Red Hat Data Grid 8.4

Spring での Data Grid の使用

Spring アプリケーションにデータグリッドを追加する

Last Updated: 2024-04-30

Red Hat Data Grid 8.4 Spring での Data Grid の使用

Spring アプリケーションにデータグリッドを追加する

法律上の通知

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

Data Grid キャッシュ機能を Spring ベースのアプリケーションに追加します。

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。

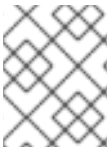
DATA GRID のドキュメント

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

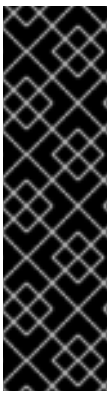
Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ を参照してください。

第1章 SPRING CACHE プロバイダーとして DATA GRID を使用する

Data Grid の依存関係をアプリケーションに追加し、Spring Cache アノテーションを使用して埋め込みキャッシュまたはリモートキャッシュにデータを保存します。

1.1. DATA GRID での SPRING キャッシュの設定

Spring アプリケーションプロジェクトに Data Grid の依存関係を追加します。Data Grid Server デプロイメントでリモートキャッシュを使用する場合は、Hot Rod クライアントプロパティーも設定する必要があります。



重要

Data Grid は Spring バージョン 5 とバージョン 6 をサポートします。Spring 6 には Java 17 が必要であることに注意してください。

このドキュメントの例には、Spring の最新バージョンのアーティファクトが含まれています。Spring 5 を使用したい場合は、以下を使用します。

- リモートキャッシュ: **infinispan-spring5-remote**
- 埋め込みキャッシュ: **infinispan-spring5-embedded**

手順

1. Data Grid と Spring 統合モジュールを **pom.xml** に追加します。
   - リモートキャッシュ: **infinispan-spring6-remote**
   - 埋め込みキャッシュ: **infinispan-spring6-embedded**

   ヒント

   Spring Boot ユーザーは、**infinispan-spring6-embedded** の代わりに次のアーティファクトを追加できます。

   - Spring Boot 3 の場合は、**infinispan-spring-boot3-starter-embedded** を追加します。
   - Spring Boot 2.x の場合は、**infinispan-spring-boot-starter-embedded** を追加します。

2. **hotrod-client.properties** ファイルで Data Grid Server デプロイメントに接続するように Hot Rod クライアントを設定します。

```
infinispan.client.hotrod.server_list = 127.0.0.1:11222
infinispan.client.hotrod.auth_username=admin
infinispan.client.hotrod.auth_password=changeme
```

Spring Cache の依存関係

リモートキャッシュ

```
<dependencies>
```

```
    <dependency>
      <groupId>org.infinispan</groupId>
      <artifactId>infinispan-spring6-remote</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>${version.spring}</version>
    </dependency>
</dependencies>
```

埋め込みキャッシュ

```
<dependencies>
    <dependency>
      <groupId>org.infinispan</groupId>
      <artifactId>infinispan-spring6-embedded</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>${version.spring}</version>
    </dependency>
</dependencies>
```

関連情報

- Hot Rod クライアント接続の設定

1.2. SPRING CACHE プロバイダーとして DATA GRID を使用する

@Enable Caching アノテーションを設定クラスの 1 つに追加してから、**@Cacheable** と **@CacheEvict** アノテーションを追加して、リモートキャッシュまたは埋め込みキャッシュを使用します。

前提条件

- Data Grid の依存関係をアプリケーションプロジェクトに追加します。
- Data Grid Server デプロイメントを使用する場合は、必要なリモートキャッシュを作成し、Hot Rod クライアントプロパティーを設定します。

手順

1. 以下のいずれかの方法で、アプリケーションコンテキストでキャッシュアノテーションを有効にします。

   宣言的

   ```
   <beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:cache="http://www.springframework.org/schema/cache"
       xmlns:p="http://www.springframework.org/schema/p"
   ```

```
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/cache
http://www.springframework.org/schema/cache/spring-cache.xsd">

        <cache:annotation-driven />

</beans>
```

プログラマティック

```
@EnableCaching @Configuration
public class Config {
}
```

2. メソッドに **@Cacheable** アノテーションを付けて、戻り値をキャッシュします。

   ヒント

   キャッシュのエントリーを直接参照するには、**key** 属性を含める必要があります。

3. メソッドに **@CacheEvict** のアノテーションを付け、キャッシュから古いエントリーを削除します。

関連情報

- Spring Framework - デフォルトのキー生成

1.3. SPRING キャッシュアノテーション

@Cacheable および **@CacheEvict** アノテーションは、メソッドにキャッシュ機能を追加します。

@Cacheable
戻り値をキャッシュに保存します。

@CacheEvict
古いエントリーを削除してキャッシュサイズを制御します。

@Cacheable
Book オブジェクトを例とすると、**BookDao#findBook(Integer bookId)** などのメソッドのデータベースから読み込み後に各インスタンスをキャッシュしたい場合は、以下のように **@Cacheable** アノテーションを追加します。

```
@Transactional
@Cacheable(value = "books", key = "#bookId")
public Book findBook(Integer bookId) {...}
```

