W06
Mark Endrei

Patterns: Self-Service Application Solutions Using WebSphere V5.0

WebSphere Technical Exchange
Agenda

- Patterns for e-business
- Self-Service patterns
- Runtime patterns
- Product mappings
- Technology options
- Guidelines
- Patterns Development Kit
- More information
Patterns for e-business

Objectives

- To provide the smallest number of Patterns for e-business, which will allow IT architects in 80% of cases to quickly develop 80% of their required infrastructure by the re-use of proven:
  - Architecture patterns
  - Design patterns
  - Runtime patterns
  - Design, development and deployment guidelines
  - Sample code

- To provide IBM software (and hardware) development architects with a finite number of well-defined requirements, which they can use to optimize their product components
The Patterns layered asset model

Customer requirements

Composite patterns

Business patterns

Integration patterns

Application patterns

Runtime patterns

Product mappings

Any methodology

Best-Practice Guidelines
- Technology Choices
- Application Design
- Application Development
- System Management
- Samples
Business patterns

- Business patterns are the basic building blocks for solutions

- **Self-Service pattern**
  - Users accessing data (including business txns) on a 24x7 basis

- **Collaboration pattern**
  - Users working with one another to share data and information

- **Information Aggregation pattern**
  - Data from multiple sources aggregated and distilled

- **Extended Enterprise pattern**
  - Integrating data and processes across enterprise boundaries
Integration patterns

- Integration patterns provide the glue to combine Business patterns to form solutions

**Integration patterns**

**Access Integration pattern**

Provide seamless and consistent access to business functions

**Application Integration pattern**

Connect, interface or integrate databases and systems
Online brokerage

- Browser
- Specialized Devices
- Wireless

Integration patterns:
- Device Transcoding Data
- Customer Preferences
- Device Specific Formats
- Personalization Data

Business patterns:
- Electronic Brokerage
- Transactions
- Integrated Content
- Aggregated Content
- Industry Content

Information Aggregation:
- Access Integration

Content Providers
Content Syndicators
Extended Enterprise
Markets
Trades
Enterprise Systems
Application Integration

Self-Service
Information Aggregation
Portal
Integrated Content
Content Aggregator

Industry Content
Customer Preferences
Customer Transcoding Data

Notes Mail
Calendar
Directory
Personalization Data
Device Specific Formats
Aggregated Content

Device Transcoding Data
Customer Preferences

Access Integration
Agenda

- Patterns for e-business
- **Self-Service patterns**
- Runtime patterns
- Product mappings
- Technology options
- Guidelines
- Patterns Development Kit
- More information
Self-Service business pattern

Allows internal and external users to interact with enterprise systems and databases

Some examples of Self-Service applications

- An insurance company makes policy information available to users and allows them to apply for a policy online
- A bank allows customers to access their accounts and pay bills online
- A company allows its employees to change their medical plan, tax information, etc., online without having to call the HR office
Self-Service application patterns

Application patterns define the application logic and data components and how they interact
- There are currently seven Self-Service application patterns defined

1. Stand-alone Single Channel
2. Directly Integrated Single Channel
3. As-is Host
4. Customized Presentation to Host
5. Router
6. Decomposition
7. Agent
Self-Service application patterns
Self-Service application patterns

Business and IT drivers

<table>
<thead>
<tr>
<th>Business Drivers</th>
<th>Stand-Alone Single Channel</th>
<th>Directly-Integrated Single Channel</th>
<th>As-Is Host</th>
<th>Customized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to market</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Improve the organizational efficiency</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Reduce the latency of business events</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Easy to adapt during Mergers &amp; Acquisitions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integration across multiple delivery channels</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unified customer view across Lines of Business (LOB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IT Drivers</th>
<th>Stand-Alone Single Channel</th>
<th>Directly-Integrated Single Channel</th>
<th>As-Is Host</th>
<th>Customized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimize application complexity</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Minimize total cost of ownership(TCO)</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Leverage existing skills</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Leverage legacy investment</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Backend application integration</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Minimize enterprise complexity</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
Stand-alone Single Channel application pattern

Business applications not currently needing integration with other applications

- One delivery channel
- Presentation tier handles user interface
- Web application tier
  - Local application data
- Business logic resides in Web application tier
- Synchronous between presentation tier and Web application tier
Directly Integrated Single Channel application pattern

Point-to-point connectivity between the user and new or existing back-end applications

