Fabrício Cabral <fabriciofx@gmail.com>

[dotnetarchitects] Validação de objetos

30 messages

Fernando Mondo <fernando.mondo@gmail.com> Mon, Jun 11, 2012 at 11:34 AM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Dando uma olhada no Blog no lambda3, me deparei com este post antigo
http://blog.lambda3.com.br/2009/09/como-tratar-erros/

é antigo eu sei, mas me deixou intrigado.

O Giovanni diz;

'Quando o usuário digitar um CPF inválido, em vez de lançar uma CPFInvalidoException, você devolve algum objeto que informe que o CPF é inválido'

Como assim? Imagina se eu estou criando um Cliente:

var cliente = new Cliente("joao", 9944558784); //nome e Cpf

Eu não devo lançar uma exception? eu acho que com OOP nossos objetos devem sempre lançar excessão nestes casos, e vocês?

```
public class Cliente
{
  public Cliente (string nome, string cpf)
  {
     if (string.isNullOrEmpty(nome))
        throw new IllegalArgumentException();

      if (string.isNullOrEmpty(cpf))
        throw new IllegalArgumentException();

      if (cpf.IsCpf())
        throw new  CPFInvalidoException();
  }
}
```

agora caso fosse uma validação de Cpf já cadastrado no sistema aí sim deve estar em outro lugar (fora da classe e um pouco antes de salvar no banco).

Se pegarmos como exemplo a classe DateTime, esta linha vai lançar um ArgumentOutOfRangeException

var data = new DateTime(2012, -11, 6);

aposto que tem um if neste contrutor. Ou estou enganado?

o unico problema que tenho con essa abordagem é que para evitar de levar uma Exception na cara eu tenho todas estas validaçãoes duplicadas na minha camada de UI. (com Data Annotations no MVC, ou Custon Validators no WebForms) o que não é nada legal.

E tem o tal do design by contract que é um if throw Exception turbinado, mas confesso que não consegui implementar pois nos mesu Unit tests levo com uma janela na cara me perguntando que quero continuar ou abortar, mas é ignorancia minha.

Então, qual é q melhor abordagem de validação? throw Exception, design by contract, não validar na

classe, devolver um objeto que informe o erro (Como isso?)

Mário Meyrelles <mariomeyrelles@gmail.com> Mon, Jun 11, 2012 at 11:43 AM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Cara, neste caso, pode sim lançar a exception como você está fazendo. É uma boa prática e as pessoas que forem utilizar seu código conseguirão aprender mais rapidamente como passar um CPF corretamente, sem serem surpreendidas com erros tardiamente descobertos.

Só não sei se vale a pena criar uma exception só para isso - eu tentaria usar alguma exception do .NET ou criaria exception mais genérica para carregar erros de negócio da aplicação.

O que talvez o Giovanni quis dizer é que você deve tratar esta exception e mostrar algo amigável para o cliente após suas validações.

2012/6/11 Fernando Mondo <fernando.mondo@gmail.com>
[Quoted text hidden]

[Quoted text hidden]

Thiago Coelho <thiagokoelho@gmail.com> Mon, Jun 11, 2012 at 11:52 AM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Achei interessante, pois se você deixar o construtor criar um novo objeto com dados inválidos poderá te dar problemas lá na frente, eu diria que isso pode ser chamado de "matar o mal pela raiz".

Em 11 de junho de 2012 11:43, Mário Meyrelles <mariomeyrelles@gmail.com> escreveu:

> Cara, neste caso, pode sim lançar a exception como você está fazendo. É uma boa prática e as pessoas que forem utilizar seu código conseguirão aprender mais rapidamente como passar um CPF corretamente, sem serem surpreendidas com erros tardiamente descobertos.
>
> Só não sei se vale a pena criar uma exception só para isso - eu tentaria usar alguma exception do .NET ou criaria exception mais genérica para carregar erros de negócio da aplicação.
>
> O que talvez o Giovanni quis dizer é que você deve tratar esta exception e mostrar algo amigável para o cliente após suas validações.
>
> 2012/6/11 Fernando Mondo <fernando.mondo@gmail.com>
>
> > Dando uma olhada no Blog no lambda3, me deparei com este post antigo
> > http://blog.lambda3.com.br/2009/09/como-tratar-erros/
> >
> > é antigo eu sei, mas me deixou intrigado.
> >
> > O Giovanni diz;
> >
> > 'Quando o usuário digitar um CPF inválido, em vez de lançar uma CPFInvalidoException, você devolve algum objeto que informe que o CPF é inválido'

