How others do it with CA Plex (SOA, Business Objects, SOAP)

George Jeffcock

Kilanne Services

Friday, June 3, 2011

ca technologies

Speaker

George Jeffcock, CA Plex Consultant

- Working with CA Plex since 1996
- Based in Stockholm, Sweden
- Currently at www.agria.se
  - Leading specialist in the provision of pet insurance for domestic pets, horses and agriculture in Sweden, Norway, Denmark and UK
  - Agria Philosophy: Providing security and peace of mind for animal lovers and their pets and creating a healthier society for pets and their owners.
  - CA 2E – CA Plex (OBASE, PATTERNS, ATOL, Websydian Web Developer & TransacXML)
- Author of Stella Tools
  - Tools to aid CA Plex development

Agenda

Today I will talk about my CA Plex experiences at the Agria Group with the backing of management but please note I do not speak on behalf of the Agria Group.

- A practical session
- Outline how CA Plex and Websydian TransacXML were used to implement projects with greater emphasis on:
  - Service Orientated Architecture
  - Delivery of Business Processes as Webservices
- Aim for Attendees to have one of two results
  - Yep I can borrow/steal/use/adapt/copy that for my site 
  - Smug as we did it better ourselves 

Both positive results from the last session hour at Chicago 2011

Plan

Back Office Tier	Business Services Tier	Webservices (Published) Tier
– Brief outline of Legacy Modernization – CA Plex Tutorial example used as Back Office System	– Requirements overview – Pattern overview – Demo	– Pattern overview – Demo
5 Minutes	25 Minutes	15 Minutes

Back Office tier

Back Office Systems

- Business critical
- Embedded business knowledge
- High maintenance cost.
- Complex structure.
- Obsolete support software.
- Obsolete hardware.
- Lack of technical expertise
- Backlog of change requests.
- Poorly documented.
- Poorly understood.

The Usual Suspects

Period of Legacy Modernization

- No Big Bang rewrite
- Leverage of existing investment
- Brownfield architecture.
- Process of application retirement.
- Service-enabled access to existing business logic and data.
- No Silver Bullet – No package

ca technologies

Business Service tier requirements

General Requirements

- Achieve the following when data can exist in many files on many databases and on many platforms.
  - Fetch a data set
  - Fetch an array of data
  - Update (Add, Delete, Change) data
  - Update an array of data
  - Validate a data set

- Provide stable & standardized interfaces.
  "Once you have seen one Fetch/Update you have seen them all"

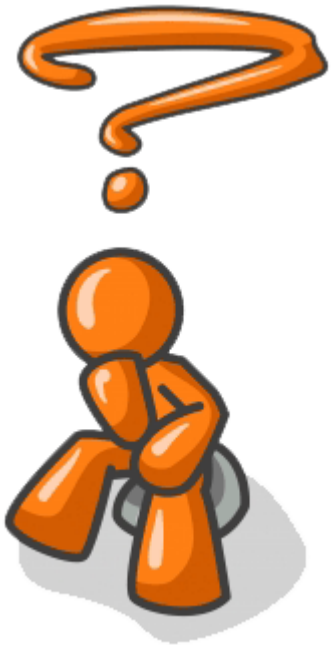
- Externalize business logic

- No direct data access (portability)

- Provide standardized message output

- Provide a method to test in isolation with out knowledge of calling tiers

- Increase Error Handling



Business Service tier patterns

Entity Patterns

- Implementation via Is A triples, Replacement, and action diagram code limited to business logic
- Must work in Combination – Add further functionality in the future and no re-code. *Not good enough just to work on initial release*

