Distributed Application Design Patterns, Part 2

Scott Seely
http://www.pluralsight.com/

Outline

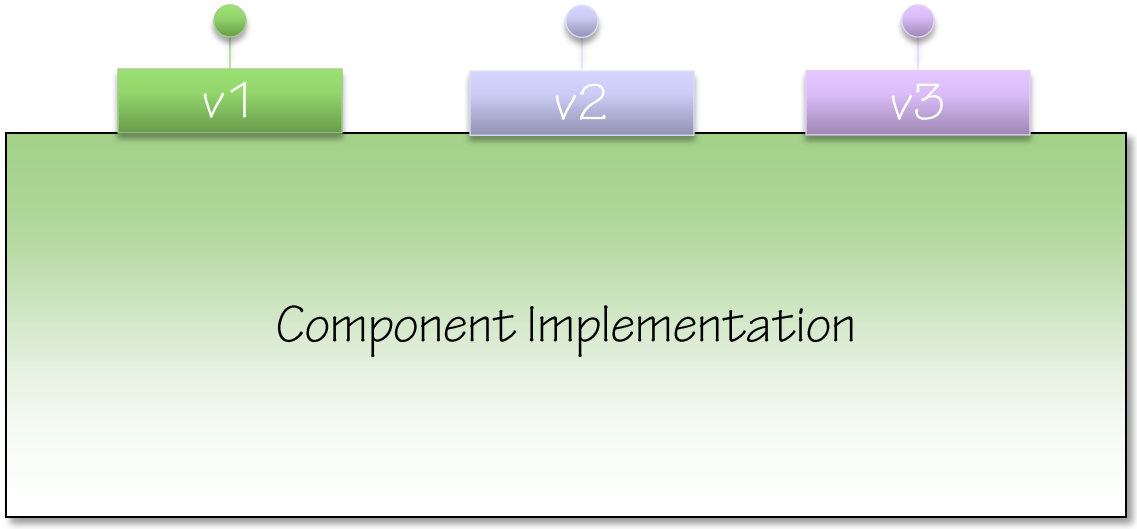
- Versioning Patterns
- Discovery Mechanisms
- Subscription Based Services
- Exception Handling Patterns

Versioning Patterns

- Façade
- Addressing
- Data

Façade

- **Message layer tied to version of service**
- **Message layer abstracts changes**
  - Translate from message version to latest code



Addressing

- **New interface = new address**
- **Numbering pattern:**
  - http://www.pluralsight.com/[service name]/[major].[minor].[build]
  - Each number increases monotonically: 1, 2, 3, 4, …, 9, 10, 11
- **Date Pattern**
  - http://www.pluralsight.com/[service name]/[yyyy]/[mm]/[dd]
  - http://www.pluralsight.com/[service name]/[yyyy].[mm].[dd]

Data Versioning Patterns

- Namespace aware XML
- Plain old data objects

Namespace Aware XML

<a:Address xmlns:a="http://www.usps.com" />

<a:Address xmlns:a="http://www.w3.org/2005/08/addressing"/>

- Leave “open space” at end of XSD (<xs:any minOccurs="0" maxOccurs="unbounded" />)
- Add new members: use open space, leaving extensibility for v.Next
- Remove/Rename members: new XML Namespace, brand new XSD

Example Schema

```
<?xml version="1.0"?>
<xs:schema
  xmlns:tns="http://www.pluralsight.com/VersionedXmlRoot/1.23"
  elementFormDefault="qualified"
  targetNamespace="http://www.pluralsight.com/VersionedXmlRoot/1.23"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="VersionedXmlRoot" nillable="true" type="tns:VersionedXmlRoot" />
  <xs:complexType name="VersionedXmlRoot">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="SomeNumber" type="xs:int" />
      <xs:element minOccurs="1" maxOccurs="1" name="SomeTime" type="xs:dateTime" />
      <xs:any minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="SomeString" type="xs:string" />
    <xs:anyAttribute />
  </xs:complexType>
</xs:schema>
```

Plain Old Data Objects

- **Add new members**
  - Add at the end of the representation.
  - Allow members to be missing, have good “default” behavior
- **Remove members:**
  - Ignore when present
  - New address if member now indicates failure situation
- **Rename members:**
  - Honor old and new name
  - Prefer new address and translate at data receive point

Examples

XML:

```
<Person>
    <FirstName>Scott</FirstName>
    <LastName>Seely</LastName>
<Person>
```

JSON:

```
{ "FirstName" : "Scott",
  "LastName" : "Seely"
}
```

Discovery Mechanisms

- **Human Mechanisms**
  - Search (Google/Bing)
  - Partnership
- **Automation Mechanisms**
  - HTML Meta-tags
  - Directories
  - Broadcast

Discovery via Search

- **Use**
  - Internal Web Services
  - Public Web Services
  - Commercial Web Services
- **Types of service**
  - REST
  - SOAP
  - HTML
  - E-mail

Discovery via Partnership

Scenario:

- Someone (sales, marketing, engineering) creates partnership
- Partnership requires working together
- Systems integration is part of this
- Integration points “discovered” during requirements, user stories, etc.

