Red Hat Data Grid 8.4

Hot Rod .NET クライアントガイド

Hot Rod .NET/C# クライアントの設定および使用

Last Updated: 2024-04-30

Red Hat Data Grid 8.4 Hot Rod .NET クライアントガイド

Hot Rod .NET/C# クライアントの設定および使用

法律上の通知

概要

Hot Rod .NET/C# クライアントを使用すると、C# ランタイムアプリケーションがリモートの Data Grid クラスターに接続し、対話できます。

目次

RED HAT DATA GRID

Data Grid は、高性能の分散型インメモリーデータストアです。

スキーマレスデータ構造

さまざまなオブジェクトをキーと値のペアとして格納する柔軟性があります。

グリッドベースのデータストレージ

クラスター間でデータを分散および複製するように設計されています。

エラスティックスケーリング

サービスを中断することなく、ノードの数を動的に調整して要件を満たします。

データの相互運用性

さまざまなエンドポイントからグリッド内のデータを保存、取得、およびクエリーします。

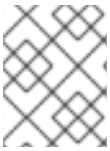
DATA GRID のドキュメント

Data Grid のドキュメントは、Red Hat カスタマーポータルで入手できます。

- Data Grid 8.4 ドキュメント
- Data Grid 8.4 コンポーネントの詳細
- Data Grid 8.4 でサポートされる設定
- Data Grid 8 機能のサポート
- Data Grid で非推奨の機能

DATA GRID のダウンロード

Red Hat カスタマーポータルで Data Grid Software Downloads にアクセスします。



注記

Data Grid ソフトウェアにアクセスしてダウンロードするには、Red Hat アカウントが必要です。

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ を参照してください。

第1章 HOT ROD .NET/C# クライアントのインストールおよび設定

.NET Framework を使用して **RemoteCache** API 経由で Data Grid クラスターと対話する Microsoft Windows システムに Hot Rod .NET/C# クライアントをインストールします。

1.1. HOT ROD .NET/C# クライアントのインストール

Data Grid は、Windows に Hot Rod .NET/C# クライアントをインストールするインストールパッケージを提供します。

前提条件

- Microsoft が .NET Framework をサポートするオペレーティングシステム
- .NET Framework 4.6.2 以降
- Windows Visual Studio 2015 以降

手順

1. Data Grid Software Downloads から **redhat-datagrid-<version>-hotrod-dotnet-client.msi** をダウンロードします。
2. Hot Rod .NET/C# クライアントの MSI インストーラーを起動し、インストールプロセスでインタラクティブなウィザードに従います。

1.2. 設定およびリモートキャッシュマネージャー API

ConfigurationBuilder API を使用して、Hot Rod .NET/C# クライアント接続と **RemoteCacheManager** API を設定してリモートキャッシュを取得および設定します。

基本設定

```
using Infinispan.HotRod;
using Infinispan.HotRod.Config;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace simpleapp
{
   class Program
   {
      static void Main(string[] args)
      {
         ConfigurationBuilder builder = new ConfigurationBuilder();
         // Connect to a server at localhost with the default port.
         builder.AddServer()
            .Host(args.Length > 1 ? args[0] : "127.0.0.1")
            .Port(args.Length > 2 ? int.Parse(args[1]) : 11222);
         Configuration config = builder.Build();
         // Create and start a RemoteCacheManager to interact with caches.
         RemoteCacheManager remoteManager = new RemoteCacheManager(conf);
```

```
            remoteManager.Start();
            IRemoteCache<string,string> cache=remoteManager.GetCache<string, string>();
            cache.Put("key", "value");
            Console.WriteLine("key = {0}", cache.Get("key"));
            remoteManager.Stop();
        }
    }
}
```

認証

```
ConfigurationBuilder builder = new ConfigurationBuilder();
// Add a server with specific connection timeouts
builder.AddServer().Host("127.0.0.1").Port(11222).ConnectionTimeout(90000).SocketTimeout(900);
// ConfigurationBuilder has fluent interface, options can be appended in chain.
// Enabling authentication with server name "node0",
// sasl mech "PLAIN", user "supervisor", password "aPassword", security realm "aRealm"
builder.Security().Authentication().Enable().ServerFQDN("node0")
    .SaslMechanism("PLAIN").SetupCallback("supervisor", "aPassword", "aRealm");
Configuration c = conf.Build();
```

暗号化

```
ConfigurationBuilder builder = new ConfigurationBuilder();
builder.AddServer().Host("127.0.0.1").Port(11222);
// Get configuration builder for encryption
SslConfigurationBuilder sslBuilder = conf.Ssl();
// Enable encryption and provide client certificate
sslBuilder.Enable().ClientCertificateFile("clientCertFilename");
// Provide server cert if server needs to be verified
sslBuilder.ServerCAFile("serverCertFilename");
Configuration c = conf.Build();
```

クロスサイトフェイルオーバー

```
ConfigurationBuilder builder = new ConfigurationBuilder();
  builder.AddServer().Host("127.0.0.1").Port(11222);
  // Configure a remote cluster and node when using cross-site failover.
  builder.AddCluster("nyc").AddClusterNode("192.0.2.0", 11322);
```

ニアキャッシュ

```
ConfigurationBuilder builder = new ConfigurationBuilder();
  builder.AddServer().Host("127.0.0.1").Port(11222);
  // Enable near-caching for the client.
  builder.NearCache().Mode(NearCacheMode.INVALIDATED).MaxEntries(10);
```