {
      if (sWakeLock == null) {
        // This is called from BroadcastReceiver, there is no init.
        var pm = PowerManager.FromContext(context);
        sWakeLock = pm.NewWakeLock(
        WakeLockFlags.Partial, "My WakeLock Tag");
      }
    }
```

```
    sWakeLock.Acquire();
    intent.SetClass(context, typeof(GCMIntentService));
    context.StartService(intent);
  }

  protected override void OnHandleIntent(Intent intent) {
    try {
      Context context = this.ApplicationContext;
      string action = intent.Action;

      if (action.Equals("com.google.android.c2dm.intent.REGISTRATION")) {
        HandleRegistration(intent);
      } else if (action.Equals("com.google.android.c2dm.intent.RECEIVE")) {
        HandleMessage(intent);
      }
    } finally {
      lock(LOCK) {
        //Sanity check for null as this is a public method
        if (sWakeLock != null) sWakeLock.Release();
      }
    }
  }

  private void HandleRegistration(Intent intent) {
    string registrationId = intent.GetStringExtra("registration_id");
    string error = intent.GetStringExtra("error");
    string unregistration = intent.GetStringExtra("unregistered");

    if (string.IsNullOrEmpty(error)) {
      var response = await SnsClient.CreatePlatformEndpointAsync(new
CreatePlatformEndpointRequest {
        Token = registrationId,
        PlatformApplicationArn = "YourPlatformArn" /* insert your platform application
ARN here */
      });
    }
  }

  private void HandleMessage(Intent intent) {
    string message = string.Empty;
    Bundle extras = intent.Extras;
    if (!string.IsNullOrEmpty(extras.GetString("message"))) {
      message = extras.GetString("message");
```

```
    } else {
      message = extras.GetString("default");
    }

    Log.Info("Messages", "message received = " + message);
    ShowNotification(this, "SNS Push", message);
    //show the message

  }

  public void ShowNotification(string contentTitle,
  string contentText) {
    // Intent
    Notification.Builder builder = new Notification.Builder(this)
      .SetContentTitle(contentTitle)
      .SetContentText(contentText)
      .SetDefaults(NotificationDefaults.Sound | NotificationDefaults.Vibrate)
      .SetSmallIcon(Resource.Drawable.Icon)
      .SetSound(RingtoneManager.GetDefaultUri(RingtoneType.Notification));

    // Get the notification manager:
    NotificationManager notificationManager =
 this.GetSystemService(Context.NotificationService) as NotificationManager;

