Chocolatey

KANSAS CITY DEVELOPER CONFERENCE
KCDC

PAIGE
TECHNOLOGIES
INTELLIGENT PAIRING. PERPETUAL SUCCESS.

KSG

AIC
Talent Solutions

KFORCE®

stackify

CENTRIQ
TRAINING

TEKsystems
Our people make IT possible.

epiq

ALIEN ARC
TECHNOLOGIES

VinSolutions
Make every connection count.

Announcements

Kansas City .net USER GROUP KCDNUG.ORG

We have no announcements ☹

Do you?

Group Slack (kcdnug.slack.com)

signup: https://kcdnugslack.herokuapp.com
Token: notabot

@kcdnug

Tonight's Sponsor

EPIQ
http://epiqglobal.com – @EpiqGlobal

Ryan Kelley

rykelley@epiqglobal.com

See Sharper

A Guided Tour Through Microsoft Docs

John Alexander

What is a software design pattern?

A **software design pattern** is a general, reusable solution to a commonly occurring problem.

-Wikipedia

What is a messaging pattern?

In software architecture, a messaging pattern is a network-oriented architectural pattern which describes how two different parts of a message passing system connect and communicate with each other.

-Wikipedia

What are some messaging patterns?

- **Message Type Patterns**
  - Command Message
  - Document Message
  - Event Message
  - Request-Reply Message
- **Messaging Channel Patterns**
  - Point-to-Point Channel
  - Publish-Subscribe Channel
  - Datatype Channel
  - Dead Letter Channel
  - Guaranteed Delivery
  - Message Bus
- **Message Routing Patterns**
  - Pipes and Filter
  - Content-Based Router
  - Content Aggregator
- **Service Consumer Patterns**
  - Transactional Client
  - Polling Consumer
  - Event-Driven Consumer
  - Durable Subscriber
  - Idempotent Receiver
  - Service Factory
  - Message Facade Pattern

Contracts and Information Hiding

  - Contract Pattern
  - Correlation Identifier
  - Message Sequence
  - Message Expiration
- **Message Transformation**
  - Envelope Wrapper
  - Content Enricher
  - Content Filter
  - Claim Check

Why would we want our applications to use messaging for communicating?

- Loosely coupled components
- Communicating across components, processes, or machines
- Flexible configuration
- Ability to scale individual components
- Message routing is a solved problem

How can we send messages in our apps?

- Use built in event model
- Build custom pub/sub interfaces
- Use queues
- Message Bus

Message Types

- A **command** is a message that can be sent from one or more senders and is processed by a single receiver. It is normally used to invoke a procedure in the receiving process.
- An **event** is a message that is published from a single sender, and is processed by (potentially) many receivers. It is normally sent in response to something that has happened.
- A **document** is a single unit of information. The important part of a document message is its content

Publish/Subscribe Messaging

```
var myPublisher = new Publisher();
var mySubscriber = new Subscriber();
myPublisher.Subscribe(mySubscriber);
…
Stuff happens…
…
myPublisher.Publish(message);
mySubscriber receives published message
```

Let’s look at some messaging examples…

Next, let's look at a "real" app…

Purpose
This application is used to processes concert ticket orders

Application Steps
1. Create Order
2. Reserve Tickets
3. Calculate Fees
4. Calculate Taxes
5. Charge Credit Card
6. Store Order

Refactor time…

Goals

- Use messages between components so they can be moved to separate processes/applications as needed.
- Make the process order easily configurable
- Create a way to multi-thread any process for performance tuning

Thank You

Brent Stewart

Alien Arc Technologies, LLC

Follow me on twitter @brentestewart

Code is available on my github: github.com/brentestewart

Wrap Up

Next Meeting: Aug 28th

John Baluka

Understanding the real value of Open Source with nopCommerce

Group Slack (kcdnug.slack.com)

signup: https://kcdnugslack.herokuapp.com
Token: notabot

Networking at “The Bar”

6101 Johnson Dr
Mission, KS 66202

Messaging Patterns

Brent Stewart

@BrentEStewart