Redis应用开发

www.huawei.com

HUAWEI

目标

- 学完本课程后，您将能够:
  - 了解**Redis**应用场景
  - 掌握**Redis**二次开发环境搭建
  - 掌握**Redis**业务开发

目录

1. **Redis应用场景介绍**
2. Redis业务流程
3. Redis应用开发
4. 应用开发案例分析
5. 常用接口示例

Redis简介

- **Redis**是一个基于网络的，高性能**key-value**内存数据库。
- **Redis**跟**memcached**类似，不过数据可以持久化，而且支持的数据类型很丰富。支持在服务器端计算集合的并、交和补集**(difference)**等，还支持多种排序功能。
- **Redis**使用场景有如下几个特点：
  - 高性能。
  - 低时延。
  - 丰富数据结构存取。
  - 支持持久化。

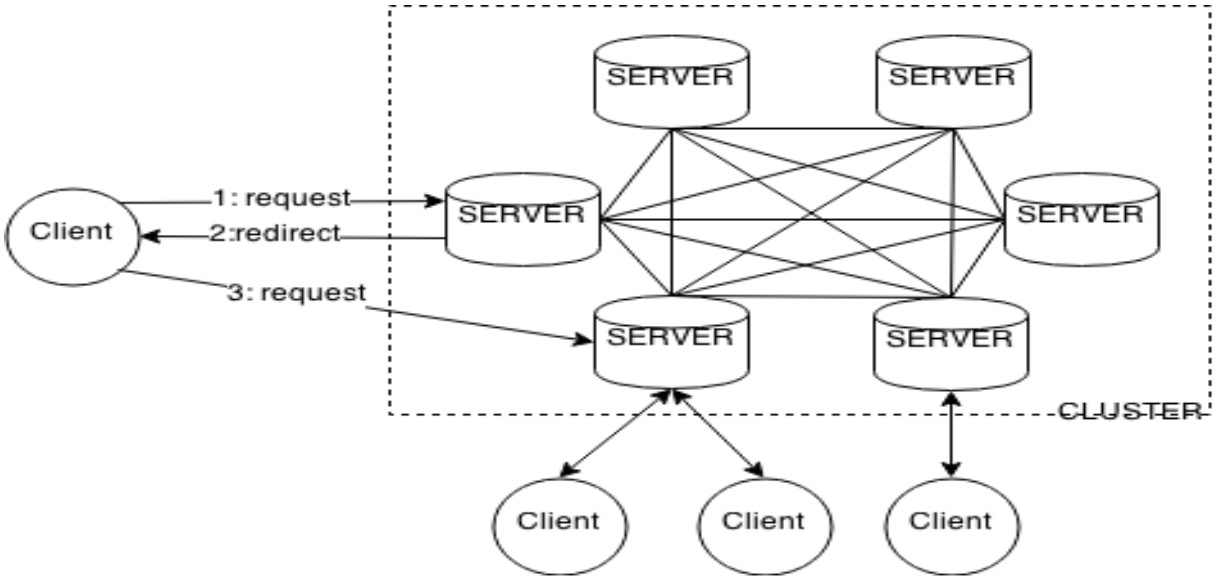
Redis应用场景介绍

- **Redis**提供了灵活多变的数据结构和数据操作，主要应用于如下场景：
  - 取最新**N**个数据的操作，比如典型的取某网站的最新文章。
  - 排行榜应用，取**TOP N**操作。这个需求与上面需求的不同之处在于，前面操作以时间为权重，这个是以某个条件为权重，比如按顶的次数排序。
  - 需要精准设定过期时间的应用，如用户会话信息。
  - 计数器应用，比如记录用户访问网站次数。
  - 构建队列系统，例如消息队列。
  - 缓存，如缓存关系数据库中的频繁访问的表数据。