    notificationManager.Notify(1001, builder.Build());
  }
}
```

Invia un messaggio dalla console SNS al tuo endpoint

1. Vai alla console SNS > Applicazioni.
2. Seleziona l'applicazione della piattaforma, seleziona un endpoint e fai clic su Pubblica sull'endpoint.
3. Digita un messaggio di testo nella casella di testo e fai clic su Pubblica messaggio per pubblicare un messaggio.

Invio di notifiche push (Xamarin iOS)

Questo documento spiega come inviare notifiche push a un'applicazione Xamarin iOS utilizzando Amazon Simple Notification Service (SNS) e l'SDK AWS Mobile per .NET and Xamarin.

Configurazione del progetto

Prerequisiti

È necessario completare tutte le istruzioni sulla configurazione dell'SDK AWS Mobile per.NET and Xamarin prima di iniziare questo tutorial.

Imposta le autorizzazioni per SNS

Segui il passaggio 2 in Configurazione dell'SDK AWS Mobile per.NET and Xamarin per allegare la policy menzionata di seguito ai ruoli dell'applicazione. Questo darà alla tua applicazione le autorizzazioni appropriate per accedere a SNS:

1. Vai alla console IAM e seleziona il ruolo IAM che desideri configurare.
2. Fai clic su Allega politica, seleziona la politica di Amazon SNSFull Access e fai clic su Allega politica.

> **Warning**
>
> L'uso di Amazon SNSFull Access non è consigliato in un ambiente di produzione. Lo utilizziamo qui per consentirti di essere subito operativo. Per ulteriori informazioni sulla specificazione delle autorizzazioni per un ruolo IAM, consulta Panoramica delle autorizzazioni dei ruoli IAM.

Ottieni l'iscrizione all'Apple iOS Developer Program

Dovrai eseguire l'app su un dispositivo fisico per ricevere notifiche push. Per eseguire l'app su un dispositivo, devi essere iscritto all'Apple iOS Developer Program Membership. Una volta sottoscritta l'iscrizione, puoi usare Xcode per generare un'identità di firma. Per ulteriori informazioni, consulta la documentazione Quick Start di App Distribution di Apple.

Crea un certificato iOS

Innanzitutto, devi creare un certificato iOS. Quindi, è necessario creare un profilo di provisioning configurato per le notifiche push. A tale scopo:

1. Vai all'Apple Developer Member Center, fai clic su Certificati, identificatori e profili.

2. Fai clic su Identificatori in App iOS, fai clic sul pulsante più nell'angolo in alto a destra della pagina Web per aggiungere un nuovo ID app iOS e inserisci una descrizione dell'ID app.
3. Scorri verso il basso fino alla sezione Aggiungi suffisso ID, seleziona Explicit App ID e inserisci l'identificatore del pacchetto.
4. Scorri verso il basso fino alla sezione Servizi app e seleziona Notifiche push.
5. Fai clic su Continue (Continua).
6. Fare clic su Submit (Invia).
7. Fai clic su Fine.
8. Seleziona l'ID app che hai appena creato, quindi fai clic su Modifica.
9. Scorri verso il basso fino alla sezione Notifiche push. Fai clic su Crea certificato in Certificato SSL di sviluppo.
10. Segui le istruzioni per creare una richiesta di firma del certificato (CSR), carica la richiesta e scarica un certificato SSL che verrà utilizzato per comunicare con Apple Notification Service (APNS).
11. Torna alla pagina Certificati, identificatori e profili. Fai clic su Tutto in Provisioning Profiles.
12. Fai clic sul pulsante più nell'angolo in alto a destra per aggiungere un nuovo profilo di provisioning.
13. Seleziona Sviluppo app iOS, quindi fai clic su Continua.
14. Seleziona l'ID dell'app, quindi fai clic su Continua.
15. Seleziona il certificato per sviluppatori, quindi fai clic su Continua.
16. Seleziona il tuo dispositivo, quindi fai clic su Continua.
17. Inserisci un nome di profilo, quindi fai clic su Genera.
18. Scaricate e fate doppio clic sul file di provisioning per installare il profilo di provisioning.

Per ulteriori informazioni sul provisioning di un profilo configurato per le notifiche push, consulta la documentazione sulla configurazione delle notifiche push di Apple.

Usa il certificato per creare l'ARN della piattaforma nella console SNS

1. Esegui l'app di KeyChain accesso, seleziona I miei certificati nella parte inferiore sinistra dello schermo, quindi fai clic con il pulsante destro del mouse sul certificato SSL che hai generato per la connessione ad APNS e seleziona Esporta. Ti verrà richiesto di specificare un nome per il file e una password per proteggere il certificato. Il certificato verrà salvato in un file P12.

2. Vai alla console SNS e fai clic su Applicazioni sul lato sinistro dello schermo.
3. Fai clic su Crea applicazione di piattaforma per creare una nuova applicazione di piattaforma SNS.
4. Inserisci un nome per l'applicazione.
5. Seleziona Apple Development per la piattaforma di notifica push.
6. Fai clic su Scegli file e seleziona il file P12 che hai creato quando hai esportato il certificato SSL.
7. Inserisci la password che hai specificato quando hai esportato il certificato SSL e fai clic su Carica credenziali dal file.
8. Fai clic su Crea applicazione di piattaforma.
9. Seleziona l'applicazione della piattaforma che hai appena creato e copia l'ARN dell'applicazione. Ne avrai bisogno nei prossimi passaggi.

Aggiungi un NuGet pacchetto per SNS al tuo progetto

Segui il passaggio 4 delle istruzioni in Configurazione dell'SDK AWS Mobile per.NET and Xamarin per aggiungere il pacchetto NuGet Amazon Simple Notification Service al tuo progetto.

Crea un client SNS