- One delivery channel
- Presentation tier handles user interface
- Web application tier
  - Local application data; Accesses one or more back-end applications
- Back-end applications
  - Existing business logic and back-end data
- Business logic can reside in Web application tier and back-end tier
- Synchronous between presentation tier and Web application tier
- Synch/asynch between application tier and the back-end
Agenda

- Patterns for e-business
- Self-Service patterns
- **Runtime patterns**
- Product mappings
- Technology options
- Guidelines
- Patterns Development Kit
- More information
Runtime patterns

The logical middleware structure that underpins the Application pattern

- Nodes group functional requirements
- Nodes interconnected to solve business problem
Stand-alone Single Channel::Runtime pattern

Outside World | Demilitarized Zone (DMZ) | Internal Network

- Public Key Infrastructure
- Domain Name Server
- User

- Protocol Firewall
- Web Application Server
- Domain Firewall

- Directory and Security Services
- Database

Stand-Alone Single Channel application

Presentation | Application

© 2003 IBM Corporation
Runtime pattern: Variation 1

<table>
<thead>
<tr>
<th>Outside World</th>
<th>Demilitarized Zone (DMZ)</th>
<th>Internal Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol Firewall</td>
<td>Web Server Redirector</td>
<td>Domain Firewall</td>
</tr>
<tr>
<td>Public Key Infrastructure</td>
<td>Application Server</td>
<td>Directory and Security Services</td>
</tr>
<tr>
<td>Domain Name Server</td>
<td></td>
<td>Database</td>
</tr>
<tr>
<td>User</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Stand-Alone Single Channel application

Presentation → Application → Database
Directly Integrated Single Channel::Runtime pattern

Outside World | Demilitarized Zone (DMZ) | Internal Network

- Public Key Infrastructure
- Domain Name Server
- User

Protocol Firewall

Web Application Server

Domain Firewall

Directory and Security Services

Existing Applications and Data

Directly Integrated Single Channel application

Presentation | Application

Application

Application
Runtime pattern: Variation 1
Runtime pattern: Other variations

Non-Functional Requirements::High Availability: Runtime patterns
- Basic Runtime pattern
- Runtime pattern: Single load balancer
- Runtime pattern: Load balancer hot standby
- Runtime pattern: Mutual high availability
- Runtime pattern: Wide area load balancing

Non-Functional Requirements::High Performance: Runtime patterns
- Basic Runtime pattern
- Runtime pattern: Redirectors
- Runtime pattern: Separation
- Runtime pattern: Caching proxy
Agenda

- Patterns for e-business
- Self-Service patterns
- Runtime patterns
- **Product mappings**
- Technology options
- Guidelines
- Patterns Development Kit
- More information
Product mappings

**Determining what products and platforms to use**
- Existing systems and platform investments
- Customer and developer skills available
- Customer choice

**IBM WebSphere Application Server V5.0 on**
- Windows
- AIX
- Linux
- OS/400 and z/OS coming soon

**Integration options**
- Web services
- J2EE Connectors and CICS
- Java Message Service and WebSphere MQ V5.3
Stand-alone Single Channel::Windows Product mapping

**Demilitarized zone**
- Windows 2000 + SP3
- IBM WebSphere Application Server V5.0 HTTP Plug-in
- IBM HTTP Server 1.3.26

**External world**
- Protocol Firewall
- Web Server Redirector
- Domain Firewall
- Application Server

**Internal network**
- Windows 2000 + SP3
- IBM WebSphere Application Server V5.0

**Directory and Security Services**
- LDAP
- Windows 2000 + SP3
- IBM SecureWay Directory V3.2.1
- IBM HTTP Server 1.3.19.1
- IBM GSKit 5.0.3
- IBM DB2 UDB EE V7.2 + FP5

**Database**
- Windows 2000 + SP3
- IBM DB2 UDB ESE V8.1
Directly Integrated Single Channel::Windows Product mapping

Demilitarized zone
- Windows 2000 + SP3
- IBM WebSphere Application Server V5.0 HTTP Plug-in
- IBM HTTP Server 1.3.26

Internal network
- Windows 2000 + SP3
- IBM SecureWay Directory V3.2.1
- IBM HTTP Server 1.3.19.1
- IBM GSKit 5.0.3
- IBM DB2 UDB EE V7.2 + FP5

Directory and Security Services
- LDAP

Application Server
- Windows 2000 + SP3
- IBM WebSphere Application Server V5.0
- IBM WebSphere MQ 5.3
- Message-driven bean application

Existing Applications and Data
- Windows 2000 + SP3
- IBM WebSphere Application Server V5.0
- IBM WebSphere MQ 5.3
- Message-driven bean application