目录

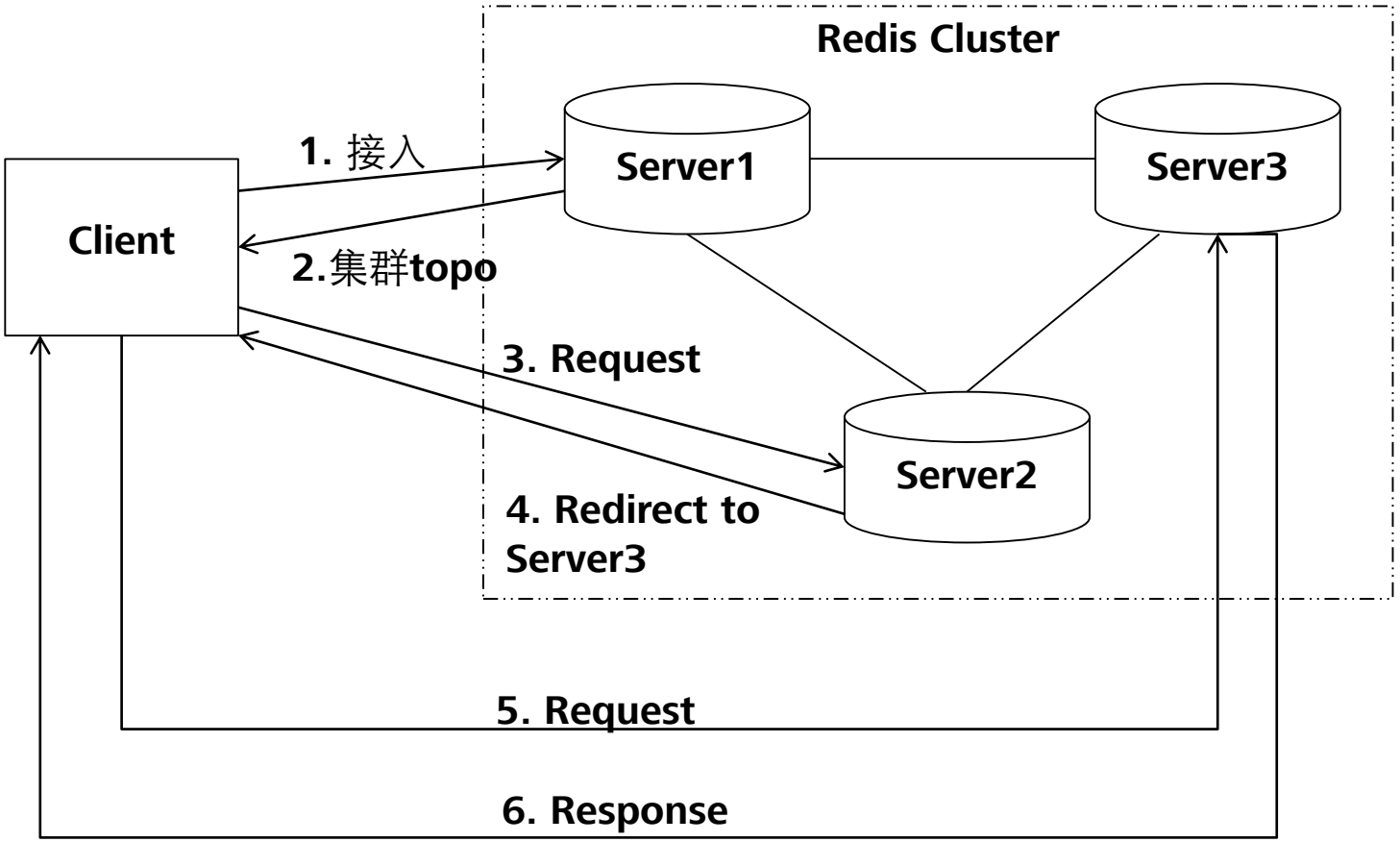
1. Redis应用场景介绍
2. **Redis业务流程**
3. Redis应用开发
4. 应用开发案例分析
5. 常用接口示例

架构回顾



- 无中心自组织的结构，节点之间使用**Gossip**协议来交换节点状态信息。
- 各节点维护**Key->Server**的映射关系。
- **Client**可以向任意节点发起请求，节点不会转发请求，只是重定向**Client**。
- 如果在**Client**第一次请求和重定向请求之间，**Cluster**拓扑发生改变，则第二次重定向请求将被再次重定向，直到找到正确的**Server**为止。