> > Como assim? Imagina se eu estou criando um Cliente:
> >
> > var cliente = new Cliente("joao", 9944558784); //nome e Cpf
> > [Quoted text hidden]
>
> [Quoted text hidden]

--
Thiago Coelho - Engenheiro de Computação
thiagocoelho.net
twitter.com/thiagokoelho

[Quoted text hidden]

Fernando Mondo <fernando.mondo@gmail.com> Mon, Jun 11, 2012 at 12:16 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

pois é, mas vocês notaram que a maioria dos exemplos na internet não usam essa abordagem.

Se pegarmos qualquer exemplo Mvc + Code First, as classes não tem nem construtor. Está mais para estruturado do que OOP, mas pelo menos evita a repetição de códito, já que a validação pode ser feita em um único lugar.

Estou tentado aplicar o design by contract mas nem no debug funciona (mesmo quando o Contratc falha o debug continua).
[Quoted text hidden]
[Quoted text hidden]

Rafael Ponte <rponte@gmail.com> Mon, Jun 11, 2012 at 12:17 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Oi Fernando,

Melhor criar um tiny type para representar o cpf, algo como:

Cliente cliente = new Cliente("joao", new Cpf("9944558784");

A regra de validação ficaria na classe Cpf, seria mais legível e simples de testar unitariamente. E nela seria onde você validaria e lançaria uma exception em caso de não conformidade.

2012/6/11 Fernando Mondo <fernando.mondo@gmail.com>
> Dando uma olhada no Blog no lambda3, me deparei com este post antigo
> [Quoted text hidden]

--
Rafael Ponte
http://www.triadworks.com.br
[Quoted text hidden]

Fernando Mondo <fernando.mondo@gmail.com> Mon, Jun 11, 2012 at 12:21 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

na verdade eu tenho uma class Documento, só não coloquei para encurtar o exemplo, mas de qualquer

forma, na classe cliente eu tenho um if para lançar exception caso a referencia da classe documento seja Null, e na classe Documento eu tenho o Regex para validar.
[Quoted text hidden]

Mário Meyrelles <mariomeyrelles@gmail.com> Mon, Jun 11, 2012 at 12:23 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Então, quanto ao framework de DbC, dá uma olhada em: http://www.codeproject.com/Articles/1863/Design-by-Contract-Framework
[Quoted text hidden]

Fernando Mondo <fernando.mondo@gmail.com> Mon, Jun 11, 2012 at 12:32 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Este eu ainda não vi, estava tentando com o nativo System.Diagnostics.Contracts. tem até este link http://www.heroisdati.com/design-by-contract-dbc-em-net/ mas não funcinou aqui.

e ainda fico com pé atraz pelo duplicação da validação.
[Quoted text hidden]

Juan Lopes <me@juanlopes.net> Mon, Jun 11, 2012 at 12:44 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Rafael Ponte: +1

2012/6/11 Rafael Ponte <rponte@gmail.com>
> Oi Fernando,
>
> Melhor criar um tiny type para representar o cpf, algo como:
>
> Cliente cliente = new Cliente("joao", new Cpf("9944558784");
>
> A regra de validação ficaria na classe Cpf, seria mais legível e simples de testar unitariamente. E nela seria onde você validaria e lançaria uma exception em caso de não conformidade.
>
> 2012/6/11 Fernando Mondo <fernando.mondo@gmail.com>
> > Dando uma olhada no Blog no lambda3, me deparei com este post antigo
> > http://blog.lambda3.com.br/2009/09/como-tratar-erros/
> >
> > é antigo eu sei, mas me deixou intrigado.
> >
> > O Giovanni diz;
> >
> > 'Quando o usuário digitar um CPF inválido, em vez de lançar uma CPFInvalidoException, você devolve algum objeto que informe que o CPF é inválido'
> >
> > Como assim? Imagina se eu estou criando um Cliente:
> >
> > var cliente = new Cliente("joao", 9944558784); //nome e Cpf
> > [Quoted text hidden]
>
> [Quoted text hidden]
> [Quoted text hidden]

--
https://github.com/juanplopes

[Quoted text hidden]

Mário Meyrelles <mariomeyrelles@gmail.com> Mon, Jun 11, 2012 at 12:49 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Então,

Validação é algo que realmente ocorre em mais de um ponto.