Database
- Windows 2000 + SP3
- IBM DB2 UDB ESE V8.1

Outside world
- Protocol Firewall
- Domain Firewall
- Web Server Redirector
Stand-alone Single Channel::Linux Product mapping

Demilitarized zone

- Red Hat Linux AS 2.1
- IBM WebSphere Application Server V5.0 HTTP Plug-in
- IBM HTTP Server 1.3.26

Outside world

Protocol Firewall

Web Server Redirector

Domain Firewall

Internal network

Application Server

- Red Hat Linux AS 2.1
- IBM WebSphere Application Server V5.0

Directory and Security Services

- Windows 2000 + SP3
- IBM SecureWay Directory V3.2.1
- IBM HTTP Server 1.3.19.1
- IBM GSKit 5.0.3
- IBM DB2 UDB EE V7.2 + FP5

LDAP

Database

- Red Hat Linux AS 2.1
- IBM DB2 UDB ESE V8.1
Directly Integrated Single Channel::Linux Product mapping

Demilitarized zone
- Red Hat Linux AS 2.1
- IBM WebSphere Application Server V5.0 HTTP Plug-in
- IBM HTTP Server 1.3.26

Internal network
- Directory and Security Services
  - LDAP
- Application Server
  - Red Hat Linux AS 2.1
  - IBM WebSphere Application Server V5.0
  - JMS Option:
    - Red Hat Linux AS 2.1
    - IBM WebSphere Application Server V5.0
    - IBM WebSphere MQ 5.3
- Database
  - Red Hat Linux AS 2.1
  - IBM DB2 UDB ESE V8.1
- Existing Applications and Data
  - Windows 2000 + SP3
  - IBM SecureWay Directory V3.2.1
  - IBM HTTP Server 1.3.19.1
  - IBM GSKit 5.0.3
  - IBM DB2 UDB EE V7.2 + FP5
- Web Services Option:
  - Red Hat Linux AS 2.1
  - IBM WebSphere Application Server V5.0
  - IBM HTTP Server 1.3.26
  - IBM DB2 UDB ESE 8.1
  - Web service EJB application
- J2C Option:
  - z/OS Release 1.3
  - IBM CICS Transaction Gateway V5.0
  - IBM CICS Transaction Server V2.2
  - CICS C-application
- JMS Option:
  - Red Hat Linux AS 2.1
  - IBM WebSphere Application Server V5.0
  - IBM WebSphere MQ 5.3
  - Message-driven bean application
Agenda

- Patterns for e-business
- Self-Service patterns
- Runtime patterns
- Product mappings
- Technology options
- Guidelines
- Patterns Development Kit
- More information
Web services

What are Web services?
- Web services are self-contained, self-describing, modular applications
- Web services combine two ubiquitous technologies
  - XML, the universal data description language
  - HTTP, widely supported by browsers and Web servers

Web services can be published, located, and invoked across the Web using
- SOAP, an XML-based RPC and messaging protocol
- WSDL, a descriptive interface and protocol binding language
- UDDI, a registry mechanism used to locate Web service descriptions
Web services - continued

**Advantages of Web services**
- Protect investments in legacy systems by using Web services to wrap these systems for integration with modern systems
- Web services are platform neutral so they can be invoked by non-Java clients
- Web services can be dynamically "discovered" and thus are "visible" to the entire Internet/intranet community

**Disadvantages of Web services**
- Adding the Web services provider stack to existing enterprise systems can be a problem
  - SOAP server footprint is significant; Technology is relatively new
- Binding to Web services dynamically requires that the contents of the registry be trusted
  - Currently, only private registries provide such control over contents
J2EE Connector

What is the J2EE Connector Architecture?
- Provides a standard way of accessing enterprise systems from J2EE applications
- Defines the common client interface (CCI) API
- The link from the API to the EIS is called a resource adapter and is provided by a third-party vendor
- Adapters are packaged as resource adapter archive (RAR) files
- J2C 1.0 support is a requirement of the J2EE 1.3 specification
- It is similar to the model for JDBC drivers
J2EE Connector - continued

Advantages of J2EE Connectors
- The CCI simplifies application integration with diverse EISs
  - Easily plug third-party or home-grown adapters into your applications
- An EIS adapter implementation can be used by all client applications
- J2EE connectors facilitate scalability and provide quality of service features transparently to the client application
- J2C-compliant adapters are portable across J2EE application servers
- J2C has low intrusion on the enterprise system, as native client interfaces are utilized