Redis数据读写流程



Redis数据读写流程

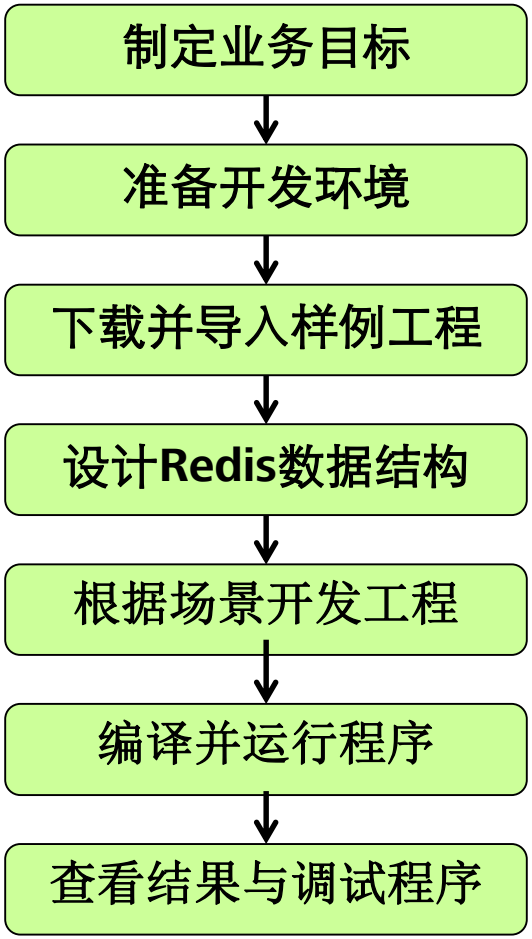
- **Redis**数据读写流程如下：

  1. 客户端选择集群中任意一个**Server**节点进行连接，并发送**cluster nodes**请求；
  2. **Server**节点返回集群拓扑，主要包括集群节点列表及槽位跟节点的映射关系，客户端在内存中缓存集群拓扑；
  3. 客户端读写数据时，根据**hash(KEY)%16384**计算得到**KEY**归属的槽位，再查槽位跟节点的映射，进一步得到**KEY**归属的节点**Server2**，直接访问该节点进行数据读写；
  4. **Server2**收到客户端的请求，检查自身是否为**KEY**归属的节点：若不是，则响应中告知**Client**需重定向的节点**Server3**；若是，则直接返回业务操作结果；
  5. 客户端收到重定向响应，重新向**Server3**发起读写请求；
  6. **Server3**收到请求，处理过程同步骤**4**。

目录

1. Redis应用场景介绍
2. Redis业务流程
3. **Redis应用开发**
4. 应用开发案例分析
5. 常用接口示例

Redis应用开发流程



Redis应用开发流程

- 制定业务目标
  - 数据量?
  - 读写性能要求?
  - 数据是否需要持久化?