Não tenho opinião formada, mas para mim, a eventual duplicação de validação pode ou não ser necessária. Explico:

No front-end você valida o e-mail. Você precisa validar o formato do e-mail também no back-end? Resposta: depende. Se você possui múltiplos clientes para um mesmo servidor, acho que faz total sentido fazer essa validação de novo. Porém, se você possui controle total do que entra no servidor, não acho que seja necessário validar se o e-mail é válido. Mas você terá outras validações no servidor, de acordo com outras regras, como por exemplo, unicidade de e-mails cadastrados. Na prática, você terá a mesma regra de validação, porém, uma será em javascript e outra em C# no servidor. Não sei se faz sentido ficar se incomodando com duplicidade das validações.

Acho que é isso.
[Quoted text hidden]

Ricardson Albuquerque <greadcadinho@gmail.com> Mon, Jun 11, 2012 at 1:16 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Se eu tiver a validação no domain eu vou estar garantindo a integridade total no uso de qualquer entidade em qualquer negocio, logo acho indispensável validações nesta camada.

Validações de processo devem ficar na application, pois geralmente você precisa validar combinações com entidades que podem estar em módulos distintos ou regras que não interessam especificamente a uma ou outra entidade.

Validações no front-end seria algo adicional para usabilidade.

Colocar validações no construtor da entidade no meu entendimento irá trazer pontos origens diversos (a cada instancia da entidade) para validação, ao invés de no momento de estar passando pelo domain a validação ocorrer de uma só vez.

2012/6/11 Mário Meyrelles <mariomeyrelles@gmail.com>
> Então,
>
> Validação é algo que realmente ocorre em mais de um ponto.
>
> Não tenho opinião formada, mas para mim, a eventual duplicação de validação pode ou não ser necessária. Explico:
>
> No front-end você valida o e-mail. Você precisa validar o formato do e-mail também no back-end? Resposta: depende. Se você possui múltiplos clientes para um mesmo servidor, acho que faz total sentido fazer essa validação de novo. Porém, se você possui controle total do que entra no servidor, não acho que seja necessário validar se o e-mail é válido. Mas você terá outras validações no servidor, de acordo com outras regras, como por exemplo, unicidade de e-mails cadastrados. Na prática, você terá a mesma regra de validação, porém, uma será em javascript e outra em C# no servidor. Não sei se faz sentido ficar se incomodando com duplicidade das validações.
>
> Acho que é isso.
>
> 2012/6/11 Fernando Mondo <fernando.mondo@gmail.com>
>> Este eu ainda não vi, estava tentando com o nativo System.Diagnostics.Contracts. tem até este link http://www.heroisdati.com/design-by-contract-dbc-em-net/ mas não funcinou aqui.

> e ainda fico com pé atraz pelo duplicação da validação.
>
> Em 11 de junho de 2012 12:23, Mário Meyrelles <mariomeyrelles@gmail.com> escreveu:
>
> > Então, quanto ao framework de DbC, dá uma olhada em: http://www.codeproject.com/Articles/1863/Design-by-Contract-Framework
> >
> > 2012/6/11 Fernando Mondo <fernando.mondo@gmail.com>
> > > na verdade eu tenho uma class Documento, só não coloquei para encurtar o exemplo, mas de qualquer forma, na classe cliente eu tenho um if para lançar exception caso a referencia da classe documento seja Null, e na classe Documento eu tenho o Regex para validar.
> > >
> > > Em 11 de junho de 2012 12:17, Rafael Ponte <rponte@gmail.com> escreveu:
> > >
> > > > Oi Fernando,
> > > >
> > > > Melhor criar um tiny type para representar o cpf, algo como:
> > > >
> > > > Cliente cliente = new Cliente("joao", new Cpf("9944558784");
> > > >
> > > > A regra de validação ficaria na classe Cpf, seria mais legível e simples de testar unitariamente. E nela seria onde você validaria e lançaria uma exception em caso de não conformidade.
> > > >
> > > > 2012/6/11 Fernando Mondo <fernando.mondo@gmail.com>
> > > > > Dando uma olhada no Blog no lambda3, me deparei com este post antigo http://blog.lambda3.com.br/2009/09/como-tratar-erros/
> > > > >
> > > > > é antigo eu sei, mas me deixou intrigado.
> > > > >
> > > > > O Giovanni diz;
> > > > >
> > > > > 'Quando o usuário digitar um CPF inválido, em vez de lançar uma CPFInvalidoException, você devolve algum objeto que informe que o CPF é inválido'
> > > > >
> > > > > Como assim? Imagina se eu estou criando um Cliente:
> > > > >
> > > > > var cliente = new Cliente("joao", 9944558784); //nome e Cpf
> > > > > [Quoted text hidden]
> > > > > [Quoted text hidden]
> > > > [Quoted text hidden]
> > > > [Quoted text hidden]
> > > [Quoted text hidden]
> > >
> > > [Quoted text hidden]
> >
> > [Quoted text hidden]
>
> [Quoted text hidden]

