错误处理

概述

Hyperledger Fabric 代码应该使用 vendor 的包 **github.com/pkg/errors** 来代替 Go 提供的标准错误类型。该软件包允许生成和显示带有错误信息的堆栈跟踪。

使用说明

应该使用 **github.com/pkg/errors** 来代替对 *fmt.Errorf()* 或 *errors.New()* 的所有调用。使用此程序包将生成一个调用堆栈，该调用堆栈会被附加到错误信息上。

使用该程序包很简单，只需对您的代码做轻微调整。

首先，您需要引入 **github.com/pkg/errors**

然后，更新您的代码生成的所有错误，以使用其中一个错误创建函数（errors.New(), errors.Errorf(), errors.WithMessage(), errors.Wrap(), errors.Wrapf()）.

注解

可用的错误创建方法的完整文档请查看 https://godoc.org/github.com/pkg/errors 。也可以参考下边通用指南的章节来了解在 Fabric 代码中使用该包的更多指南。

最后，将所有记录器或 fmt.Printf() 调用的格式指令从 `%s` 更改为 `%+v` ，以打印调用堆栈和错误信息。

Hyperledger Fabric 中错误处理的通用准则

- 若要处理用户请求，应记录并返回错误。
- 若错误来自外部来源（如 Go 依赖库或 vendor 的包），则用 errors.Wrap() 包装该错误，为其生成一个调用堆栈。
- 若错误来自另一个 Fabric 函数，当有需要时，在不影响调用堆栈的情况下使用 errors.WithMessage() 在错误信息中添加更多上下文。
- Panic 不应被传播给其他软件包。

示例程序

以下案列程序清楚地展示了如何使用软件包：

```
package main

import (
  "fmt"

  "github.com/pkg/errors"
)
```

```
func wrapWithStack() error {
  err := createError()
  // do this when error comes from external source (go lib or vendor)
  return errors.Wrap(err, "wrapping an error with stack")
}
func wrapWithoutStack() error {
  err := createError()
  // do this when error comes from internal Fabric since it already has stack trace
  return errors.WithMessage(err, "wrapping an error without stack")
}
func createError() error {
  return errors.New("original error")
}

func main() {
  err := createError()
  fmt.Printf("print error without stack: %s\n\n", err)
  fmt.Printf("print error with stack: %+v\n\n", err)
  err = wrapWithoutStack()
  fmt.Printf("%+v\n\n", err)
  err = wrapWithStack()
  fmt.Printf("%+v\n\n", err)
}
```