HTML Meta-tags

- **ATOM and RSS Feeds (blogs!)**

```
<link rel="alternate" type="application/rss+xml"
title="Scott Seely's Blog"
href="http://feedproxy.google.com/ScottSeelysBlog" />

<link rel="alternate" type="application/atom+xml"
title="Scott Seely (Atom 1.0)"
href="http://devlicio.us/blogs//atom.aspx" />
```

- **OpenID pointer:**

```
<link rel="openid2.provider" type="application/x-
openid-kvf" href="http://api.myspace.com/openid" />
```

Discovery via Directories

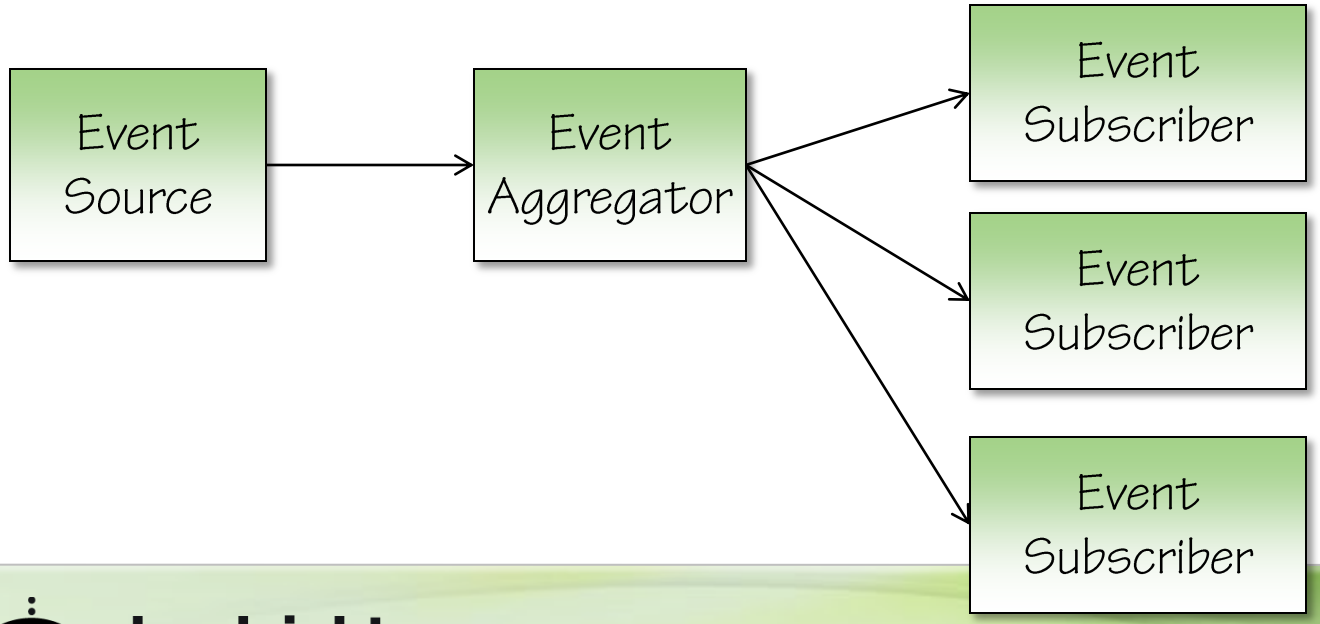
- **UDDI: Universal Description, Discovery, and Integration**
  - Publish/Find/Bind
- **Found "in house" for SOA stacks**
- **Supported by Microsoft, SAP, IBM, and Apache (jUDDI)**
- **Allows for location independence**
- **Usage Pattern:**
  - Developer searches for service using UDDI browser
  - Developer writes code that uses UDDI Web service to discover deployment of found service
  - Code talks to service via address stored in UDDI registry

Discovery via Broadcast

- Service chooses to listen for set of service types
- Client states type of service it wants
- Services that hear broadcast and implement service respond with service address
- Classic implementation: broadcast over UDP.
- WS-Discovery popular implementation of protocol.
- PeerNet Resolution Protocol another discovery mechanism

Subscription Based Services

- Event producers publishes list of topics
- Notification service aggregates list
- Subscribers register interest in topics
- Producer notifies notification service on event
- Notification service fans out topic, manages subscriptions



Exception Handling Patterns

- Client receives server error
- Client error
- Connectivity errors

Client Receives Server Error

- **Server unavailable**
  - Server: Include info on when service will be ready.
  - Client: Read info and do not try again until later
- **Server hiccup**
  - Server: Include info on how error was recorded
  - Client: Record error message. Try again, no more than 3x
- **Server closed connection**
  - Server: Close connection when things look wrong
  - Client:
    1. Record error.
    2. Retry no more than 3x.
    3. After 3x, email local technical support.

Client Error

- **Authentication/Authorization**
  - Server: Tell client authentication/authorization failed.
  - Client: Log error. Stop talking to server until auth is updated.
- **Invalid Request**
- **Throttle**
  - Server: Tell client throttle limit exceeded (# of requests, time of day, etc.) Tell client when throttle reopens (timespan if possible)
  - Client: Wait until timespan elapses

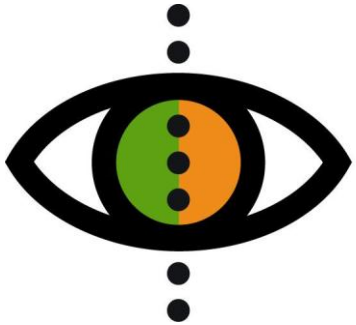
Connectivity Error

- **Endpoint not found/connection refused**
  - Server: Deny connection
  - Client: Log error. Stop talking to server until endpoint updated.
- **Timeout error**
  - Server: Close the connection
  - Client: Re-open the connection, send message again

Summary

- Versioning Patterns
- Discovery Mechanisms
- Subscription Based Services
- Exception Handling Patterns

For more in-depth online developer training visit



pluralsight

see what you can learn

on-demand content from authors you trust