--
Ricardson Albuquerque

[Quoted text hidden]

Fabrício Cabral <fabriciofx@gmail.com> Mon, Jun 11, 2012 at 1:32 PM
To: dotnetarchitects@googlegroups.com

Este é um tema bem interessante.

Concordo plenamente que a validação no domain/application TEM (MUST) que

existir. Mas, e no frontend e no banco de dados?

O Ricardson aborda a questão de usabilidade para a validação no frontend. Mas em que ponto isto influencia? E só é relevante para a usabilidade?

No banco de dados é importante se você for compartilhar a sua base de dados com outra aplicação, sem passar pelo seu domain/application. Mas alguém permite isso hoje em dia? Faz sentido ainda manter esse cuidado na base de dados?

[]'s

2012/6/11 Ricardson Albuquerque <greadcadinho@gmail.com>
[Quoted text hidden]

--
--fx

Denis Ferrari <denis.sisinf@gmail.com> Mon, Jun 11, 2012 at 1:33 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Oi Pessoal,

Existe uma discussão antiga no grupo, muito rica em argumentos, seguem os links que achei:
https://groups.google.com/group/dotnetarchitects/browse_thread/thread/ce6c8b26ffa903cc?hl=pt
https://groups.google.com/group/dotnetarchitects/browse_thread/thread/6c95dc17fdd18ad3/480f2ca6d765ab99?show_docid=480f2ca6d765ab99&fwc=1&hl=pt

Validação de domínio mantém a integridade do sistema. Validação de cliente é para ajudar o usuário a completar sua tarefa de forma assistida. Se você tiver um domínio isolado, até o ponto que eu entendo, duplicações de algumas validações podem acontecer.

Como o ponto aqui é a integridade do sistema, você poderia mantê-la das duas formas, mas a única forma garantida é com exceções, pois um CPF com estado inválido poderia não ser verificado em algum ponto do sistema (cpf.IsValid) por esquecimento, por exemplo.

Sobre tipos, se o tipo aparecer no seu negócio, não tem porque usar strings para representá-lo. O Elemar fez um post sobre isso aqui. Mesmo usando tipos "complexos", você poder ensinar o NHibernate como tratá-los.

Acho que é isso.

Abraços!

Denis Ferrari
www.heroisdati.com
[Quoted text hidden]
[Quoted text hidden]

Mário Meyrelles <mariomeyrelles@gmail.com> Mon, Jun 11, 2012 at 1:40 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Sim, concordo. É obrigatório ter validação pelo menos na application, domínio, servidor, seja lá o que for..

Acho que é preciso clarificar: é interessante sim assegurar o funcionamento do contrato usando, se necessário, exceptions no construtor. Assegurar o contrato é muito mais do que apenas validação. Se eu inicializar um tipo de forma errada eu preciso aprender o mais cedo possível o que está errado. Isso é usabilidade do seu framework. Isso é facilitar a vida do dev. Nem todos os contratos têm a ver com validação.

Se você estiver desenvolvendo uma aplicação web normal, não tem como deixar de falar em validação no

client, usando javascript. E na prática, isso significa que você terá que assegurar que o usuário não faça coisas erradas no sistema - validar e duplicar o código sim, infelizmente. Odeio este tipo de trabalho, mas é o onde se gasta mais tempo e se gasta mais esforço braçal rs.

