Domain Annotations

Erik Doernenburg

ThoughtWorks®

bleeding-edge .net

GREAT INDIAN DEVELOPER SUMMIT

Metadata in Code

Languages have used several ad hoc mechanisms to store metadata

• keywords	`Override`
• formatted comments	`/// <summary>`
• naming conventions	`testFooBar()`
• marker interfaces	`Serializable`

.NET Attributes

Attributes are an extensible facility

- language features (keywords) are not

Attributes are available at runtime

- metadata in comments is not

Attributes are first class language elements

- supported by IDE, compiler, and reflection

Attributes can have program code

`[Test]`
`[Browsable(true)]`
	markers for frameworks/tools

`[SoapDocumentMethod]`
`[DisplayName("Address")`
	configuration/mapping

`[AssemblyTitle(…)]`	developer/user information

`[Audit]`
`[RequiresTransaction(…)]`
	point-cut markers

`[MaxValue(1000)]`
`[MaxLength(50)]`
	parameterized markers

`@Override`
`[Layer("Domain")]`
	structural rules

`[PartyPlaceThing]`	domain specific annotation

Domain Annotations

Added to domain objects

- classes, methods, even arguments

Concerned with domain concepts

- type defined in domain namespace

Can have multiple uses

- might even be created before it is used

What you'll see

Leroy's Lorries

- a stock management / logistics application
- based on a real application we worked on

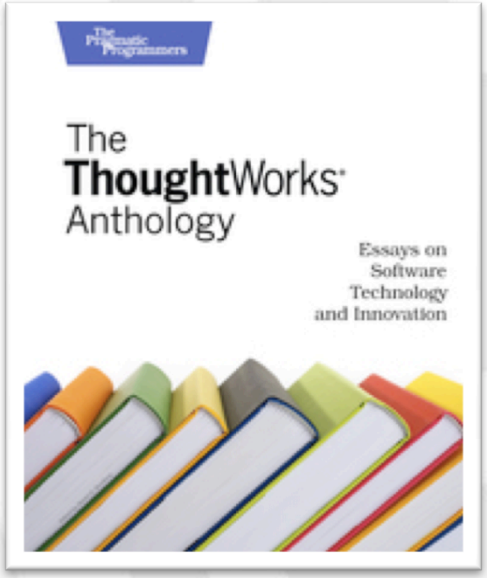
What you’ll see

Attributes with polymorphic implementations

- validation – length and unique constraints

Domain attributes

- data classification & navigational hints
- attributes are used across auditor, permission checker, and data loader services



The Pragmatic Programmers

The ThoughtWorks Anthology

Essays on Software Technology and Innovation

Thank you!

erik.doernenburg.com
www.thoughtworks.com

Attributes are code

Should describe something intrinsic

- generally change when code changes
- should not change for

Should only contain default configuration

- overrideable by external mechanism
- **but**: consider convention for default configuration!

Don’t go crazy!

Should only be used where they make sense…

- `[TestFixture]` is redundant
- `[SetUp]` has better alternatives

Should not have side effects

- if they contain code, this should be stateless (function-like) code

Assumptions…

Should be used cautiously

- `[WebMethod]` does not make a method a service

Never assume presence of an annotation

- objects may be decorated