Disadvantages of J2EE Connectors
- J2C only supports synchronous communication. Asynchronous support is expected in the J2C 1.5 or 2.0 specification
- With J2C 1.0, parts of the CCI are resource adapter specific. If you change the resource adapter the client application will be impacted
Java Message Service

What is Java Message Service?
- The JMS API enables a Java programmer to access message-oriented middleware from the Java programming model
- JMS has two messaging styles
  - Point-to-point
  - Publish/subscribe
- Communications are asynchronous
  - Clients can receive messages without making a request
  - Clients can send messages without waiting for a reply
- Communications are loosely coupled
  - The sender and receiver do not have to be active or aware of each other
- JMS is only a specification
  - Each enterprise messaging system vendor must provide classes that implement the specification for their specific messaging system
Java Message Service - continued

Advantages of JMS

- It is the first enterprise messaging API that has achieved wide cross-industry support
- It leverages existing, enterprise-proven messaging systems
- It simplifies development of enterprise applications by providing standard messaging concepts and conventions
- A common interface for accessing diverse messaging systems
- Extend existing message-based applications by adding new JMS clients that are integrated fully with existing non-JMS clients

Disadvantages of JMS

- The common interface may not support all the functionality offered by the messaging vendor
  - You might have to write vendor-specific code to access such functionality
XML messaging

XML can be used to specify the content of messages between servers, within an enterprise or for business-to-business connections

Advantages of XML
- Industry standard for data interchange
- Cross-platform tool support
- Human readable

Disadvantages of XML
- Complexity, large size, standardization

DTD or XML Schema can be used to define/validate XML documents

Disadvantages of DTD
- Poor typing, not defined as XML

Advantages of XML Schema
- Strong typing for elements and attributes, null values
- Defined as XML, making them programmatically accessible
Integration technology selection guidelines

The current infrastructure and skills
- Do you have a messaging system on the enterprise tier?
  - Then JMS makes sense
- Do you have an EIS system, such as CICS or IMS?
  - J2C might be the better choice
- Infrastructure and skills flexible/developing?
  - Web services can help

Time to market
- Web service enabling an application can be relatively fast with the Web services development tools available
Integration technology selection guidelines - continued

Future expansion plans
- Do you need to integrate with existing infrastructure and with planned infrastructure?
  - Web services can provide a low-dependency and low-cost migration path in such a case

Reliability
- JMS and WebSphere MQ can provide assured transfer of data, even when the enterprise application is unavailable

Transaction support
- Web services currently do not offer support for transactions
  - If your application needs transactional management, it might be worthwhile considering either JMS or J2EE Connectors
Agenda

- Patterns for e-business
- Self-Service patterns
- Runtime patterns
- Product mappings
- Technology options
- Guidelines
- Patterns Development Kit
- More information
Guidelines - Development and runtime environment

IBM WebSphere Studio Application Developer V5.0 base edition
- Integrated J2EE application development and test environment
- Visual editors and wizards for working with J2EE applications, modules, EJBs, servlets, JSPs, XML, etc.
- Web services wizards for generating providers and clients

IBM WebSphere Application Server V5.0 base edition
- As per Runtime patterns

IBM WebSphere MQ V5.3
- As per Runtime patterns

IBM CICS Transaction Server and Transaction Gateway
- As per Runtime patterns
- ECI and EPI J2C adapters with the IBM CICS Transaction Gateway
Guidelines - Web application frameworks

True to the MVC design pattern, Struts applications have three major components
- Controller - Struts ActionServlet and Action class
- View - Struts form beans and JavaServer Pages
- Model - application's business logic

Struts helps developers
- Map HTTP parameters to JavaBeans
- Validate that a form is filled in correctly
- Provide standard error responses
- Message internationalization
- Avoid hard coding links to JSPs

Like all frameworks, Struts has disadvantages
- Customization may be difficult, but can also prevent bad practices
- Learning curve
Web services - Guidelines

Web service provider and requester application structure
Guidelines - Web services

Static vs. dynamic discovery

There are two ways of binding to a Web service

- **Static service**
  - Service definition is obtained through a proprietary channel (an e-mail, for example) and stored locally

- **Dynamic discovery**
  - Service definition is obtained from a trusted UDDI registry
  - Client application searches the registry at runtime
Guidelines - J2EE Connectors and CICS

Local vs remote CICS Transaction Gateway (TG)

- Local CICS TG on application server tier
- Remote CICS TG on z/OS enterprise tier

