

第17期-异常处理机制

2020年8月10日 13:01

异常的处理模型

- 自动 **处理所有异常**，如果是API/AJAX请求，会向客户端返回一个标准格式化的错误。
- 自动隐藏 **内部详细错误** 并返回标准错误消息。
- 为异常消息的 **本地化** 提供一种可配置的方式。
- 自动为标准异常设置 **HTTP状态代码**，并提供可配置选项，以映射自定义异常。

自动异常处理

当满足下面任意一个条件时，AbpExceptionFilter 会处理此异常。

- 当 action 方法返回类型是 object result (而不是view result) 并有异常抛出时。
- 当一个请求为AJAX(Http请求头中X-Requested-With为XMLHttpRequest)时。
- 客户端接受的返回类型为application/json(Http请求头中accept 为application/json) 时。

如果异常被处理过，则会自动记录日志并将格式化的 JSON 消息返回给客户端。

错误消息格式

每个错误消息都是 RemoteServiceErrorResponse 类的实例，序列化后返回。

```
{
  "error": {
    "message": "This topic is locked and can not add a new message"
  }
}
```

错误代码

如果自定义异常，应实现 IHasErrorCode 接口来填充错误代码字段。

```
{
  "error": {
    "code": "App:010042",
    "message": "This topic is locked and can not add a new message"
  }
}
```

错误详细信息

抛出的异常应实现 `IHasErrorDetails` 接口来填充错误详细信息字段。

```
{
  "error": {
    "code": "App:010042",
    "message": "This topic is locked and can not add a new message",
    "details": "A more detailed info about the error..."
  }
}
```

验证错误

当抛出的异常实现 `IHasValidationErrors` 接口时, `validationErrors` 是一个可被填充的标准字段。

```
{
  "error": {
    "code": "App:010046",
    "message": "Your request is not valid, please correct and try again!",
    "validationErrors": [{
      "message": "Username should be minimum length of 3.",
      "members": ["userName"]
    },
    {
      "message": "Password is required",
      "members": ["password"]
    }
  ]
}
```

`AbpValidationException` 已经实现了 `IHasValidationErrors` 接口。

日志记录

被捕获的异常会被自动记录到日志中。

日志级别

默认情况下异常被记录为 `Error` 日志级别, 可以通过实现 `IHasLogLevel` 接口指定日志级别。

```
public class MyException : Exception, IHasLogLevel
{
    public LogLevel LogLevel { get; set; } = LogLevel.Warning;
```

```
}
```

异常自定义日志

某些异常类型可能需要记录额外日志信息，可以通过实现 `IExceptionWithSelfLogging` 接口来记录指定日志。

```
public class MyException : Exception, IExceptionWithSelfLogging
{
    public void Log(ILogger logger)
    {
        //...log additional info
    }
}
```

方法 `ILogger.LogException` 用来记录异常日志，在需要时可以使用其它扩展方法。

业务异常

大多数异常都是业务异常，可以实现 `IBusinessException` 接口来标记异常为业务异常。

`BusinessException` 除了实现 `IHasErrorCode`, `IHasErrorDetails`, `IHasLogLevel` 接口外，还实现了 `IBusinessException` 接口，其默认日志级别为 `Warning` 级别。

```
throw new BusinessException( "<code-namespace>:<error-code>" );
```

格式: `<code-namespace>:<error-code>`

异常本地化

错误消息到客户端，有两种模型：用户友好异常和使用错误代码。

用户友好异常

如果异常实现了 `IUserFriendlyException` 接口，那么 ABP 不会修改 `Message` 和 `Details` 属性，而直接将它发送给客户端。

`UserFriendlyException` 类是内建的 `IUserFriendlyException` 接口的实现。

```
throw new UserFriendlyException( "Username should be unique!" );
```

```
throw new
UserFriendlyException( _stringLocalizer[ "UserNameShouldBeUniqueMessage" ] );
```

IUserFriendlyException 接口派生自 IBusinessException 接口。

UserFriendlyException 类派生自 BusinessException 类。

使用错误代码

```
services.Configure<AbpExceptionLocalizationOptions>(options =>
{
    options.MapCodeNamespace("App", typeof(QaResource));
});

{
    "culture": "en",
    "texts": {
        "App:010002": "You can not vote your own answer!"
    }
}
```

使用消息格式化参数

```
throw new BusinessException("App:010046")
{
    Data =
    {
        {"UserName", "john"}
    }
};

throw new BusinessException("App:010046").WithData("UserName", "john");
```

WithData 支持有多个参数的链式调用 (如.WithData(...).WithData(...))。

```
{
    "culture": "en",
    "texts": {
        "App:010046": "Username should be unique. ' {UserName}' is already taken!"
    }
}
```

HTTP 状态代码映射

- 对于 AbpAuthorizationException, 未登录 401 未认证, 已登录 403 未授权。
- 对于 AbpValidationException 返回 400 (错误的请求)。
- 对于 EntityNotFoundException 返回 404 (未找到)。
- 对于 IBusinessException 和 IUserFriendlyException 返回 403 未授权。
- 对于 NotImplementedException 返回 501 (未实现)

- 对于其他异常 (基础架构中未定义的) 返回 500 (服务器内部错误)

DefaultHttpExceptionStatusCodeFinder: IHttpExceptionStatusCodeFinder

```
services.Configure<AbpExceptionHttpStatusCodeOptions>(options =>
{
    options.Map("App:010002", HttpStatusCode.NotFound);
});
```