2012/6/11 Ricardson Albuquerque <greadcadinho@gmail.com>

> Se eu tiver a validação no domain eu vou estar garantindo a integridade total no uso de qualquer entidade em qualquer negocio, logo acho indispensável validações nesta camada.
> [Quoted text hidden]

[Quoted text hidden]

Mário Meyrelles <mariomeyrelles@gmail.com> Mon, Jun 11, 2012 at 1:52 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Sinceramente Fabrício, no banco ultimamente eu não faço validação. Esta necessidade não tem aparecido nos meus sistemas. Tive sempre a sorte de ter a certeza de que os dados chegando apenas pelo servidor de aplicação. E, validando tudo antes de salvar no banco já me permite ter a certeza de que os dados estão corretos também a trazê-los de volta para a aplicação. Mas isso é uma peculiaridade da minha realidade - pode ser que faça total sentido ter mais e mais validações em outros tipos de sistema.

2012/6/11 Fabrício Cabral <fabriciofx@gmail.com>
[Quoted text hidden]

[Quoted text hidden]

Fernando Mondo <fernando.mondo@gmail.com> Mon, Jun 11, 2012 at 2:11 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

percebi que o Giovanni mudou de opinião já que ele fez este comentário no site do Elemar:

"Gosto da ideia também. Mas não gosto nada do IsValid.
Pra mim, se o CPF for inválido, logo no construtor eu lançaria um InvalidCPFException. A ideia é que se tenho um CPF, ele é válido. E para o NullObject eu usaria 000.000.000-00, que, pelo que me lembro, é válido também.
Prefiro DbC no lugar de programação defensiva."
[Quoted text hidden]
[Quoted text hidden]

---

edmilson hora <edmilson_hora@yahoo.com.br> Mon, Jun 11, 2012 at 4:44 PM
Reply-To: dotnetarchitects@googlegroups.com
To: "dotnetarchitects@googlegroups.com" <dotnetarchitects@googlegroups.com>

Rafael, acho que vc matou a questão, eu prefiro trabalhar desta forma, para os casos de CPF/CNPJ assim em qualquer lugar que eles aparecererem estarão automaticamente válidos. Lançando excessão no caso de invalido é claro.

De: Rafael Ponte <rponte@gmail.com>
Para: dotnetarchitects@googlegroups.com
Enviadas: Segunda-feira, 11 de Junho de 2012 12:17
Assunto: Re: [dotnetarchitects] Validação de objetos
[Quoted text hidden]
[Quoted text hidden]

Andre Nobre <andrenobre25@gmail.com> Mon, Jun 11, 2012 at 1:10 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Fernando,
eu também, há muito tempo, escrevi sobre isto. E compartilho da opinião do Giovanni: exceções são exceções :)

O que eu quero dizer é que exceções devem ser utilizadas em situações onde não consigamos prever um determinado comportamento, e não para tratar validações. Existem diversas formas disso ser feito, e uma delas é a maneira proposta pelo Giovanni. Você pode retornar uma estrutura de dados que identifique o problema ou você pode criar algum acesso para que os clientes de seu código (outros códigos) possam verificar se o CPF está válido ou não.

Como alguém citou, caso o cliente tente executar o código e o CPF esteja inválido, aí sim, TALVEZ, o throw se faça necessário. Caso contrário, em todas as situações que você consiga prever e ter o controle sobre, opte por não utilizar o lançamento de exceções. Veja mais no link abaixo:

http://andrenobre.wordpress.com/2007/04/13/diga-no-ao-throw/

Abraços.
[Quoted text hidden]
[Quoted text hidden]

Andre Nobre <andrenobre25@gmail.com> Mon, Jun 11, 2012 at 8:18 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Eu acho que existem diversas formas de se trabalhar para atingir o mesmo objetivo, mas eu não consigo entender a criação de um objeto como um Exception para falar para o cliente que o CPF está inválido. Desta maneira vocês retornam em um primeiro momento pro cliente uma Exception com propriedades como Data, Message, HelpLink, HResult, StackTrace, TargetSite, tudo isso para uma informação tão simples.

Abraços.
[Quoted text hidden]
[Quoted text hidden]

Andre Nobre <andrenobre25@gmail.com> Mon, Jun 11, 2012 at 8:26 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Complementando, eu concordo que existe uma hora onde não há saída. Não posso depender do cliente chamar um método de validação para verificar se o CPF está válido ou não, isto é impossível. Mas gosto de pensar que o cliente deve ter uma oportunidade ou uma maneira mais simples e direta de saber se há algo errado ou não nas informações fornecidas.