```
var snsClient = new AmazonSimpleNotificationServiceClient(credentials, region);
```

Registra la tua applicazione per le notifiche remote

Per registrare un'applicazione, richiamate RegisterForRemoteNotifications il vostro UIApplication oggetto, come illustrato di seguito. Inserisci il codice seguente in AppDelegate .cs, inserendo l'ARN dell'applicazione della piattaforma dove richiesto di seguito:

```
public override bool FinishedLaunching(UIApplication app, NSDictionary options) {
// do something
var pushSettings = UIUserNotificationSettings.GetSettingsForTypes (
  UIUserNotificationType.Alert |
  UIUserNotificationType.Badge |
  UIUserNotificationType.Sound,
  null
);
app.RegisterUserNotifications(pushSettings);
app.RegisterForRemoteNotifications();
// do something
```

```
  return true;
}

public override void RegisteredForRemoteNotifications(UIApplication application, NSData
 token) {
  var deviceToken = token.Description.Replace("<", "").Replace(">", "").Replace(" ",
 "");
  if (!string.IsNullOrEmpty(deviceToken)) {
    //register with SNS to create an endpoint ARN
    var response = await SnsClient.CreatePlatformEndpointAsync(
    new CreatePlatformEndpointRequest {
      Token = deviceToken,
      PlatformApplicationArn = "YourPlatformArn" /* insert your platform application
 ARN here */
    });
  }
}
```

Invia un messaggio dalla console SNS al tuo endpoint

1. Vai alla console SNS > Applicazioni.
2. Seleziona l'applicazione della piattaforma, seleziona un endpoint e fai clic su Pubblica sull'endpoint.
3. Digita un messaggio di testo nella casella di testo e fai clic su Pubblica messaggio per pubblicare un messaggio.

Inviare e ricevere notifiche SMS

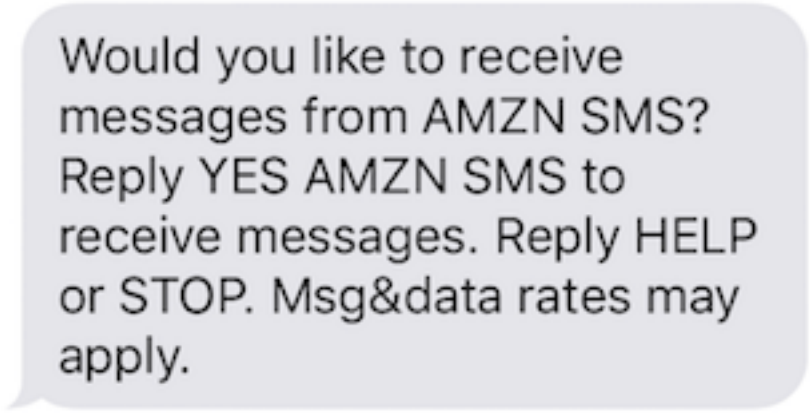
Puoi utilizzare Amazon Simple Notification Service (Amazon SNS) per inviare e ricevere notifiche Short Message Service (SMS) a telefoni cellulari e smartphone dotati di SMS.

> **Note**
>
> Le notifiche SMS sono attualmente supportate per i numeri di telefono negli Stati Uniti d'America. I messaggi SMS possono essere inviati solo da argomenti creati nella regione Stati Uniti orientali (Virginia settentrionale). Tuttavia, puoi pubblicare messaggi relativi ad argomenti creati nella regione Stati Uniti orientali (Virginia settentrionale) da qualsiasi altra regione.

Creazione di un argomento

Per creare un argomento:

1. Nella console Amazon SNS, fai clic su Crea nuovo argomento. Viene visualizzata la finestra di dialogo Crea nuovo argomento.
2. Nella casella Topic name (Nome argomento) inserisci un nome per l'argomento.
3. Nella casella Nome visualizzato, digitate un nome da visualizzare. All'argomento deve essere assegnato un nome visualizzato perché i primi dieci (10) caratteri del nome visualizzato vengono utilizzati come parte iniziale del prefisso del messaggio di testo. Il nome visualizzato immesso verrà visualizzato nel messaggio di conferma che SNS invia all'utente (il nome visualizzato di seguito è «AMZN SMS»).



1. Fai clic su Create topic (Crea argomento). Il nuovo argomento viene visualizzato nella pagina Topics (Argomenti).
2. Seleziona il nuovo argomento, quindi fai clic sul relativo ARN. Viene visualizzata la pagina Topic Details (Dettagli argomento).
3. Copia l'argomento ARN, poiché ti servirà quando ti iscrivi a un argomento nel passaggio successivo.

