# Red Hat Data Grid 7.3

## Data Grid Spring Boot スターター

Data Grid を使用した Spring Boot アプリケーションのビルド

Last Updated: 2022-02-07

# Red Hat Data Grid 7.3 Data Grid Spring Boot スターター

Data Grid を使用した Spring Boot アプリケーションのビルド

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

## 法律上の通知

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Data_Grid_Spring_Boot_Starter.ent file |.

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 で Spring Boot プロジェクトを迅速に稼働させます。

# 目次

**前書き** . . . . . 3

**第1章 プロジェクトの設定** . . . . . 4
1.1. SPRING BOOT STARTER バージョン 4
1.2. 使用モードの依存関係の追加 4

**第2章 埋め込みモードでの実行** . . . . . 5
2.1. EMBEDDEDCACHEMANAGER BEAN の追加 5
2.2. キャッシュマネージャーの設定 BEAN 5
2.3. SPRING キャッシュサポートの有効化 6

**第3章 サーバーモードでの実行** . . . . . 7
3.1. REMOTECACHEMANAGER の設定 7
3.2. キャッシュマネージャーの設定 BEAN 7
3.3. SPRING キャッシュサポートの有効化 8
3.4. DATA GRID 統計の公開 8

**第4章 SPRINGSESSION の使用** . . . . . 10
4.1. SPRING SESSION サポートの有効化 10

**第5章 アプリケーションのプロパティー** . . . . . 11

# 前書き

Data Grid スターターは、Data Grid とシームレスに対話するために必要なすべての Spring Boot プロジェクトを含む、管理された推移的な依存関係のセットを提供します。

### ヒント

Data Grid Spring Boot スターターは、Spring Boot を使い始めるための便利な方法を提供しますが、これはオプションです。Spring Boot で Data Grid を使用するには、必要な依存関係を追加するだけです。

# 第1章 プロジェクトの設定

Data Grid Spring Boot スターターの依存関係をプロジェクトに追加します。

## 1.1. SPRING BOOT STARTER バージョン

Data Grid は Spring Boot 1.5.x および 2.x をサポートします。Spring Boot 1.5.x の場合は、Spring 4 の依存関係を使用します。Spring Boot 2.x の場合は、Spring 5 の依存関係を使用します。

- Spring Boot 1.5.x:**${version.infinispan.starter}** を 1.0. **7.Final-redhat-00021** に置き換えます。
- Spring Boot 2.x:**${version.infinispan.starter}** を **2.1.10.Final-redhat-00007** に置き換えます。

## 1.2. 使用モードの依存関係の追加

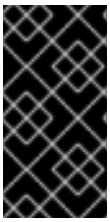
Data Grid は、使用モードごとに異なる依存関係を提供します。次のいずれかを **pom.xml** ファイルに追加します。

**埋め込みモード**

```
<dependency>
  <groupId>org.infinispan</groupId>
  <artifactId>infinispan-spring-boot-starter-embedded</artifactId>
  <version>${version.infinispan.starter}</version>
</dependency>
```

**リモートクライアント/サーバーモード**

```
<dependency>
  <groupId>org.infinispan</groupId>
  <artifactId>infinispan-spring-boot-starter-remote</artifactId>
  <version>${version.infinispan.starter}</version>
</dependency>
```



**重要**

Data Grid Spring Boot スターターは、Red Hat OpenShift Application Runtimes などの他のプロジェクトとは異なる Spring Boot バージョンを使用します。他のプロジェクトと互換性を確保するために特定の Spring Boot バージョンを使用する場合は、プロジェクトに正しい依存関係を追加する必要があります。

# 第2章 埋め込みモードでの実行

インメモリーデータストレージ用にプロジェクトに Data Grid ライブラリーを埋め込みます。

## 2.1. EMBEDDEDCACHEMANAGER BEAN の追加

1. **infinispan-spring-boot-starter-embedded** をプロジェクトのクラスパスに追加して、埋め込みモードを有効にします。
   このスターターは、デフォルトでクラスパス上の **infinispan-spring-boot-starter-remote** で Remote Client/Server モードで動作します。

2. 次の例のように、Spring **@Autowired** アノテーションを使用して、Java 構成クラスに **EmbeddedCacheManagerBean** を含めます。

```
private final EmbeddedCacheManager cacheManager;

@Autowired
public YourClassName(EmbeddedCacheManager cacheManager) {
   this.cacheManager = cacheManager;
}
```