Abraços.
[Quoted text hidden]
[Quoted text hidden]

Fernando Mondo <fernando.mondo@gmail.com> Tue, Jun 12, 2012 at 1:06 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Imagine a seguinte situação:
Em um formulário o usuario deve digitar uma data em um TextBox, ao clicar em Concluir nós pegamos os dados e atribuimos para um objeto do tipo data.

var data = new DateTime(int.parse(txtAno.Text), int.parse(txtMes.Text), int.parse(txtDia.Text));

Ninguém faria assim né? Porque se não levaria com uma Exception caso algum dos dados estiver incorreto (um número negativo talvez).

Eu preciso validar os dados dos TextBox antes de instanciar o DateTime.

Do mesmo modo se eu colocar uns Ifs Throw nas minhas classes de Dominio (Cliente, Cpf, etc), quem estiver programando a UI vai validar os dados antes de instanciar-las para evitar as Exceptions
[Quoted text hidden]

Juan Lopes <me@juanlopes.net> Tue, Jun 12, 2012 at 1:53 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Ninguém faria assim?

O único objeto capaz de verificar se uma data é válida é a própria data. Ou você vai replicar a lógica de validação de data que já existe no componente? Quer ver uns exemplos interessantes?

A hora 20:59:60 BRT será válida no próximo dia 30 (30 de junho de 2012). Isso porque segundo o UTC, vira e mexe é preciso adicionar um "leap second", que é um segundo forjado para manter o que convencionamos como "dia" o mais parecido possível com o dia solar.

Dia 29 de fevereiro só existe em anos bissextos. Que acontecem de 4 em 4 anos, exceto nos anos divisíveis por 100, a menos que sejam divisíveis por 400

Os dias de 5 a 14 de outubro de 1582 não existem no calendário gregoriano.

Você realmente quer escrever essa lógica na sua aplicação?

2012/6/12 Fernando Mondo <fernando.mondo@gmail.com>

> Imagine a seguinte situação:
> Em um formulário o usuario deve digitar uma data em um TextBox, ao clicar em Concluir nós pegamos os dados e atribuimos para um objeto do tipo data.
>
> var data = new DateTime(int.parse(txtAno.Text), int.parse(txtMes.Text), int.parse(txtDia.Text));
>
> Ninguém faria assim né? Porque se não levaria com uma Exception caso algum dos dados estiver incorreto (um número negativo talvez).
>
> Eu preciso validar os dados dos TextBox antes de instanciar o DateTime.
>
> Do mesmo modo se eu colocar uns Ifs Throw nas minhas classes de Dominio (Cliente, Cpf, etc), quem estiver programando a UI vai validar os dados antes de instanciar-las para evitar as Exceptions
>
> Em 11 de junho de 2012 20:26, Andre Nobre <andrenobre25@gmail.com> escreveu:
>
> > Complementando, eu concordo que existe uma hora onde não há saída. Não posso depender do cliente chamar um método de validação para verificar se o CPF está válido ou não, isto é impossível. Mas gosto de pensar que o cliente deve ter uma oportunidade ou uma maneira mais simples e direta de saber se há algo errado ou não nas informações fornecidas.
> >
> > Abraços.
> >
> > Em 11 de junho de 2012 20:18, Andre Nobre <andrenobre25@gmail.com> escreveu:
> >
> > > Eu acho que existem diversas formas de se trabalhar para atingir o mesmo objetivo, mas eu não consigo entender a criação de um objeto como um Exception para falar para o cliente que o CPF está inválido.
> > > Desta maneira vocês retornam em um primeiro momento pro cliente uma Exception com propriedades como Data, Message, HelpLink, HResult, StackTrace, TargetSite, tudo isso para uma informação tão simples.
> > >
> > > Abraços.
> > >
> > > Em 11 de junho de 2012 16:44, edmilson hora <edmilson_hora@yahoo.com.br> escreveu:

> > Rafael, acho que vc matou a questão, eu prefiro trabalhar desta forma, para os casos de CPF/CNPJ assim em qualquer lugar que eles aparecererem estarão automaticamente válidos. Lançando excessão no caso de invalido é claro.
> >
> > ---
> >
> > **De:** Rafael Ponte <rponte@gmail.com>
> > **Para:** dotnetarchitects@googlegroups.com
> > **Enviadas:** Segunda-feira, 11 de Junho de 2012 12:17
> > **Assunto:** Re: [dotnetarchitects] Validação de objetos
> >
> > Oi Fernando,
> >
> > Melhor criar um tiny type para representar o cpf, algo como:
> >
> > Cliente cliente = new Cliente("joao", new Cpf("9944558784");
> >
> > A regra de validação ficaria na classe Cpf, seria mais legível e simples de testar unitariamente. E nela seria onde você validaria e lançaria uma exception em caso de não conformidade.
> >
> > 2012/6/11 Fernando Mondo <fernando.mondo@gmail.com>
> > > Dando uma olhada no Blog no lambda3, me deparei com este post antigo http://blog.lambda3.com.br/2009/09/como-tratar-erros/
> > >
> > > é antigo eu sei, mas me deixou intrigado.
> > >
> > > O Giovanni diz;
> > >
> > > 'Quando o usuário digitar um CPF inválido, em vez de lançar uma CPFInvalidoException, você devolve algum objeto que informe que o CPF é inválido'
> > >
> > > Como assim? Imagina se eu estou criando um Cliente:
> > >
> > > var cliente = new Cliente("joao", 9944558784); //nome e Cpf
> > [Quoted text hidden]
> [Quoted text hidden]
> [Quoted text hidden]
>
> [Quoted text hidden]
>
> [Quoted text hidden]

[Quoted text hidden]

--
https://github.com/juanplopes
[Quoted text hidden]

Winston Pacheco Junior <winston.pacheco@gmail.com> Tue, Jun 12, 2012 at 2:32 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

+1 pra Juan... A data sabe o que é uma data válida...
[Quoted text hidden]

Fernando Mondo <fernando.mondo@gmail.com> Tue, Jun 12, 2012 at 3:20 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

peraí, como faria então? colocava um try cath e mostrava a exception pro usuario?

o que eu estou dizendo desde o inicio é que tem if throw nas classes "bem modeladas" e que não deve ser diferente nas classes do meu dominio.

é logico que não vou tratar todas as regras no UI mas devo garantir que seja passado pele menos números positivos e não nulos.

uma outra abordagem, que é a defendida pelo Giovanni Bassi no link do meu primeiro post onde a validação esta à parte. Neste contexto, a classe data seria uma simples extrutura de dados que aceitasse qualuqe valor e depois poderiamos chamar um data.IsValid() para validá-la.

Uma coisa para refletir é como que o Asp.Net Mvc faz o Bind de um input para uma data. Ele retorna null caso não esteja em formato válido. será que ele usa um Try cath para tentar converter a string do request em uma data?
[Quoted text hidden]

Juan Lopes <me@juanlopes.net> Tue, Jun 12, 2012 at 3:37 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Sou contra IsValid. Por dois motivos:

Objetos não devem existir em estado inválido.

Pedir perdão é sempre melhor que pedir permissão.
[Quoted text hidden]

Andre Nobre <andrenobre25@gmail.com> Tue, Jun 12, 2012 at 4:17 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Mas no caso da data, o que eu recomendo fazer é validar as situações em que vc TEM CERTEZA que irão gerar exceptions. Isto é evitar, de forma pró-ativa, o lançcamento desnecessário de exceções. Se você sabe que não pode usar número negativo, porque permitir que o objeto Data retorna uma exception? Acho mais adequado validar, no contexto correto, e retornar ao usuário ao invés de esperar o objeto lançar a exceção.

O que não se aplica nos casos onde exija uma análise maior, como no exemplo do Juan Lopes. Isto sim eu concordo que devemos deixar o objeto se virar. Mas nos exemplos como o acima, não.

Abraços.
[Quoted text hidden]
[Quoted text hidden]

Andre Nobre <andrenobre25@gmail.com> Tue, Jun 12, 2012 at 4:19 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Alias, ótima discussão e ótimos links.
[Quoted text hidden]

Alexandre Santos Costa <alexandresantoscosta@gmail.com> Tue, Jun 12, 2012 at 10:52 PM
Reply-To: dotnetarchitects@googlegroups.com

To: dotnetarchitects@googlegroups.com

Pessoal,

Dando minha opinião e lembrando da palestra do Elemar no DNAD eu diria que: depende.