- Remote CICS TG on z/OS recommended in most situations
  - CICS client-interface components are provided by z/OS enterprise tier
  - The CICS TG on zSeries can be shared by multiple client applications
  - Availability of the CICS client-interface depends only on enterprise tier
  - Standard tools and procedures can be used at the enterprise tier
  - CICS resources and skills can remain focused on z/OS
Guidelines - JMS and WebSphere MQ

Local vs remote queue managers

- Connecting to the local queue manager has several major advantages
  - Higher probability of connecting to a local queue manager than a remote queue manager
  - The time it takes to establish a network connection to the queue manager is avoided
  - The local queue manager can distribute the work among multiple brokers
Guidelines - XML messages

- Use descriptive tag and attribute names
- Group elements in related sets
- Separate metadata from the real content
  - Use Header and Body elements for example
- Include a message version number in the message metadata

```xml
<?xml version="1.0" encoding="UTF-8"?>
<RateRequest
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="RateRequest.xsd">
  <Header>
    <SourceName>PDKLite</SourceName>
    <Version>1</Version>
    <CreateDate>2001-12-31T12:00:00</CreateDate>
  </Header>
  <Body>
    <Planet>JUPITER</Planet>
  </Body>
</RateRequest>
```
XML development

WebSphere Studio Application Developer V5.0

- XML Editor
- DTD / XML Schema generator
- JavaBeans generator
Class diagram for XML message

```
Message
+ Message ( )
+ Message ( )
# setElement ( )
# getElement ( )
+ toString ( )

DOMParser

MessageResolver
+ resolveEntity ( )

DOMWriter

«JavaInterface» Document

ExchangeRateRequest
+ ExchangeRateRequest ( )
+ ExchangeRateRequest ( )
+ getSourceName ( )
+ setSourceName ( )
+ getVersion ( )
+ setVersion ( )
+ getCreateDate ( )
+ setCreateDate ( )
+ getPlanet ( )
+ setPlanet ( )

DocumentBuilderFactory

DocumentBuilder

DocumentBuilderFactory
```
Sequence diagram for Web services requester and provider

1: FinanceBean

2: ExchangeRateBean

3: ExchangeRateWSBean

4: ExchangeRateServerProxy

5: ExchangeRateRequest

6: ExchangeRateReply

7: RPC RouterServlet

8: ExchangeRateServerBean

9: ExchangeRateBean

10: FinanceBean

1: getExchangeRate (planet)

2: getExchangeRate (planet)

3: setEndPoint (url)

4: ExchangeRateRequest (msgSource, msgPlanet)

5: toString()

6: getExchangeRateXML (reqMsg)

7: SOAP Request

8: getExchangeRateXML (reqMsg)

9: ExchangeRateReply (msg)

10: getRate()
Source code for invoking Web service

```java
/**
 * Returns the exchange rate between the Guild and a planet
 * @param planet The name of the planet to get exchange rate for
 * @return int The exchange rate between guild and planet
 * @throws EJBEException
 */

public int getExchangeRate(String planet) throws EJBEException {

    int rate = 0;

    try {
        com.ibm.pdk.ejb.exchangeWS.proxy.ExchangeRateServerProxy proxy =
            new com.ibm.pdk.ejb.exchangeWS.proxy.ExchangeRateServerProxy();
        proxy.setEndPoint(new URL(endPointURL));

        // create the XML request message
        ExchangeRateRequest exchangeRateRequest = new ExchangeRateRequest("PDKLite", planet);
        String reqMsg = exchangeRateRequest.toString();

        String repMsg = proxy.getExchangeRateXML(reqMsg);

        // create an XML reply object from the reply message and get the rate
        ExchangeRateReply exchangeRateReply = new ExchangeRateReply(repMsg);
        rate = exchangeRateReply.getRate();
    } catch (Exception e) {
        // ...
    }

    return rate;
}
```
Agenda

- Patterns for e-business
- Self-Service patterns
- Runtime patterns
- Product mappings
- Technology options
- Guidelines
- Patterns Development Kit
- More information
Patterns Development Kit
PDK architecture overview
Agenda

- Patterns for e-business
- Self-Service patterns
- Runtime patterns
- Product mappings
- Technology options
- Guidelines
- Patterns Development Kit
- More information
More information

- This presentation is based on redbook SG24-6591
- ITSO Patterns residencies
- E-mail: endrei@us.ibm.com
More information

- **Books**
  - *Patterns for e-business: A Strategy for Reuse*
    by Jonathan Adams, Srinivas Koushik, Guru Vasudeva, George Galambos
    Published by IBM Press
    ISBN: 1-931182-02-7

- **Web sites**
  - ibm.com/developerWorks/patterns
  - ibm.com/redbooks