Skeleton Container Structures

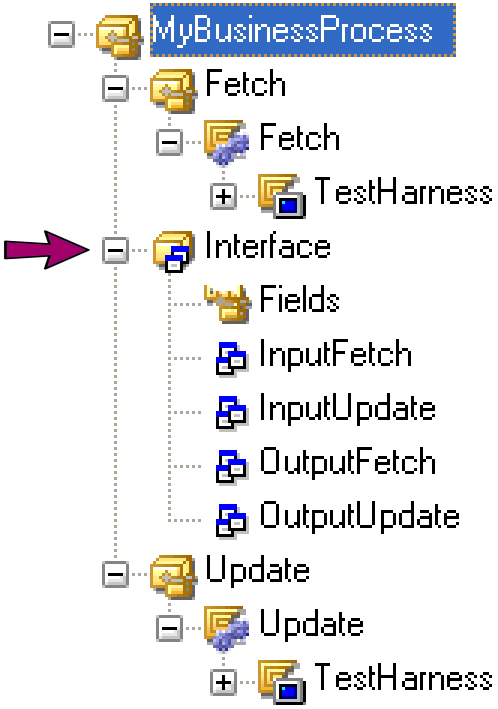
- Common Fetch, Update, Interface containers. In order to provide uniformity across models and ability to create complex abstracts in a simple manner while allowing for replacement Plex methodology

Icons

- Visual Distinction
- Help Developer Training
- Model Clarity

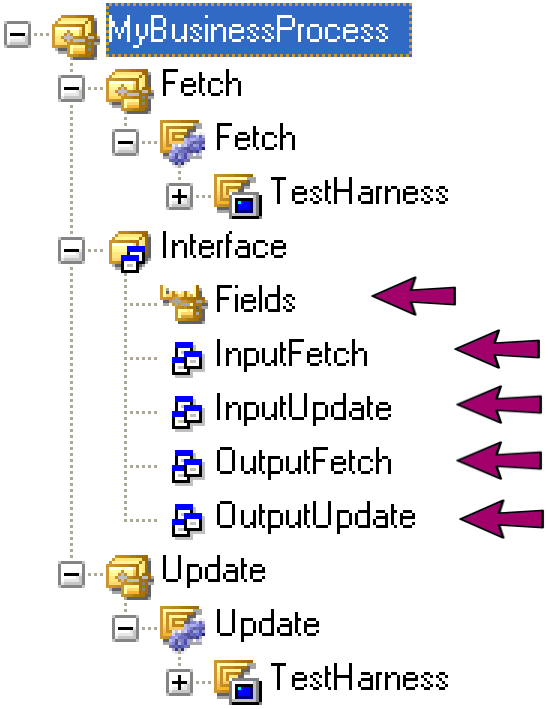
Business Service tier patterns

- Interface Entity
  - Scopes the business service interfaces
  - Allows abstract development
  - Quick development as fields are added as Has Relationships are then available to all interfaces
  - Quick maintenance, changes need only be done in one place.
  - Helps impact analysis
  - Helps project development as the model clearly breaks down the need for Interface development as well business logic development



Business Service tier patterns

- Scoped Function Field Container
  - Unique Fields to Service
- Field Lists
  - Keeps Triples
  - Non implemented Views
  - Collection of Fields from many Tables/Sources
  - Interface Contract
  - Only what is necessary
  - Flags to absolute minimum (ModeFlag)
  - Used across models, fncs, local variables, impact analysis, change management



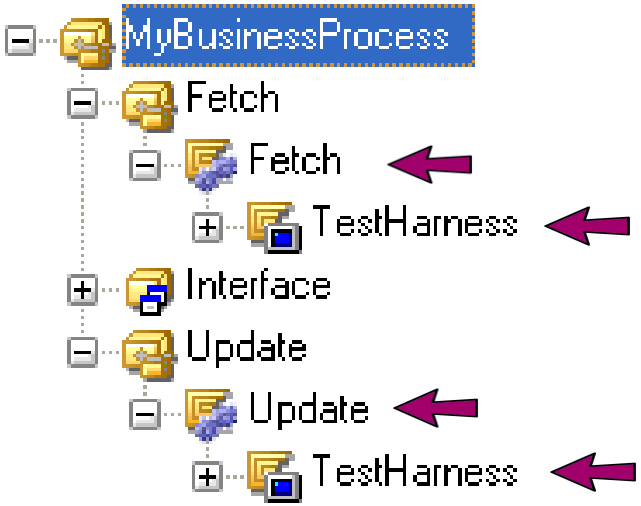
Business Service tier patterns

Scoped Service Function

- Black Box
- No data formatting
- Validation - Allowed Values
- Initialize Variables & Fields
- Inherit ErrorHandling / CheckCallStatus
- Message Handling
- ProcessBlockFetch

