Red Hat Data Grid 8.4

Data Grid コードチュートリアル

Data Grid 機能の使用方法を学ぶ

Last Updated: 2024-06-05

Red Hat Data Grid 8.4 Data Grid コードチュートリアル

Data Grid 機能の使用方法を学ぶ

法律上の通知

概要

さまざまな Data Grid 機能および使用状況パターンを示すリモートキャッシュおよび埋め込みキャッシュのコードチュートリアルを実行します。

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。

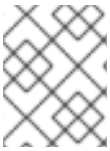
DATA GRID のドキュメント

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、用語の置き換えは、今後の複数のリリースにわたって段階的に実施されます。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

第1章 リモートキャッシュ

複数の Data Grid Server インスタンスをデプロイし、リモートキャッシュクラスターを作成します。これにより、Hot Rod および REST クライアントからの高速アクセスで、耐障害性があり、スケーラブルなデータ層が提供されます。

1.1. リモートキャッシュチュートリアル

これらのチュートリアルを実行するには、Data Grid Server のインスタンスを 1 つ以上ローカルで実行する必要があります。各チュートリアルで、**admin/password** 認証情報を使用して、**localhost:11222** で実行中のサーバーに接続します。ただし、Docker インスタンスが検出され、サーバーが実行されていない場合は、**Testcontainers** を使用してローカルサーバーを起動します。

ディストリビューションを ダウンロード し、以下のコマンドを実行します。

```
$ ./bin/cli.sh user create admin -p "password"
$ ./bin/server.sh
```

注記

Data Grid Server は、デフォルトで認証および承認を有効にします。**admin** という名前のユーザーを作成すると、Data Grid Server への管理アクセスが付与されます。

リモートキャッシュチュートリアルの構築および実行

次のように、IDE で直接、またはコマンドラインから、リモートキャッシュチュートリアルを構築して実行することができます。

```
$ mvn -s /path/to/maven-settings.xml clean package exec:exec
```

1.2. HOT ROD JAVA クライアントチュートリアル

- Data Grid には、少なくとも Java 11 が必要です。ただし、Java 8 を必要とするアプリケーションで実行している Hot Rod Java クライアントは、古いバージョンのクライアントライブラリーを引き続き使用できます。

チュートリアルのリンク	説明
リモートキャッシュの使用例	リモート分散キャッシュの動作を示す最も単純なコードサンプル。
キャッシュごとの設定	Data Grid Server に接続するときに動的にキャッシュを設定する方法を示します。
ニアキャッシュ	リモートキャッシュの読み取りパフォーマンスを改善するために、ニアキャッシュの設定方法を示します。
キャッシュ管 API	管理 API を使用して、キャッシュとキャッシュテンプレートを動的に作成する方法を示します。

チュートリアルのリンク	説明
エンコーディング	キャッシュのエンコーディングがどのように機能するかを示します。
クライアントリスナー	クライアントリスナーを使用して、リモートキャッシュ内のデータが変更されたことを検出します。
クエリー	リモートキャッシュ値をクエリーする方法を示します。
継続的なクエリー	継続的なクエリーおよびリモートキャッシュの使用方法を示します。
トランザクション	リモートトランザクションの仕組みを示します。
セキュアなキャッシュ	承認が有効になっているキャッシュの設定方法を示します。
TLS 認証	TLS 承認を使用して Data Grid Server に接続する方法を示します。
カウンター	リモートカウンターの仕組みを示します。
マルチマップ	リモートマルチマップの仕組みを示します。
タスクの実行	サーバータスクを登録する方法と、Hot Rod クライアントからの実行方法を示します。
JUnit 5 およびテストコンテナー	Data Grid および JUnit 5 拡張機能の使用方法を示します。
Persistence	Data Grid と永続キャッシュの使用方法を示します。
Redis クライアント	Data Grid と Redis クライアントを使用して、Resp プロトコルを使用して読み取りと書き込みを行う方法を示します。
リアクティブ API	Mutiny に基づいたリアクティブ API で Data Grid を使用する方法を示します。

Data Grid のドキュメント

Hot Rod Java クライアントのリソースの詳細については、以下のドキュメントを参照してください。

- Hot Rod Java クライアントガイド
- データのマーシャリングとエンコードに関するガイド
- Data Grid キャッシュのクエリー

- REST API
- 応答プロトコル
- Smallrye Mutiny

第2章 埋め込みキャッシュ

Java プロジェクトへの依存関係として Data Grid を追加し、アプリケーションのパフォーマンスを向上させ、複雑なユースケースを処理する機能を提供する埋め込みキャッシュを使用します。

2.1. 埋め込みキャッシュチュートリアル

以下のように、埋め込みキャッシュチュートリアルは、IDE で直接、またはコマンドラインから実行できます。

```
$ mvn -s /path/to/maven-settings.xml clean package exec:exec
```

