InterSystems公式
Toshihiko Minamoto · 2023年3月12日 3m read

# クラスの生成方法、呼び出し方法の変更（IRIS 2023.1〜）

IRIS 2023.1でのメソッドコードの生成と呼び出し方法の変更について、お知らせしたいと思います。

IRIS のクラスは、2 つの主要なランタイムコンポーネントで構成されています。

1. クラスディスクリプタ - クラスを構成するメソッド、プロパティ、クラスパラメータ、およびこれらのそれぞれに関連する属性（パブリック/プライベート設定など）が最適化されたリストです。
2. ObjectScriptコード - メソッドが呼び出されたときに実行される ObjectScript コードを含むルーチンのセットです。

クラス/オブジェクトのメソッドを呼び出すと、ディスパッチコードがクラス記述子からメソッドを探し、呼び出しが許可されているかどうかを確認し、正しいクラスコンテキストを設定し（その過程で $this を更新）、最後に関連するクラスルーチンにある ObjectScript コードを呼び出します。

IRIS 2023.1 では、ObjectScript の生成方法が改善され、常にクラス記述子を介してこのコードにディスパッチし、すべての正しいチェックを適用して、正しいクラスコンテキストで ObjectScript を実行するようになりました。この変更前は、ObjectScript コードを手動で直接呼び出すことができましたが、これは無効な結果や許可されていない ObjectScript コードを実行することにつながる可能性があります。

この変更の結果、IRIS 2023.1 以降、ObjectScript のメソッド コードを直接呼び出そうとすると、<NOLINE> エラーが報告されます (これは公式には許可されていませんでした)。

**例**

IRIS 2023.1以前は、以下のようなメソッドを持つクラスUser.Test.clsをコンパイルすると、

```
method Test() {
  Write "Test",!
}
```

「User.Test.1」ルーチンに次のようなObjectScriptコードが生成されます。

```
zTest() public {
  Write "Test",!
}
```

現状、これは通常のINTルーチンなので、このラベルを「Do zTest^User.Test.1() 」と呼び出すことができますが、これはクラスディスクリプタによる正しいディスパッチをバイパスするので、このメソッドがプライベートである場合に呼び出しが許可されているかどうかをチェックしません。また、クラス/オブジェクトコンテキストを設定しないので、クラス/オブジェクトコンテキストに依存するこのメソッドのロジックは失敗するか、予測できない結果を返す可能性があります。

IRIS 2023.1 では以下のように生成されます。

```
Test() methodimpl {
 Write "Test",!
}
```

このラベルはクラスディスクリプタを介してのみ呼び出し可能で、直接呼び出そうとするとランタイム<NOLINE>エラーが発生するようになりました。また、以前は%以外のメソッドではラベル名の前に'z'を付けて、ラベルを直接呼び出さないようにしていましたが、現在、これらのラベルは明示的に呼び出せないので、プロシージャブロックメソッドではこの'z'というプレフィックスを付けなくなり、生成コードの読みやすさが改善されました。

#ObjectScript #InterSystems IRIS #InterSystems公式

---

**ソースURL:**
https://jp.community.intersystems.com/post/%E3%82%AF%E3%83%A9%E3%82%B9%E3%81%AE%E7%94%9F%E6%88%90%E6%96%B9%E6%B3%95%E3%80%81%E5%91%BC%E3%81%B3%E5%87%BA%E3%81%97%E6%96%B9%E6%B3%95%E3%81%AE%E5%A4%89%E6%9B%B4%EF%BC%88iris-20231%EF%BD%9E%EF%BC%89