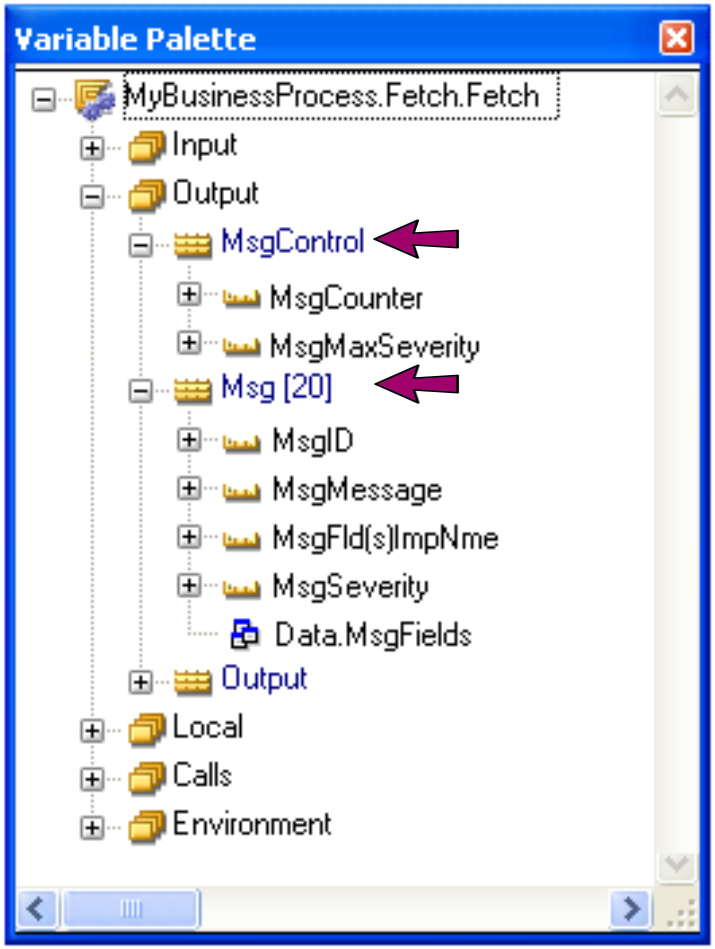
Scoped TestHarness / TestBench

- No Action Diagram coding
- No Replacement just IS A at entity level
- Testing in Isolation
- Identifying which Tier in Error
- Assume and recreate poor quality data entry
- Remove all panel controls – Raw data entry