Redis应用开发流程

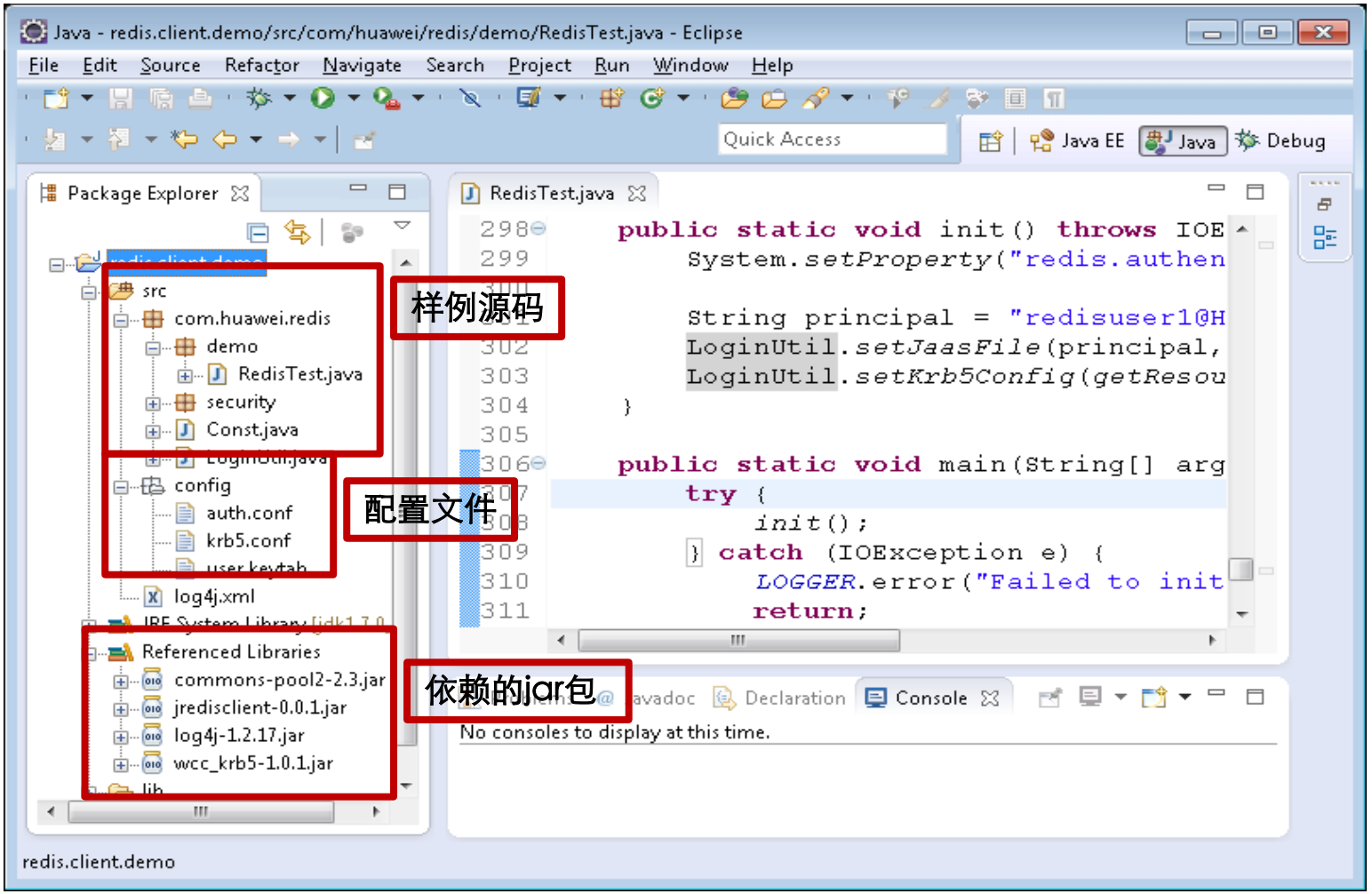
- 准备开发环境

准备项	说明
操作系统	**Windows**系统，推荐**Windows 7**以上版本。
安装**JDK**	开发环境的基本配置。版本要求：**1.7**或者**1.8**。
安装和配置**Eclipse**	用于开发**Redis**应用程序的工具。
网络	确保客户端与**Redis**服务主机在网络上互通。
Redis集群	登录**FusionInsight Manager**，在“服务管理 > **Redis** > **Redis**管理”页面创建**Redis**集群。
Redis用户	创建**Redis**用户，并获取其认证凭据文件，用于登录**FusionInsight**平台并通过认证。