上記の例では、**findBook(Integer bookId)** が **books** という名前のキャッシュに保存されている **Book** インスタンスを返す場合。

@CacheEvict

@CacheEvict アノテーションを使用すると、**books** キャッシュ全体をエビクトするか、特定の #bookId に一致するエントリーのみを削除するかどうかを指定できます。

キャッシュエビクション全体

deleteAllBookEntries() メソッドに **@CacheEvict** のアノテーションを付け、以下のように **allEntries** パラメーターを追加します。

```
@Transactional
@CacheEvict (value="books", key = "#bookId", allEntries = true)
public void deleteAllBookEntries() {...}
```

エントリーベースのエビクション

deleteBook(Integer bookId) メソッドに **@CacheEvict** のアノテーションを付け、以下のようにエントリーに関連付けられたキーを指定します。

```
@Transactional
@CacheEvict (value="books", key = "#bookId")
public void deleteBook(Integer bookId) {...}
```

1.4. キャッシュ操作のタイムアウトの設定

Data Grid Spring Cache プロバイダーは、読み取りおよび書き込み操作の実行時にデフォルトでブロック動作に設定されます。キャッシュ操作は同期され、タイムアウトしません。

必要な場合は、操作がタイムアウトするまでに待機する最大時間を設定できます。

手順

- **SpringEmbeddedCacheManagerFactoryBean** または **SpringRemoteCacheManagerFactoryBean** のいずれかで、アプリケーションのコンテキスト XML で、以下のタイムアウトプロパティーを設定します。
  リモートキャッシュの場合は、**hotrod-client.properties** ファイルにこれらのプロパティーを追加することもできます。

プロパティー	説明
infinispan.spring.operation.read.timeout	読み取り操作が完了するまでの待機時間をミリ秒単位で指定します。デフォルトは **0** で、待機時間が無制限であることを意味します。
infinispan.spring.operation.write.timeout	書き込み操作が完了するまでの待機時間をミリ秒単位で指定します。デフォルトは **0** で、待機時間が無制限であることを意味します。

以下の例は、**SpringRemoteCacheManagerFactoryBean** のコンテキスト XML のタイムアウトプロパティーを示しています。

```
<bean id="springRemoteCacheManagerConfiguredUsingConfigurationProperties"
    class="org.infinispan.spring.remote.provider.SpringRemoteCacheManagerFactoryBean">
  <property name="configurationProperties">
    <props>
```

```
            <prop key="infinispan.spring.operation.read.timeout">500</prop>
            <prop key="infinispan.spring.operation.write.timeout">700</prop>
        </props>
    </property>
</bean>
```

第2章 SPRING セッションでのセッションの外部化

Spring アプリケーションのセッションデータを Data Grid キャッシュに、コンテナーとは独立して保存します。

2.1. SPRING セッションでのセッションの外部化

Spring Session API を使用して、セッションデータを Data Grid に外部化します。

手順

1. **pom.xml** に依存関係を追加します。
   - 埋め込みキャッシュ: **infinispan-spring6-embedded**
   - リモートキャッシュ: **infinispan-spring6-remote**
     以下は、リモートキャッシュの例になります。

```
<dependencies>
  <dependency>
    <groupId>org.infinispan</groupId>
    <artifactId>infinispan-core</artifactId>
  </dependency>
  <dependency>
    <groupId>org.infinispan</groupId>
    <artifactId>infinispan-spring6-remote</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>${version.spring}</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.session</groupId>
    <artifactId>spring-session-core</artifactId>
    <version>${version.spring}</version>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>${version.spring}</version>
  </dependency>
</dependencies>
```

2. 適切な **FactoryBean** を指定して、**CacheManager** インスタンスを公開します。
   - 埋め込みキャッシュ: **SpringEmbeddedCacheManagerFactoryBean**
   - リモートキャッシュ: **SpringRemoteCacheManagerFactoryBean**
3. 適切なアノテーションで Spring Session を有効にします。
   - 埋め込みキャッシュ: **@EnableInfinispanEmbeddedHttpSession**
   - リモートキャッシュ: **@EnableInfinispanRemoteHttpSession**

これらのアノテーションには、オプションのパラメーターがあります。

- **maxInactiveIntervalInSeconds** は、セッションの有効期限を秒単位で設定します。デフォルトは **1800** です。
- **cacheName** は、セッションを格納するキャッシュの名前を指定します。デフォルトは **sessions** です。

以下の例は、完全なアノテーションベースの設定を示しています。

```
@EnableInfinispanEmbeddedHttpSession
@Configuration
public class Config {

  @Bean
  public SpringEmbeddedCacheManagerFactoryBean springCacheManager() {
    return new SpringEmbeddedCacheManagerFactoryBean();
  }

  //An optional configuration bean responsible for replacing the default
  //cookie that obtains configuration.
  //For more information refer to the Spring Session documentation.
  @Bean
  public HttpSessionIdResolver httpSessionIdResolver() {
     return HeaderHttpSessionIdResolver.xAuthToken();
  }
}
```