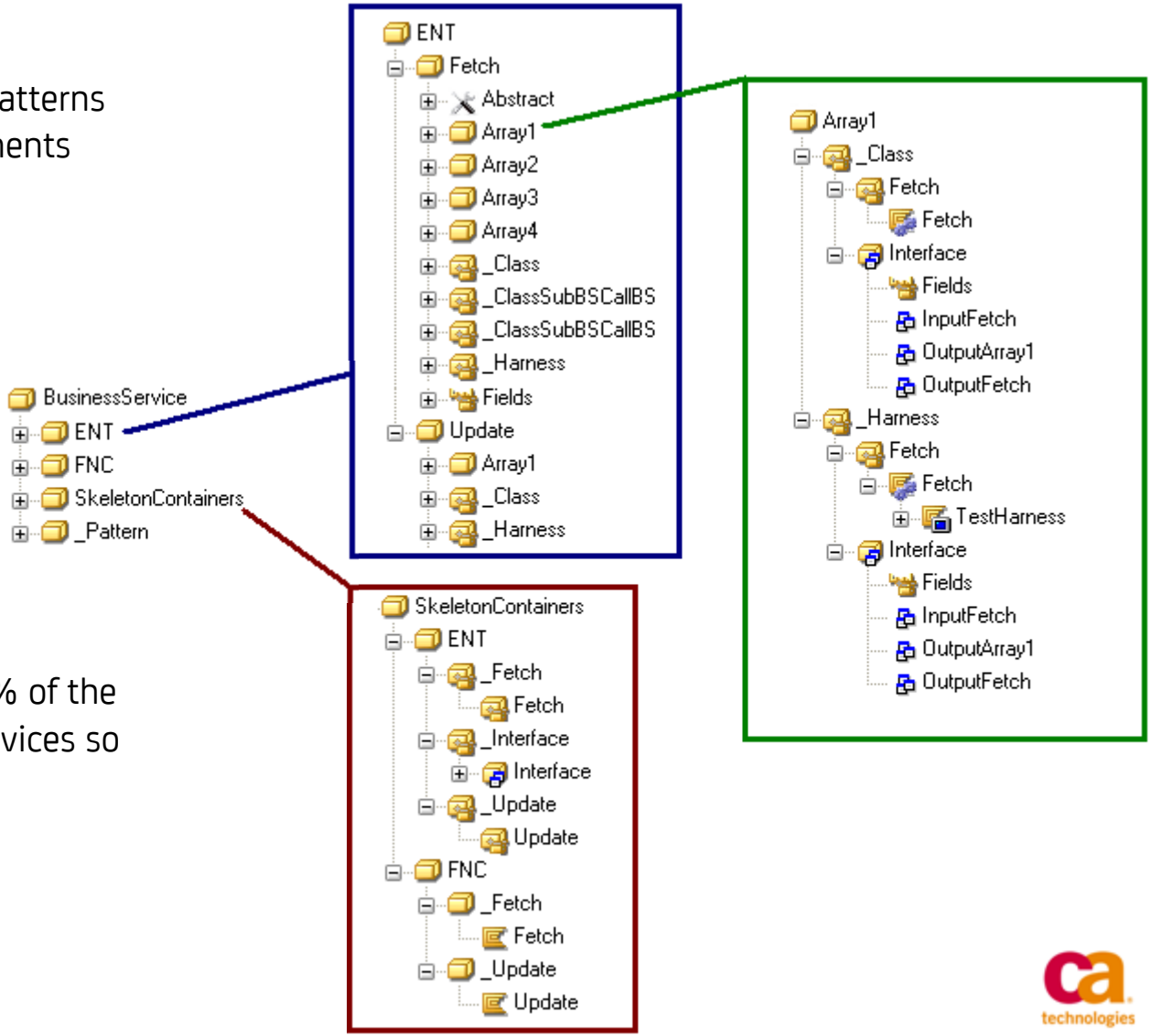
Business Service tier patterns

- Messaging
  - More time spent here less time explaining to calling Tiers developers Errors and Functionality.
  - Integrated into your Go Sub Send Message
  - No developer coding but for populating msg details
- **MsgControl (SingleInstance)**
  - MsgCounter – Total number of messages
  - MsgMaxSeverity – What was the most severe message contained in the Array
- **Msg (MultipleInstance)**
  - MsgID – so that calling tier can use their own messages but recognize yours – Msg Imp Nme
  - MsgFld(s)ImpNme – Calling tier recognizes what fields the msg refers to
  - MsgSeverity
    - Info
    - FailedValidation
    - ApplicationError



Business Service tier patterns

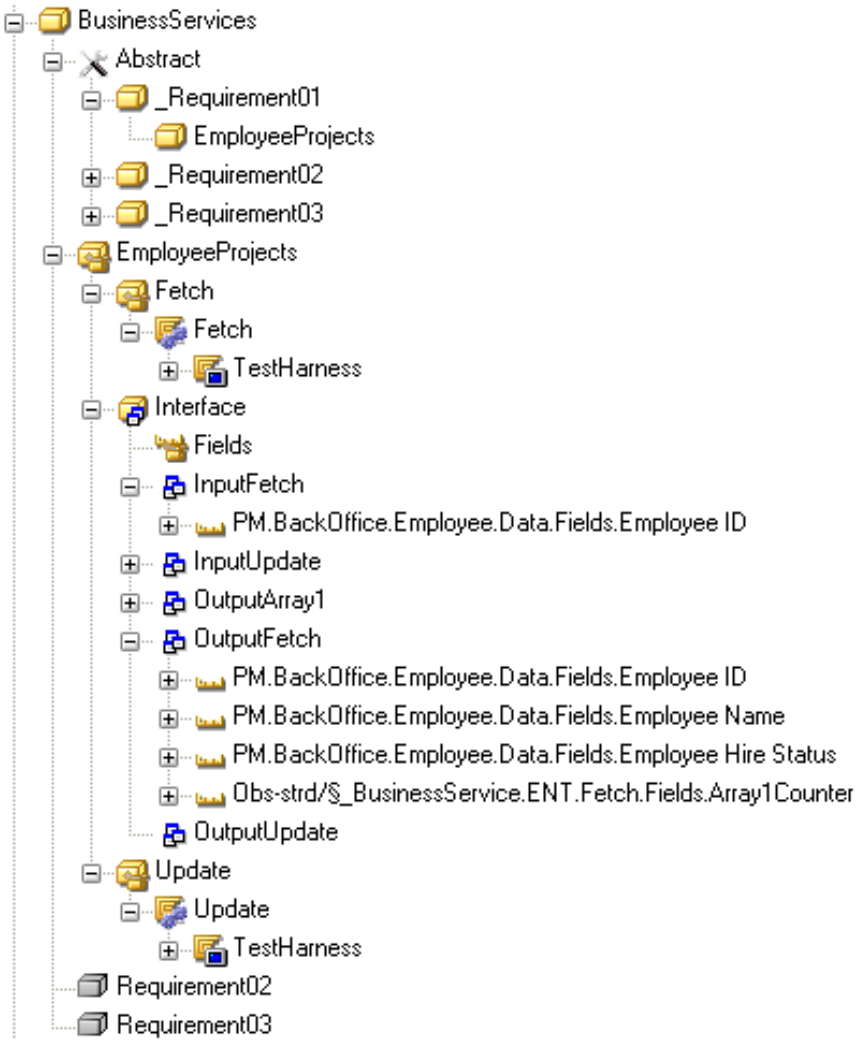
Aim was to produce a set of patterns that met 85% of the requirements



