2-Singleton

-
-
-
    -
    -
    -
    -
    -
- JDKSingleton
    -
    -
    -
    - DB9500E

JDKJDK

1.
2. ()

`public`()

,

```
public class Singleton {
    //
    private static Singleton instance = new Singleton();
    //,
    private Singleton() {
    }
    //
    public static Singleton getInstance() {
        return instance;
    }
}
```

classloderinstanceSingletoninstancelazy loadingSingletoninstanceSingletonHoldergetInstanceSingletonHolderinstance instanceSingletonSingletoninstance

```
public class StaticInnerClassSingleton {
    //
    private static class SingletonHolder {
        private static final StaticInnerClassSingleton INSTANCE = new
StaticInnerClassSingleton();
    }
    //
    private StaticInnerClassSingleton() {
    }
    //
    public static final StaticInnerClassSingleton getInstance() {
        return SingletonHolder.INSTANCE;
    }
}
```

lazygetInstancenew

synchronizedifinstance

```
public class SynchronizedSingleton {
    //
    private static SynchronizedSingleton instance;
    //
    private SynchronizedSingleton(){}
    //,
    public static synchronized SynchronizedSingleton getInstance() {
        //
        if (instance == null) {
            instance = new SynchronizedSingleton();
        }
        return instance;
    }
}
```

volatile

1. A
2. A
3. BBABAB()

```
public class VolatileSingleton {
    private static volatile VolatileSingleton singleton;

    private VolatileSingleton() {
    }

    public static VolatileSingleton getSingleton() {
        if (singleton == null) {
            synchronized (VolatileSingleton.class) {
                if (singleton == null) {
                    singleton = new VolatileSingleton();
                }
            }
        }
        return singleton;
    }
}
```

final

final

```
class FinalWrapper<T> {
    public final T value;

    public FinalWrapper(T value) {
        this.value = value;
    }
}

public class FinalSingleton {
    private FinalWrapper<FinalSingleton> helperWrapper = null;

    public FinalSingleton getHelper() {
        FinalWrapper<FinalSingleton> wrapper = helperWrapper;

        if (wrapper == null) {
            synchronized (this) {
                if (helperWrapper == null) {
                    helperWrapper = new FinalWrapper<FinalSingleton>(new
FinalSingleton());
                }
                wrapper = helperWrapper;
            }
        }
        return wrapper.value;
    }
}
```

JDKSingleton

JDKDB9500E

`Runtime`JavajavaJVMJVMRuntimeJVM Java Runtime

JavaJVMRuntime

java.lang.Runtime#getRuntime()

```
public class Runtime {
    private static Runtime currentRuntime = new Runtime();

    /**
     * Returns the runtime object associated with the current Java application.
     * Most of the methods of class <code>Runtime</code> are instance
     * methods and must be invoked with respect to the current runtime object.
     *
     * @return  the <code>Runtime</code> object associated with the current
     *          Java application.
     */
    public static Runtime getRuntime() {
        return currentRuntime;
    }
}
```

java.awt.Desktop#getDesktop()

```
public static synchronized Desktop getDesktop(){
    if (GraphicsEnvironment.isHeadless()) throw new HeadlessException();
    if (!Desktop.isDesktopSupported()) {
        throw new UnsupportedOperationException("Desktop API is not " +
                                                "supported on the current platform");
    }

    sun.awt.AppContext context = sun.awt.AppContext.getAppContext();
    Desktop desktop = (Desktop)context.get(Desktop.class);

    if (desktop == null) {
        desktop = new Desktop();
        context.put(Desktop.class, desktop);
    }

    return desktop;
}
```

java.lang.System#getSecurityManager()

```
public final class System {
 private static volatile SecurityManager security = null;
 public static SecurityManager getSecurityManager() {
     return security;
 }
}
```

DB9500E

DB9500E

DB9500E

```
public static JedisTemplate getRedisTemplate(String key, boolean isCluster) {
    JedisPool jedisPool;
    JedisTemplate jedisTemplate;
    String SentinelUrl;
    if (isCluster) {
        String[] values = key.split(",");
        StringBuilder sb = new StringBuilder();
        for (String value : values) {
            sb.append(value).append(":").append(SolrConsts.SENTINEL_PORT).append(",");
        }
        SentinelUrl = sb.toString();
        synchronized (redisMap) {
            jedisTemplate = redisMap.get(key);
            if (jedisTemplate == null) {
                String url =
                    "sentinel://" + SentinelUrl +
"?poolName=cds&poolSize=5&masterName="
                        + SolrConsts.SENTINEL_MASTER_NAME;
                jedisPool = new JedisPoolBuilder().setUrl(url).buildPool();
                jedisTemplate = new JedisTemplate(jedisPool);
                redisMap.put(key, jedisTemplate);
                logger.debug("the redis url is " + url);
            }
        }
    } else {
        synchronized (redisMap) {
            jedisTemplate = redisMap.get(key);
            if (jedisTemplate == null) {
                String url = "direct://" + key + ":" + SolrConsts.REDIS_PORT
                    + "?poolName=cds&poolSize=5";
                jedisPool = new JedisPoolBuilder().setUrl(url).buildPool();
                jedisTemplate = new JedisTemplate(jedisPool);
                redisMap.put(key, jedisTemplate);
                logger.debug("the redis url is " + url);
            }
        }
    }

    return jedisTemplate;
}
```