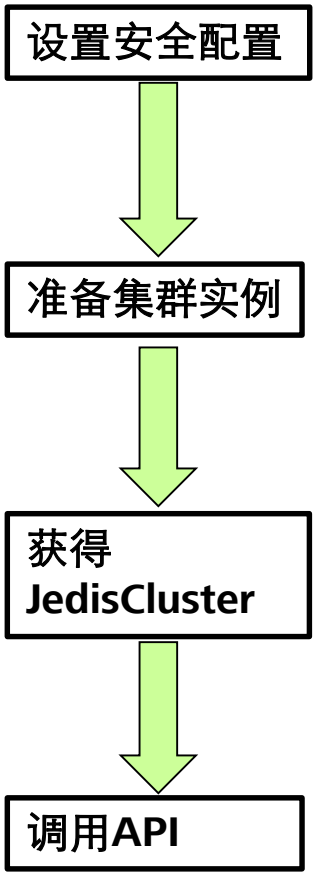
Redis应用开发流程

- 下载并导入样例工程
  - 下载并解压**Redis**客户端压缩包。
  - 在**FusionInsight Manager**页面新建用户，用于登陆与操作。
  - 下载用户的认证凭据文件。
  - 导入样例工程到**Eclipse**开发环境。
  - 配置认证凭据文件到**Redis**客户端样例工程。

Redis应用开发流程



一个例子



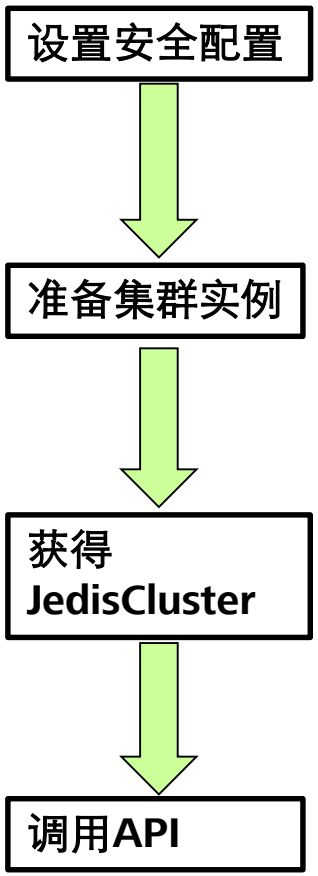

```
System.setProperty("redis.authentication.jaas", "true");

String principal = "redisuser@HADOOP.COM";
LoginUtil.setJaasFile(principal,
getResource("config/user.keytab"));
LoginUtil.setKrb5Config(getResource("config/krb5.conf"));
```

```
Set<HostAndPort> hosts = new HashSet<HostAndPort>();
hosts.add(new HostAndPort(Const.IP_1, Const.PORT_1));
hosts.add(new HostAndPort(Const.IP_2, Const.PORT_2));
// add more host...
```

```
// socket timeout(connect, read), unit: ms
int timeout = 5000;
client = new JedisCluster(hosts, timeout);
```

```
client.setex(key, 5, "A0BC9869FBC92933255A37A1D21167B2");
String sessionId = client.get(key);
//...
client.rpush(key, "Fine thanks. I'm having fun with redis.");
//...
client.hset(key, "name", "John");
```

一个例子




```
System.setProperty("redis.authentication.jaas", "true");

String principal = "redisuser@HADOOP.COM";
LoginUtil.setJaasFile(principal,
getResource("config/user.keytab"));
LoginUtil.setKrb5Config(getResource("config/krb5.conf"));
```

```
Set<HostAndPort> hosts = new HashSet<HostAndPort>();
hosts.add(new HostAndPort(Const.IP_1, Const.PORT_1));
hosts.add(new HostAndPort(Const.IP_2, Const.PORT_2));
// add more host...
```

```
// socket timeout(connect, read), unit: ms
int timeout = 5000;
client = new JedisCluster(hosts, timeout);
```

```
client.setex(key, 5, "A0BC9869FBC92933255A37A1D21167B2");
String sessionId = client.get(key);
//...
client.rpush(key, "Fine thanks. I'm having fun with redis.");
//...
client.hset(key, "name", "John");
```

Redis应用开发流程

- 设计**Redis**数据结构
  - 取最新**N**个数据的操作：**list**。
  - 排行榜应用，取**TOP N**操作。以某个条件为权重，比如按顶的次数排序：**sorted set**。
  - 利用**INCR**，**DECR**命令来构建计数器系统。
  - **Uniq**操作，获取某段时间所有数据排重值：**set**。
  - 使用**list**可以构建队列系统。