That said it has matched 100% of the current needs for Business Services so far…

business service tier demo

DEMO



Webservices tier patterns (published)

Aim was to model the Business Services into a SOAP service

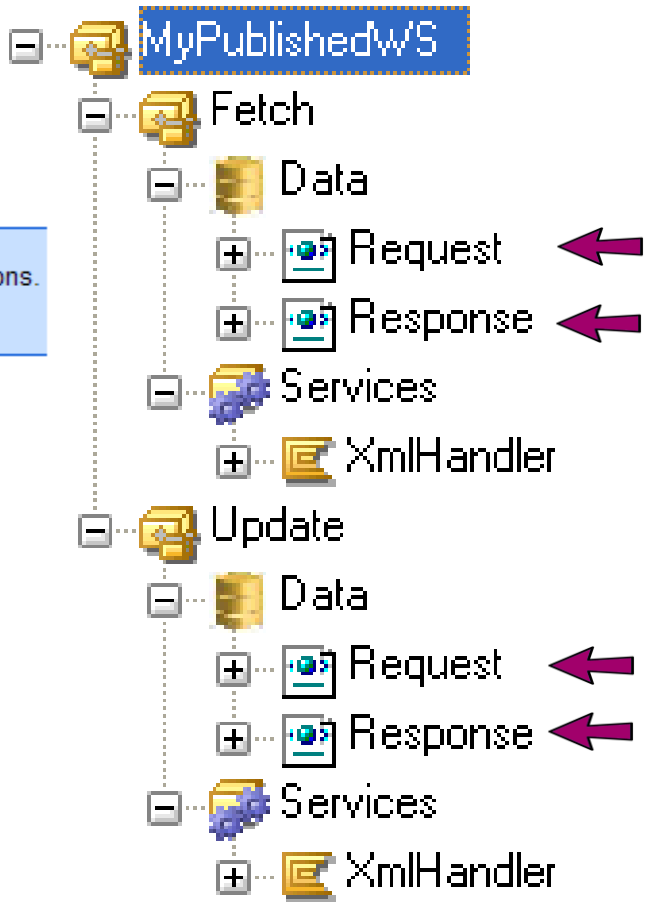
- Limited TransacXML experience and licenses within Department so solution must be simple, plug-in and play
- Advantage is we know exactly how the BS looks like as opposed to consumed services
- Create matching Fetch & Update patterns to wrapper the BS versions
- WEBSYDIAN™ Websydian TransacXML .ity granular patterns, first rate support and cumentation

- Entity Patterns
  - Implementation via Is A triples, replacement, Naming objects and Has relationships and absolutely no Action Diagram Code
- Skeleton Container Structures
  - Common Fetch, Update containers. Maintain the BS uniformity and allow entity level replacement.

ca technologies

Webservices tier patterns (published)