```
arn:aws:sns:us-west-2:111122223333:MyTopic
```

Iscriviti a un argomento utilizzando il protocollo SMS

Crea un client SNS, passando l'oggetto delle credenziali e la regione del tuo pool di identità:

```
var snsClient = new AmazonSimpleNotificationServiceClient(credentials, region);
```

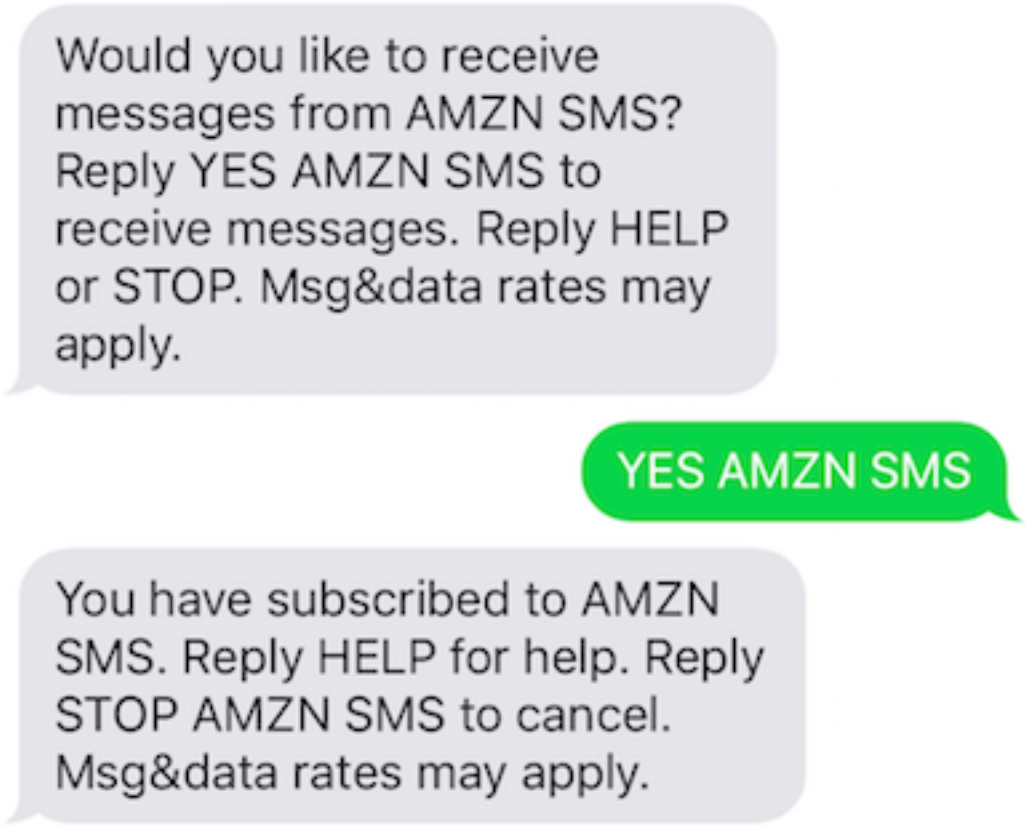
Per iscriverti a un argomento, richiama `SubscribeAsync` e passagli l'ARN dell'argomento a cui vuoi iscriverti, il protocollo («sms») e il numero di telefono:

```
var response = await snsClient.SubscribeAsync(topicArn, "sms", "1234567890");
```

Riceverai un arn di sottoscrizione nell'oggetto di risposta all'iscrizione. Il tuo arn di iscrizione ha il seguente aspetto:

```
arn:aws:sns:us-west-2:123456789012:MyTopic:6b0e71bd-7e97-4d97-80ce-4a0994e55286
```

Quando un dispositivo si iscrive a un argomento, SNS invierà un messaggio di conferma al dispositivo e l'utente dovrà confermare di voler ricevere notifiche, come mostrato di seguito:

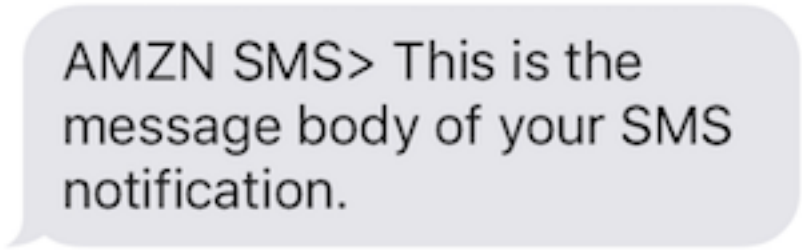


Dopo che l'utente si è iscritto all'argomento, riceverà messaggi SMS quando lo pubblicherai su quell'argomento.

Pubblica un messaggio

Per pubblicare un messaggio su un argomento:

1. Accedi alla Console di gestione AWS e apri la console Amazon SNS.
2. Nel riquadro di navigazione sinistro, fai clic su Topics (Argomenti) e seleziona l'argomento in cui effettuare la pubblicazione.
3. Fai clic su Pubblica sull'argomento.
4. Nella casella Oggetto, digita un oggetto.
5. Nella casella Messaggio, digitare un messaggio. Amazon SNS invia il testo che inserisci nella casella Messaggio agli abbonati SMS, a meno che tu non inserisca anche il testo nella casella Oggetto. Poiché Amazon SNS include un prefisso del nome visualizzato per tutti i messaggi SMS inviati, la somma del prefisso del nome visualizzato e del payload del messaggio non può superare 140 caratteri ASCII o 70 caratteri Unicode. Amazon SNS tronca i messaggi che superano questi limiti.
6. Fai clic su Publish Message (Pubblica messaggio). Amazon SNS visualizza una finestra di dialogo di conferma. Il messaggio SMS viene visualizzato sul tuo dispositivo abilitato agli SMS, come mostrato di seguito.



Inviare messaggi agli HTTP/HTTPS endpoint

Puoi utilizzare Amazon SNS per inviare messaggi di notifica a uno o più endpoint HTTP o HTTPS. Di seguito è riportato il procedimento:

1. Configura il tuo endpoint per ricevere messaggi Amazon SNS.
2. Sottoscrivi un HTTP/HTTPS endpoint a un argomento.
3. Conferma la tua iscrizione.
4. Pubblica una notifica sull'argomento. Amazon SNS invia quindi una richiesta HTTP POST che invia il contenuto della notifica all'endpoint sottoscritto.

Configura il tuo HTTP/HTTPS endpoint per ricevere messaggi Amazon SNS

Segui le istruzioni nella fase 1 di Invio di messaggi Amazon SNS agli HTTP/HTTPS endpoint per configurare il tuo endpoint.

Sottoscrivi il tuo HTTP/HTTPS endpoint al tuo argomento Amazon SNS

Crea un client SNS, passando l'oggetto delle credenziali e la regione del tuo pool di identità:

```
var snsClient = new AmazonSimpleNotificationServiceClient(credentials, region);
```

Per inviare messaggi a un endpoint HTTP o HTTPS tramite un argomento, devi effettuare la sottoscrizione dell'endpoint all'argomento Amazon SNS. Specificate l'endpoint utilizzando il relativo URL:

```
var response = await snsClient.SubscribeAsync(
  "topicArn",
  "http",  /* "http" or "https" */
  "endpointUrl" /* endpoint url beginning with http or https */
);
```

Conferma della sottoscrizione a

Dopo la sottoscrizione a un endpoint, Amazon SNS invierà un messaggio di conferma dell'iscrizione all'endpoint. Il codice dell'endpoint deve recuperare il `SubscribeURL` valore dal messaggio di conferma dell'abbonamento e visitare la posizione specificata dall'endpoint `SubscribeURL` stesso o renderlo disponibile all'utente in modo che sia possibile visitare manualmente il `SubscribeURL` (ad esempio, se si utilizza un browser Web).

Amazon SNS non invierà messaggi all'endpoint fino alla conferma dell'abbonamento. Quando visiterai `SubscribeURL`, otterrai una risposta con un documento XML contenente un elemento `SubscriptionArn` che specifica l'ARN della sottoscrizione.

Invia messaggi all'endpoint HTTP/HTTPS

È possibile inviare un messaggio agli abbonati di un argomento pubblicandolo sull'argomento. Invoca `PublishAsync` e passagli l'argomento ARN e il tuo messaggio.

```
var response = await snsClient.PublishAsync(topicArn, "This is your message");
```

Risoluzione dei problemi SNS

Utilizzo dello stato di consegna nella console Amazon SNS

La console Amazon SNS contiene una funzione Delivery Status che ti consente di raccogliere feedback sui tentativi di recapito riusciti e non riusciti dei tuoi messaggi alle piattaforme di notifica push mobili (Apple (APNS), Google (GCM), Amazon (ADM), Windows (WNS e MPNS) e Baidu.

Fornisce anche altre informazioni importanti come i tempi di permanenza in Amazon SNS. Queste informazioni vengono acquisite in un gruppo Amazon CloudWatch Log creato automaticamente da Amazon SNS quando questa funzionalità è abilitata tramite la console Amazon SNS o tramite Amazon SNS. APIs

Per istruzioni sull'uso della funzionalità Delivery Status, consulta Using the Delivery Status di Amazon SNS sul blog di AWS Mobile.

Best practice per l'utilizzo dell'SDK AWS Mobile per.NET and Xamarin

Ci sono solo alcuni principi fondamentali e best practice che è utile conoscere quando si utilizza l'SDK AWS Mobile per .NET and Xamarin.

- Usa Amazon Cognito per ottenere le credenziali AWS anziché codificare le tue credenziali nell'applicazione. Se codifichi le tue credenziali nella tua applicazione, potresti finire per esporle al pubblico, permettendo ad altri di effettuare chiamate verso AWS usando le tue credenziali. Per istruzioni su come utilizzare Amazon Cognito per ottenere le credenziali AWS, consulta Configurazione dell'SDK AWS Mobile per.NET and Xamarin.
- Per le best practice sull'uso di S3, consulta questo articolo sul blog di AWS.
- Per le best practice sull'utilizzo di DynamoDB, consulta DynamoDB Best Practices nella DynamoDB Developer Guide.

Cerchiamo sempre di aiutare i nostri clienti ad avere successo e accogliamo con favore i feedback, quindi non esitare a postare sui forum AWS o a segnalare un problema su GitHub.

Libreria di documentazione dei servizi AWS

Ogni servizio nell'SDK AWS Mobile per .NET and Xamarin ha una guida per sviluppatori separata e un riferimento all'API di servizio che fornisce informazioni aggiuntive che potresti trovare utili.

Amazon Cognito Identity

- Guida per sviluppatori di Cognito
- Riferimento all'API del servizio Cognito Identity

Amazon Cognito Sync

- Guida per sviluppatori di Cognito
- Riferimento all'API del servizio Cognito Sync

Amazon Mobile Analytics

- Guida per gli sviluppatori di Mobile Analytics
- Riferimento all'API del servizio Mobile Analytics

Simple Storage Service (Amazon S3)

- Guida per sviluppatori S3
- Guida introduttiva a S3
- Riferimento all'API del servizio S3

Amazon DynamoDB

- Guida per gli sviluppatori di DynamoDB
- Guida introduttiva a DynamoDB
- Riferimento all'API del servizio DynamoDB

Servizio Amazon Simply Notification (SNS)

- Guida per sviluppatori SNS
- Riferimento all'API del servizio SNS

Altri link utili

- Glossario dei termini di AWS
- Informazioni su AWS Credentials

Risoluzione dei problemi

Questo argomento descrive alcune idee per la risoluzione dei problemi che potresti incontrare quando usi l'SDK AWS Mobile for .NET and Xamarin.

Assicurati che IAM Role disponga delle autorizzazioni richieste

Quando si chiamano i servizi AWS, l'app deve utilizzare un'identità proveniente da un pool di identità di Cognito. Ogni identità nel pool è associata a un ruolo IAM (Identity and Access Management).

A un ruolo sono associati uno o più file di policy che specificano a quali risorse AWS hanno accesso gli utenti assegnati al ruolo. Per impostazione predefinita, vengono creati due ruoli per pool di identità: uno per gli utenti autenticati e uno per gli utenti non autenticati.

Dovrai modificare il file di policy esistente o associare un nuovo file di policy alle autorizzazioni richieste dall'app. Se la tua app consente sia utenti autenticati che non autenticati, a entrambi i ruoli devono essere concesse le autorizzazioni per accedere alle risorse AWS di cui l'app ha bisogno.

Il seguente file di policy mostra come concedere l'accesso a un bucket S3:

```
{
   "Statement":[
      {
         "Action":[
            "s3:AbortMultipartUpload",
            "s3:DeleteObject",
            "s3:GetObject",
            "s3:PutObject"
         ],
         "Effect":"Allow",
         "Resource":"arn:aws:s3:::MYBUCKETNAME/*",
         "Principal":"*"
      }
   ]
}
```

Il seguente file di policy mostra come concedere l'accesso a un database DynamoDB:

```
{
   "Statement":[
```

```
      {
         "Effect":"Allow",
         "Action":[
            "dynamodb:DeleteItem",
            "dynamodb:GetItem",
            "dynamodb:PutItem",
            "dynamodb:Scan",
            "dynamodb:UpdateItem"
         ],
         "Resource":"arn:aws:dynamodb:us-west-2:123456789012:table/MyTable"
      }
   ]
}
```

Per ulteriori informazioni sulla specificazione delle policy, consulta IAM Policies.

Utilizzo di un debugger proxy HTTP

Se il servizio AWS che la tua app sta chiamando ha un endpoint HTTP o HTTPS, puoi utilizzare un debugger HTTP/HTTPS proxy per visualizzare le richieste e le risposte e ottenere maggiori informazioni su ciò che sta accadendo. Sono disponibili diversi debugger proxy HTTP, come:

- Charles: un proxy di debug web per Windows e OSX
- Fiddler: un proxy di debug web per Windows

Sia Charles che Fiddler richiedono alcune configurazioni per poter visualizzare il traffico crittografato SSL, leggi la documentazione di questi strumenti per ulteriori informazioni. Se utilizzi un proxy di debug web che non può essere configurato per visualizzare traffico crittografato, apri il file aws_endpoints_json e imposta il tag HTTP per il servizio AWS necessario per il debug su true.

Cronologia dei documenti

La tabella seguente descrive le modifiche importanti alla documentazione dall'ultima versione dell'SDK AWS Mobile per .NET and Xamarin.

- Versione API: 27/08/2015
- Ultimo aggiornamento della documentazione: 23/02/2021

Modifica	Versione API	Descrizione	Data di rilascio
Archiviato	2015-08-27	L'SDK AWS mobile per Xamarin è incluso in. AWS SDK per .NET Questa guida fa riferimento alla versione archiviata di Mobile SDK per Xamarin.	23/02/2021
Versione GA	27/08/2015	Versione GA	27/08/2015
Versione beta	28/07/2015	Versione beta	rn2015-07-28