Redis应用开发流程

- 根据场景开发工程
  - 梳理业务场景流程
  - 设计各模块接口
  - 如果使用的是安全集群，需要进行安全认证
  - 熟悉**Redis**提供的相应**API**
  - 调用业务需要的**API**实现各功能

Redis应用开发流程

- 编译并运行程序
  - 在开发环境**Eclipse**中，右击**RedisTest.java**，选择“**Run as > Java Application**”运行对应的应用程序工程。

Redis应用开发流程

- 查看结果与调试程序
  - 在**Eclipse**的**Console**窗口可查看**Redis API**返回结果是否符合预期。
  - 也可通过安装客户端后使用**redis-cli**连接**redis**集群查看程序运行结果。（如数据是否已写入，写入的数据跟预期是否一致）。

目录

1. Redis应用场景介绍
2. Redis业务流程
3. Redis应用开发
4. **应用开发案例分析**
5. 常用接口示例

应用开发案例分析

- 业务目标
  - 某在线推荐业务对实时性具有较高的要求，主要体现为推荐结果的获取是跟随用户对业务的请求完成的，如果推荐结果的获取过程延时很高，势必会影响业务本身的体验。因此在线推荐业务需要考虑如何确保推荐结果获取的实时性。
- 业务方案
  - 引入**Redis**，增加**Cache**机制降低从数据库和文件系统读取数据的次数和数据量。
  - 将已经计算出的推荐结果缓存到**Redis**，下一次获取同一批推荐结果的时候从缓存直接读取。

应用开发案例分析

- 数据结构设计
  - 计算过程中使用的用户信息使用**hash**结构存取，**KEY**设计为**userinfo-<user id>**，**field**为用户的属性，如姓名、性别、年龄、爱好等。

    例如：userinfo-19810101,name,zhangsan

    userinfo-19810101,sex,female
  - 推荐结果（商品）由于可能存在多个，且为避免一个商品重复推荐，故使用**set**结果存取，**KEY**设计为**res-<user id>**。
- 数据读写原则
  - **MR**任务每天定时将用户信息从后端存储源（**HBase**）批量导入到**Redis**中。
  - 业务系统获取数据时，先从**Redis**中获取，获取不到再去后端**HBase**获取或即时计算，并同步写入到**Redis**。

目录

1. Redis应用场景介绍
2. Redis业务流程
3. Redis应用开发
4. 应用开发案例分析
5. 常用接口示例

样例说明--Redis集群初始化

- 通过指定集群中一个或多个实例的**IP**跟端口号，创建**JedisCluster**实例。

```
public RedisTest() {
    Set<HostAndPort> hosts = new HashSet<HostAndPort>();
    hosts.add(new HostAndPort(Const.IP_1, Const.PORT_1));
    hosts.add(new HostAndPort(Const.IP_2, Const.PORT_2));
    // add more host...

    // socket timeout(connect, read), unit: ms
    int timeout = 5000;
    client = new JedisCluster(hosts, timeout);
}
```

- 通过**FusionInsight Manager**，用户可以查看**Redis**集群包含的**Redis**实例。
- 创建**JedisCluster**实例时建议指定**3**个实例，且**3**个实例分布在不同主机。
- **Redis**节点中角色**Redis_1**对应的端口是**22400**， **Redis_2**对应的端口是**22401**，以此类推。
- 样例代码中的**Const.IP_1**、**Const.IP_2**、**Const.PORT_1**、**Const.PORT_2**的值请修改为实际环境的**IP**跟端口。

样例说明—String类型操作（1）

- 该样例介绍了**String**类型的常见操作：
  - **set:** 将字符串值 **value** 关联到 **key** 。
  - **get:**返回 **key** 所关联的字符串值。
  - **setex:** 将值**value**关联到**key**，并设置**key**的生存时间 **(**以秒为单位**)**。
  - **append:** 将**value**追加到**key**原值的末尾，类似**Java String**类型的 “**+**”操作。