- A SOAP based webservice consists of a Request and Response.
- +GetNameOfField replaced +GetNameOfFieldByXML_Label



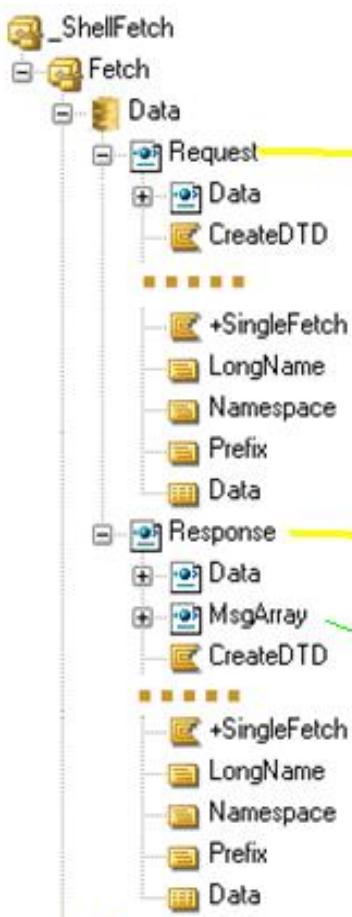
The behavior of both XmlElement and XmlRepeatingElement are controlled by a number of options. These options are set on the function ElementOptions.

fields are handled as if they were **simple elements**. This way it is possible to reuse existing fields without changing them."

- Function: ElementOptions NullValueCreate Set to Yes
- Limitation to approach: http://www.websydian.com/websydiandoc/v61/source/Websydian%20v6.1/TransacXML/wsyxml/parts.htm#ElementOptions

Webservices tier patterns (published)

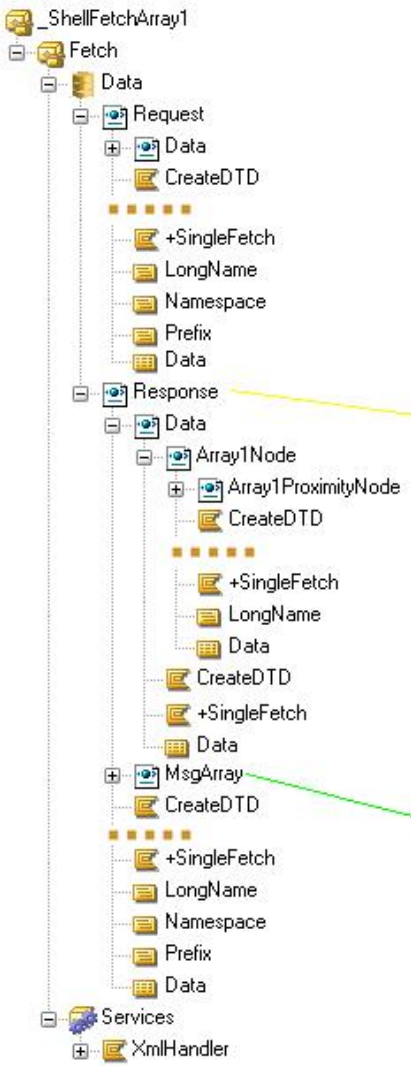
- Root Elements Request & Response (Complex Element)
  - LongName
  - Includes Data (Complex Element)
  - Response includes MsgArray (Complex Element)




```
- <soapenv:Body>
  - <emp:Request>
    - <emp:Data>
        <emp:EmpID>1</emp:EmpID>
      </emp:Data>
    </emp:Request>
  </soapenv:Body>
```

```
- <SOAP-ENV:Body>
  - <p1:Response>
    + <p1:Data>
    - <p1:MsgArray Counter="1" MaxSeverity="20">
      - <p1:MsgDtls>
          <p1:MsgID>MSG0010</p1:MsgID>
          <p1:MsgSeverity>20</p1:MsgSeverity>
          <p1:Message>The data could not be read</p1:Message>
          <p1:MsgElement>EmpID</p1:MsgElement>
        </p1:MsgDtls>
      </p1:MsgArray>
    </p1:Response>
  </SOAP-ENV:Body>
```

Webservices tier patterns (published)



W3C XML Document: Arrays Represented as

- Complex Element with Attributes
- Multiple occurrence Complex Element with Simple elements