   これで、埋め込みモードで Data Grid を使用できるようになりました。以下は簡単な例です。

```
cacheManager.getCache("testCache").put("testKey", "testValue");
System.out.println("Received value from cache: " +
cacheManager.getCache("testCache").get("testKey"));
```

## 2.2. キャッシュマネージャーの設定 BEAN

以下の設定 Bean を使用してキャッシュマネージャーをカスタマイズします。

- **InfinispanGlobalConfigurer**
- **InfinispanCacheConfigurer**
- **設定**
- **InfinispanConfigurationCustomizer**
- **InfinispanGlobalConfigurationCustomizer**

**注記**

**InfinispanGlobalConfigurerBean** は 1 つしか作成できません。ただし、他の Bean を使用して複数の構成を作成できます。

### InfinispanCacheConfigurer Bean

```
@Bean
public InfinispanCacheConfigurer cacheConfigurer() {
 return manager -> {
  final Configuration ispnConfig = new ConfigurationBuilder()
                    .clustering()
```

```
                    .cacheMode(CacheMode.LOCAL)
                    .build();

  manager.defineConfiguration("local-sync-config", ispnConfig);
 };
}
```

### 設定 Bean

次のように、設定するキャッシュに Bean 名をリンクします。

```
@Bean(name = "small-cache")
public org.infinispan.configuration.cache.Configuration smallCache() {
    return new ConfigurationBuilder()
        .read(baseCache)
        .memory().size(1000L)
        .memory().evictionType(EvictionType.COUNT)
        .build();
}

@Bean(name = "large-cache")
public org.infinispan.configuration.cache.Configuration largeCache() {
    return new ConfigurationBuilder()
        .read(baseCache)
        .memory().size(2000L)
        .build();
}
```

### カスタマイザー Bean

```
@Bean
public InfinispanGlobalConfigurationCustomizer globalCustomizer() {
   return builder -> builder.transport().clusterName(CLUSTER_NAME);
}

@Bean
public InfinispanConfigurationCustomizer configurationCustomizer() {
   return builder -> builder.memory().evictionType(EvictionType.COUNT);
}
```

## 2.3. SPRING キャッシュサポートの有効化

**@EnableCaching** アノテーションをアプリケーションに追加し、Spring Cache のサポートを有効にします。

このスターターが **EmbeddedCacheManager** Bean を検出すると、Spring Cache の実装を提供する新しい **SpringEmbeddedCacheManager** をインスタンス化します。

# 第3章 サーバーモードでの実行

カスタム TCP バイナリーワイヤプロトコルである Hot Rod を使用して、リモートの Data Grid クラスターからデータを保存および取得します。

## 3.1. REMOTECACHEMANAGER の設定

1. スターターが **RemoteCacheManager** Bean を作成できるように Data Grid サーバーの場所を指定します。
   このスターターは、最初にクラスパスの **hotrod-client.properties** ファイルからサーバーを見つけようとします。見つからない場合は、スターターは **application.properties** ファイルからサーバーの場所の特定を試みます。

   - **hotrod-client.properties**:

     ```
     infinispan.client.hotrod.server_list=127.0.0.1:11222
     ```

   - **application.properties**:

     ```
     infinispan.remote.server-list=127.0.0.1:11222
     ```

2. Spring **@Autowired** アノテーションを使用して、独自のカスタムキャッシュマネージャークラスをアプリケーションに含めます。

   ```
   private final RemoteCacheManager cacheManager;

   @Autowired
   public YourClassName(RemoteCacheManager cacheManager) {
      this.cacheManager = cacheManager;
   }
   ```

## 3.2. キャッシュマネージャーの設定 BEAN

次の設定 Bean を使用してキャッシュマネージャーをカスタマイズします。

- **InfinispanRemoteConfigurer**
- **設定**
- **InfinispanRemoteCacheCustomizer**

**注記**

**InfinispanRemoteConfigurerBean** は 1 つしか作成できません。ただし、他の Bean を使用して複数の構成を作成できます。

### InfinispanRemoteConfigurer Bean

```
@Bean
public InfinispanRemoteConfigurer infinispanRemoteConfigurer() {
   return () -> new ConfigurationBuilder()
      .addServer()
```

```
        .host("127.0.0.1")
        .port(12345)
        .build();
}
```