チュートリアルのリンク	説明
分散キャッシュ	分散キャッシュの仕組みを示します。
レプリケートされたキャッシュ	レプリケートされたキャッシュの仕組みを示します。
無効化されたキャッシュ	無効化されたキャッシュの仕組みを示します。
トランザクション	トランザクションの仕組みを示します。
ストリーム	分散ストリームの仕組みを示します。
JCache の統合	JCache の仕組みを示します。
機能マップ	Functional Map API の仕組みを示します。
Map API	Map API が Data Grid キャッシュとどのように機能するかを示します。
マルチマップ	マルチマップの使用方法を示します。
クエリー	Data Grid Query を使用して、キャッシュ値でフルテキストクエリーを実行します。
クラスター化されたリスナー	クラスター化されたリスナーを使用して埋め込みキャッシュ内のデータが変更されたことを検出します。
カウンター	埋め込みのクラスター化カウンターの使用方法を示します。
クラスター化されたロック	埋め込みのクラスター化ロックの使用方法を示します。

チュートリアルのリンク	説明
クラスター化された実行	埋め込みのクラスター化カウンターの使用方法を示します。

Data Grid のドキュメント

埋め込みキャッシュのリソースの詳細については、以下のドキュメントを参照してください。

- Data Grid キャッシュの埋め込み
- Data Grid キャッシュのクエリー

2.2. KUBERNETES と OPENSHIFT のチュートリアル

このチュートリアルには、Kubernetes/OpenShift で Infinispan ライブラリーモードを (マイクロサービスとして) 実行する方法が記載されています。

前提条件: バックグラウンドで実行されている Maven および Docker デーモン。

前提条件

- 実行中の Openshift または Kubernetes クラスター

チュートリアルのビルド

このチュートリアルは、Maven コマンドを使用してビルドします。

```
mvn package
```

target/ ディレクトリーに、**docker** (生成された Dockerfile を含む) や、Kubernetes および OpenShift デプロイメントテンプレートを含む **classes/META-INF/jkube** などの追加のディレクトリーが含まれています。

ヒント

Docker デーモンがダウンしている場合、ビルドで Dockerfiles の処理が省略されます。手動でオンにするには、**docker** プロファイルを使用します。

チュートリアルを Kubernetes にデプロイする

これは JKube Maven プラグインによって処理されます。以下を呼び出してください。

```
mvn k8s:build k8s:push k8s:resource k8s:apply -Doptions.image=<IMAGE_NAME> (1)
```

(1) **IMAGE_NAME** は、Kubernetes にデプロイするコンテナーの FQN に置き換える必要があります。このコンテナーは、Kubernetes クラスター内からアクセスでき、プッシュする権限を持っているリポジトリーに作成する必要があります。

表示とスケールアップ

この時点ですべてが稼働しているはずです。OpenShift または Kubernetes クラスターにログインしてアプリケーションをスケールします。

```
kubectl scale --replicas=3  deployment/$(kubectl get rs --namespace=myproject | grep infinispan |
awk '{print $1}') --namespace=myproject
```

チュートリアルのアンデプロイ
これは JKube Maven プラグインによって処理されます。以下を呼び出してください。

```
mvn k8s:undeploy
```

第3章 SPRING および SPRING BOOT

3.1. SPRING および SPRING BOOT のチュートリアル

注記

これらのコードチュートリアルでは、Data Grid Server を使用し、1つ以上の実行中のインスタンスが必要です。

Spring サンプルの実行

Spring Boot を使用せずに Spring を使用して 2 つの簡単なチュートリアルを実行できます。

- テストキャッシュ

```
$ {package_exec}@spring-caching
```

- テストのアノテーション

```
$ {package_exec}@spring-annotations
```

Spring Boot サンプルの実行

```
$ mvn -s /path/to/maven-settings.xml spring-boot:run
```

actuator 統計の表示

ブラウザーで**http://localhost:8080/actuator/metrics** に移動し、利用可能なメトリックのリストを表示します。キャッシュメトリクスの前に **cache** が付いています。タグを使用して各キャッシュの各メトリックを表示します。たとえば、basque-names キャッシュの puts 統計の場合は、以下のようになります。

http://localhost:8080/actuator/metrics/cache.puts?tag=name:basque-names

Prometheus を使用した統計の収集

このプロジェクトの **prometheus.yml** ファイルには、Prometheus が Spring アクチュエーターが公開するメトリックを取得できるように、**host.docker.internal** バインディングが含まれます。

以下のコマンドの **YOUR_PATH** の値を、Prometheus が実行されているディレクトリーに変更してから実行します。

Podman

```
$ podman run -d --name=prometheus -p 9090:9090 -v YOUR_PATH/integrations/spring-boot/prometheus.yml:/etc/prometheus/prometheus.yml prom/prometheus --config.file=/etc/prometheus/prometheus.yml
```

チュートリアルのリンク	説明

チュートリアルのリンク	説明
Spring Boot および Spring Cache のリモートモード	Spring Boot と Data Grid Server で Spring Cache を使用する方法を示します。
Spring Boot および Spring Session のリモートモード	Spring Boot と Data Grid Server で Spring Session を使用する方法を示します。
Spring Boot および Spring Cache の埋め込みモード	Spring Boot および Data Grid Embedded で Spring Cache を使用する方法を示します。
Spring Boot および Spring Session の埋め込みモード	Spring Boot および Data Grid Embedded で Spring Session を使用する方法を示します。
Spring Boot なしで埋め込みまれた Spring Cache	Spring Boot なしで埋め込まれた Spring Cache と Data Grid の使用方法を示します。

Data Grid のドキュメント

その他の資料は、以下のドキュメントを参照してください。

- Spring での Data Grid の使用
- Data Grid Spring Boot スターター