Plex Model: Arrays Represented as

- Parent entity is a XMLElement has AttributeFieldUT
- Child entity is a XMLRepeatingElement has ElementFieldUT
- LongName

```
- <SOAP-ENV:Body>
  - <p1:Response>
    - <p1:Data>
        <p1:EmpID>1</p1:EmpID>
        <p1:EmpName>Employee One</p1:EmpName>
        <p1:EmpHireStatus>CT</p1:EmpHireStatus>
      - <p1:EmpProjects Counter="2">
        - <p1:Project>
            <p1:ProjectID>Pr1</p1:ProjectID>
            <p1:ProjectDescription>Project One</p1:ProjectDescription>
          </p1:Project>
        + <p1:Project>
        </p1:EmpProjects>
      </p1:Data>
      <p1:MsgArray Counter="0" MaxSeverity="" />
    </p1:Response>
  </SOAP-ENV:Body>
```

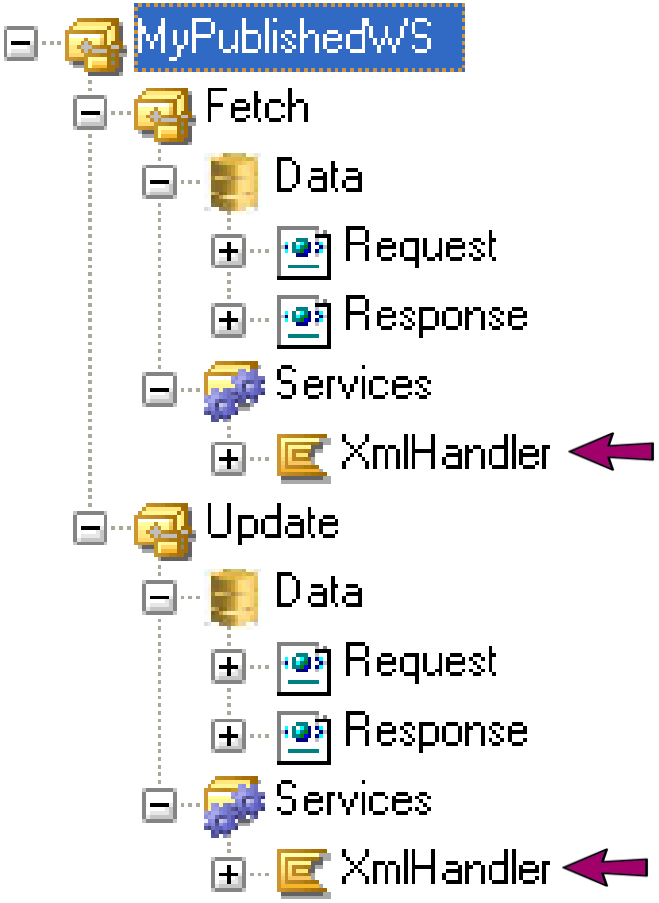
Webservices tier patterns (published)

- XMLHandler - read the incoming request (SoapPayload) and send outgoing Response (SoapResponseMesssage)

- Sub ProcessRequestDocument
  - Reads Root, Gets Data
  - Call BS with Input mapped from Get Data

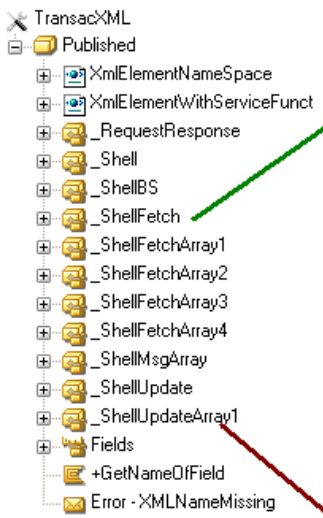
- Sub CreateDataResponseDocument
  - Creates Root
  - Detemines Response Type
  - Creates Data mapped from BS