Á validação no lado cliente não melhora só a usabilidade mas impede que requisições "desnecessárias" sejam feitas ao servidor, economizando recursos como rede e CPU do servidor. Se pensarmos em cenários com aplicações móveis onde nem sempre a velocidade da internet é boa ou usuários em localizações remotas como o a região norte isso faz todo sentido.

A validação na camada de aplicação nos permite a execução de workflows que se utilizam de mais de uma Entidade, muito útil em processamentos batch por exemplo.

A validação do domínio nos garante que nossas Entidades estejam sempre em um estado válido.

Acredito que a integridade referencial nos bancos de dados seja sufuciente para nos fornecer dados de qualidade, mas quando existem regras mais complexas e a base é compartilhada por sistemas legados que não podem ser alterados, acho que o uso consciente de algum código essa camada é bem vindo.

Todos os casos devem ser bem estudados, mas a necessidade vai surgindo conforme se conhece melhor o problema.

Atenciosamente,

Alexandre

De: dotnetarchitects@googlegroups.com [mailto:dotnetarchitects@googlegroups.com] **Em nome de** Andre Nobre
Enviada em: terça-feira, 12 de junho de 2012 16:20
Para: dotnetarchitects@googlegroups.com

[Quoted text hidden]

[Quoted text hidden]

[Quoted text hidden]

[Quoted text hidden]

Juan Lopes <me@juanlopes.net> Tue, Jun 12, 2012 at 10:57 PM

Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Claro que depende. Estou aqui defendendo o uso de exceptions para border cases, mas escrevo um sistema de alta performance e na maior parte do tempo evito exceptions para não penalizar a performance.

O ponto é que aqui estamos falando sobre design de código. E assim como não usar exceptions por performance é uma otimização às vezes precipitada, dizer que toda validação precisa ser feita à priori porque no browser é assim também é exagero.

2012/6/12 Alexandre Santos Costa <alexandresantoscosta@gmail.com>
[Quoted text hidden]
[Quoted text hidden]
[Quoted text hidden]

Alexandre Santos Costa <alexandresantoscosta@gmail.com> Fri, Jun 15, 2012 at 9:00 PM
Reply-To: dotnetarchitects@googlegroups.com
To: dotnetarchitects@googlegroups.com

Juan,

Reforço que o quis dizer é que depende.

Se estamos codificando para uma intranet onde o custo de uma requisição ao servidor é quase 0 do ponto de vista de desempenho da rede não vejo problema algum em as validações estarem apenas no server side.

Da mesma forma que validações no banco de dados eu colocaria apenas se o mesmo fosse utilizado por mais de uma aplicação que não compartilhassem de uma camada de serviços para realizar este acesso.

Onde trabalho temos diversos cenários, a maioria muito mal implementada, e constantemente processamentos batch noturnos simplesmente capotam pois a informação esperada não está no formato “corretoo” graças a falta de validações.

Uma vez em uma reunião de incidentes estava s discutindo um sistema que o usuário ao clicar em salvar recebia a famosa página de erro do ASP.NET.

Quando questionada a analista disse que a culpa era do usário que não havia selecionado nenhum item no grid antes de salvar e que o sistema não fazia a validação pois eles já haviam falado para o usuário não fazer isso.

Quanto ao uso de Exceptions sempre as usei nas linguagens que ofereciam este recurso e são muito interessantes pra reportar o que realmente ocorreu quando da execução de determinada operação.

SE não fossem não teríamos a FileNotFoundException, a ArgumentNullException entre outras que existem para nos indicar exatamente o que ocorreu e tomar as devidas ações.

E como você mesmo citou quando se trata de desempenho extremo como são as aplicaç~eos que você trabalha o design da aplicação já deve ser concebido para utilizar ao máximos os recursos e fugir das exceções, oque é diferente quando temos uma interface com usuários criativos que adoram inventar jeitos novos de encontrar bugs velhos.

Grato,

Alexandre

De: dotnetarchitects@googlegroups.com [mailto:dotnetarchitects@googlegroups.com] **Em nome de** Juan Lopes
Enviada em: terça-feira, 12 de junho de 2012 22:57

[Quoted text hidden]

[Quoted text hidden]
[Quoted text hidden]