**設定 Bean**

```
@Bean
public org.infinispan.client.hotrod.configuration.Configuration customConfiguration() {
    new ConfigurationBuilder()
        .addServer()
        .host("127.0.0.1")
        .port(12345)
        .build();
}
```

**InfinispanRemoteCacheCustomizer Bean**

```
@Bean
public InfinispanRemoteCacheCustomizer customizer() {
    return b -> b.tcpKeepAlive(false);
}
```

**ヒント**

**@Ordered** アノテーションを使用して、カスタマイザーを特定の順序で適用します。

## 3.3. SPRING キャッシュサポートの有効化

**@EnableCaching** アノテーションをアプリケーションに追加し、Spring Cache のサポートを有効にします。

Data Grid スターターが **RemoteCacheManager** Bean を検出すると、Spring Cache の実装を提供する新しい **SpringRemoteCacheManager** をインスタンス化します。

## 3.4. DATA GRID 統計の公開

Data Grid は、Spring Boot Actuator をサポートして、キャッシュ統計をメトリクスとして公開します。

Actuator を使用するには、以下を **pom.xml** ファイルに追加します。

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
  <version>${version.spring.boot}</version>
 </dependency>

<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
  <version>${version.spring.boot}</version>
</dependency>
```

次に、プログラムまたは宣言的に、適切なキャッシュインスタンスの統計をアクティベートする必要があります。

### プログラムで

```
@Bean
public InfinispanCacheConfigurer cacheConfigurer() {
  return cacheManager -> {
    final org.infinispan.configuration.cache.Configuration config =
        new ConfigurationBuilder()
            .jmxStatistics().enable()
            .build();

    cacheManager.defineConfiguration("my-cache", config);
  };
}
```

### 宣言

```
<local-cache name="my-cache" statistics="true"/>
```

Spring Boot Actuator レジストリーは、アプリケーションの起動時にキャッシュインスタンスをバインドします。キャッシュを動的に作成する場合は、次のように、 **CacheMetricsRegistrarBean** を使用してキャッシュを Actuator レジストリーにバインドする必要があります。

```
@Autowire
CacheMetricsRegistrar cacheMetricsRegistrar;

@Autowire
CacheManager cacheManager;
...

cacheMetricsRegistrar.bindCacheToRegistry(cacheManager.getCache("my-cache"));
```

# 第4章 SPRINGSESSION の使用

## 4.1. SPRING SESSION サポートの有効化

Data Grid Spring Session サポートは **SpringRemoteCacheManager** と **SpringEmbeddedCacheManager** をもとに構築されます。このスターターは、デフォルトでこれらの Bean を生成します。

プロジェクトで Spring セッションを使用するには、以下を実行します。

1. このスターターをプロジェクトに追加します。
2. Spring Session をクラスパスに追加します。
3. 次のアノテーションを設定に追加します。
   - **@EnableCaching**
   - **@EnableInfinispanRemoteHttpSession**
   - **@EnableInfinispanEmbeddedHttpSession**

# 第5章 アプリケーションのプロパティー

**application.properties** または **application.yaml** を使用してプロジェクトを構成します。

```
# List Infinispan or Data Grid servers by IP address or hostname at port 11222.
infinispan.remote.server-list=127.0.0.1:11222

#
# Embedded Properties - Uncomment properties to use them.
#

# Enables Infinispan or Data Grid capabilities in your application.
# Values are true (default) or false.
#infinispan.embedded.enabled =

# Sets the Spring state machine ID.
#infinispan.embedded.machineId =

# Sets the name of the Infinispan or Data Grid cluster.
#infinispan.embedded.clusterName =

# Specifies a XML configuration file that takes priority over the global
# configuration bean or any configuration customizer.
#infinispan.embedded.configXml =

#
# Server Properties - Uncomment properties to use them.
#

# Specifies a custom filename for Hot Rod client properties.
#infinispan.remote.clientProperties =

# Enables remote Infinispan or Data Grid servers.
# Values are true (default) or false.
#infinispan.remote.enabled =

# Defines a comma-separated list of Infinispan or Data Grid servers
# in this format: `host1[:port],host2[:port]`.
#infinispan.remote.serverList =

# Sets a timeout value, in milliseconds, for socket connections.
#infinispan.remote.socketTimeout =

# Sets a timeout value for initializing connections with Infinispan or
# Data Grid servers.
#infinispan.remote.connectTimeout =

# Sets the maximum number of attempts to connect to Infinispan or
# Data Grid servers.
#infinispan.remote.maxRetries =
```