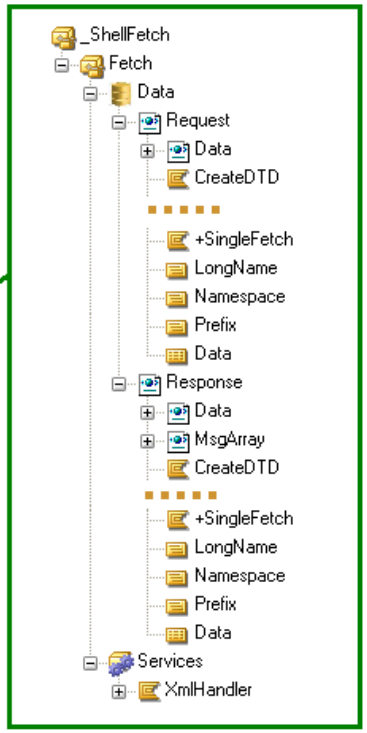
- Sub CreateErrorResponseDocument
  - Creates MsgArray mapped from BS

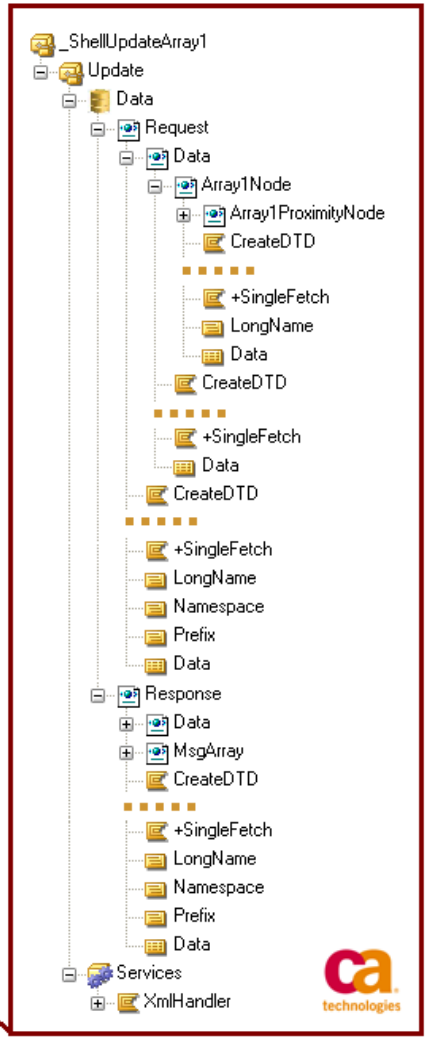


Webservices tier patterns (published)

Aim was to produce a set of patterns that wrappered the BS objects with no action diagram coding or logic or formatting





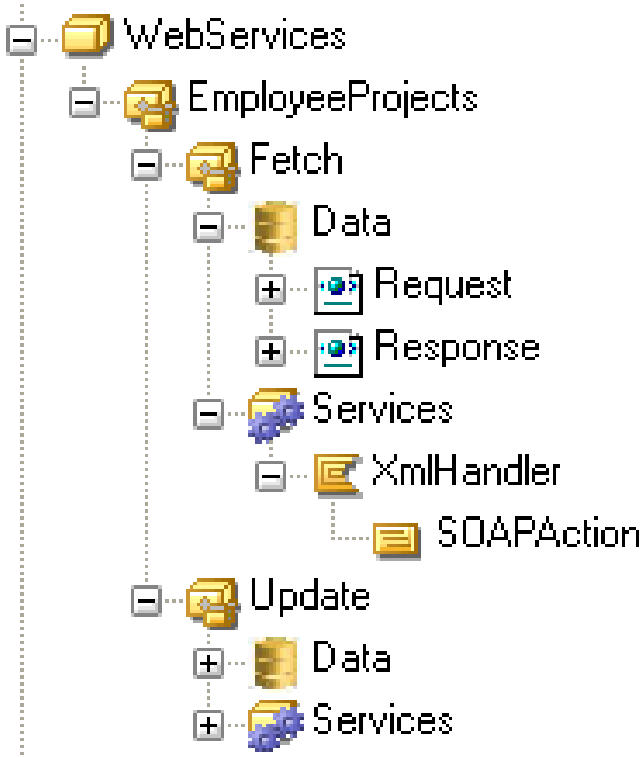


Some Outcomes:

- Reduction in development time
- Increased developer resources
- Decreased bugs
- Less tier developer confusion through Messaging and Interface uniformity

Webservices tier demo (published)

DEMO



Thank you for participating and that concludes the session and indeed concludes Chicago 2011 sessions.

ca technologies