样例说明—String类型操作（2）

```
// Save user's session ID, and set expire time
client.setex(key, 5, "A0BC9869FBC92933255A37A1D21167B2")
String sessionId = client.get(key);
LOGGER.info("User " + key + ", session id: " + sessionId
try {
    Thread.sleep(10000);
} catch (InterruptedException e) {
    LOGGER.warn("InterruptedException");
}

sessionId = client.get(key);


client.set(key, "hello");
String value = client.get(key);
LOGGER.info("Value: " + value);

client.append(key, " world");
value = client.get(key);
LOGGER.info("After append, value: " + value);
```

样例说明—List类型操作（1）

- 该样例介绍了**List**（列表）类型的常见操作：
  - **lpush/rpush:** 往列表头/尾添加一个元素。
  - **lrange:** 返回列表中**start**至**end**之间的元素（下标从**0**开始）。
  - **llen:** 获取列表的长度。
  - **lpop/rpop:** 从列表头/尾获取一个元素，并将其中列表中删除。

样例说明—List类型操作（2）

```
// Right push
client.rpush(key, "Hello how are you?");
client.rpush(key, "Fine thanks. I'm having fun with redi
client.rpush(key, "I should look into this NOSQL thing A

// Fetch all data
List<String> messages = client.lrange(key, 0, -1);
LOGGER.info("All messages: " + messages);

long len = client.llen(key);
LOGGER.info("Message count: " + len);

// Fetch the first element and delete it from list
String message = client.lpop(key);
LOGGER.info("First message: " + message);
len = client.llen(key);
```

样例说明—Hash类型操作（1）

- 该样例介绍了**Hash**类型的常见操作：
  - **hset:** 将哈希表 **key** 中的域 **field** 的值设为 **value**。
  - **hget:** 返回哈希表 **key** 中给定域 **field** 的值。
  - **hgetall:** 返回哈希表 **key** 中，所有的域和值。
  - **hmset:** 同时将多个 **field-value** (域-值)对设置到哈希表 **key** 中。
  - **hincrby:** 为哈希表 **key** 中的域 **field** 的值加上增量 **increment**。
  - **hkeys:** 返回哈希表 **key** 中的所有域。
  - **kvals:** 返回哈希表 **key** 中所有域的值。
  - **hmget:** 返回哈希表 **key** 中，一个或多个给定域的值。
  - **hexists:** 查看哈希表 **key** 中，给定域 **field** 是否存在。
  - **hdel:** 删除哈希表 **key** 中的一个或多个指定域，不存在的域将被忽略。

样例说明—Hash类型操作（2）

```
// like Map.put()
client.hset(key, "id", "J001");
client.hset(key, "name", "John");
client.hset(key, "gender", "male");
client.hset(key, "age", "35");
client.hset(key, "salary", "1000000");

// like Map.get()
String id = client.hget(key, "id");
String name = client.hget(key, "name");


Map<String, String> user2 = new HashMap<String, S
user2.put("id", "L002");
user2.put("name", "Lucy");
user2.put("gender", "female");
user2.put("age", "25");
user2.put("salary", "200000");
client.hmset(key, user2);
client.hincrBy(key, "salary", 50000);
id = client.hget(key, "id");
String salary = client.hget(key, "salary");
```

样例说明—Hash类型操作（3）

```
Set<String> keys = client.hkeys(key);
LOGGER.info("all fields: " + keys);
// like Map.values()
List<String> values = client.hvals(key);
LOGGER.info("all values: " + values);

// Fetch some fields
values = client.hmget(key, "id", "name");


// like Map.containsKey();
boolean exist = client.hexists(key, "gender");
LOGGER.info("Exist field gender? " + exist);

// like Map.remove();
client.hdel(key, "age");
keys = client.hkeys(key);
```

总结

- 本课程首先介绍了**Redis**组件的应用场景，业务数据读写流程，然后介绍了**Redis**应用的开发流程，并以一个实例介绍了**Redis**常用接口的使用方法。
- 学完本章后，可以根据具体的业务场景，选择合适的**Redis**数据结构，进行**Redis**数据的存取。

习题

- 某应用需要取**TOP N**操作，应该使用**Redis**的何种数据结构。
- **Redis Server**收到非归属本节点的**KEY**操作，会将请求转发到正确的节点上吗?

Thank you

www.huawei.com