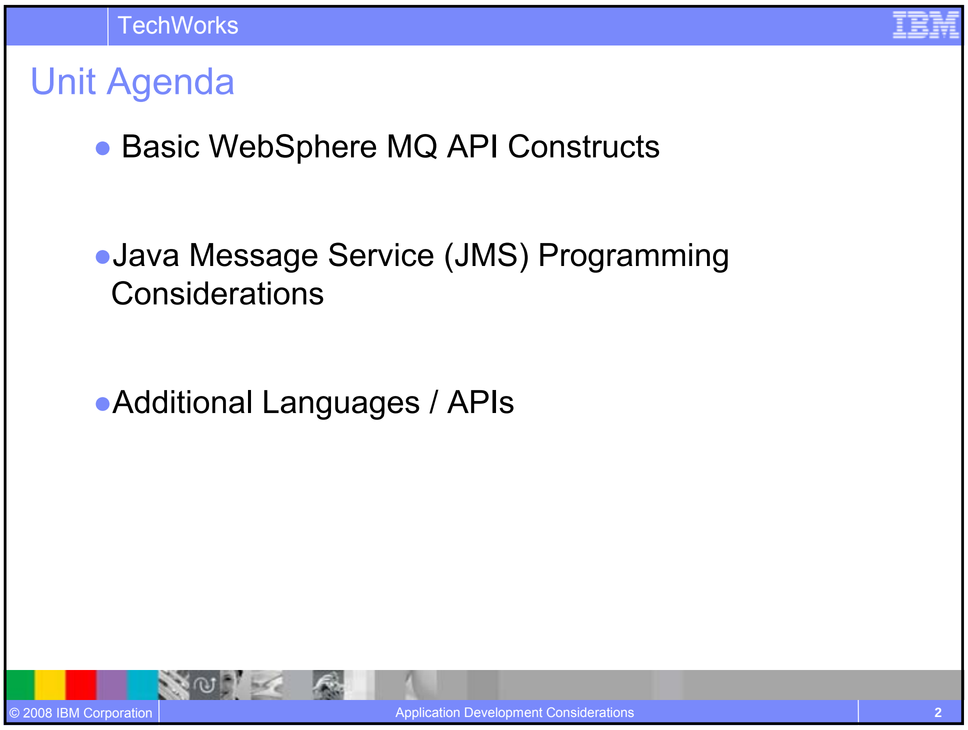
IBM Software Group

WebSphere MQ V7.0

Application Development

© 2008 IBM Corporation

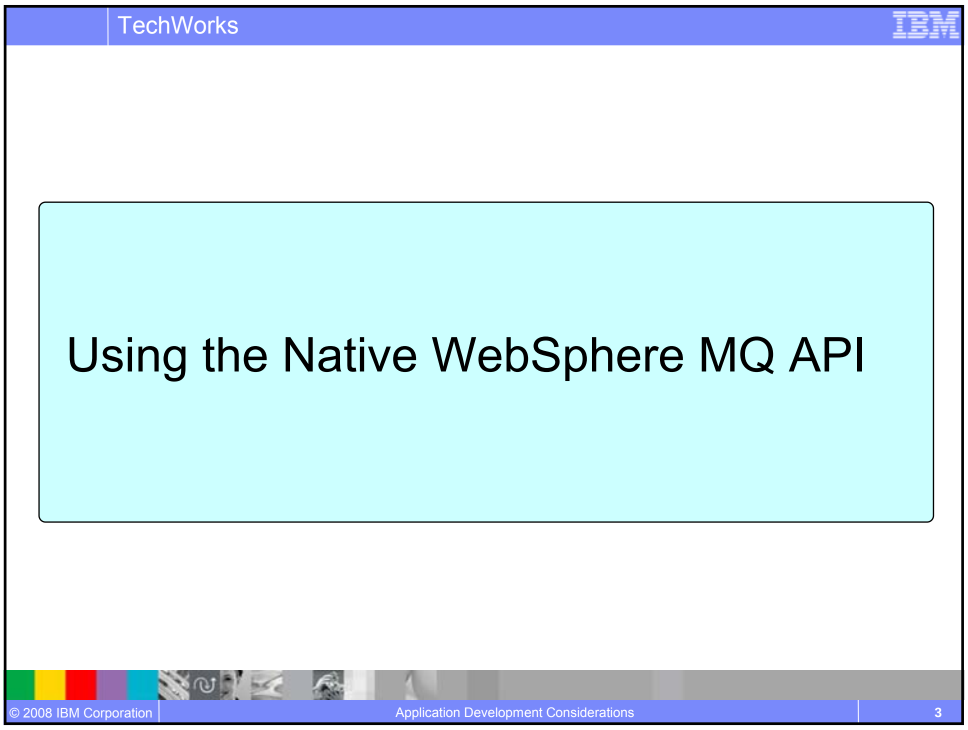


TechWorks

Unit Agenda

- Basic WebSphere MQ API Constructs
- Java Message Service (JMS) Programming Considerations
- Additional Languages / APIs

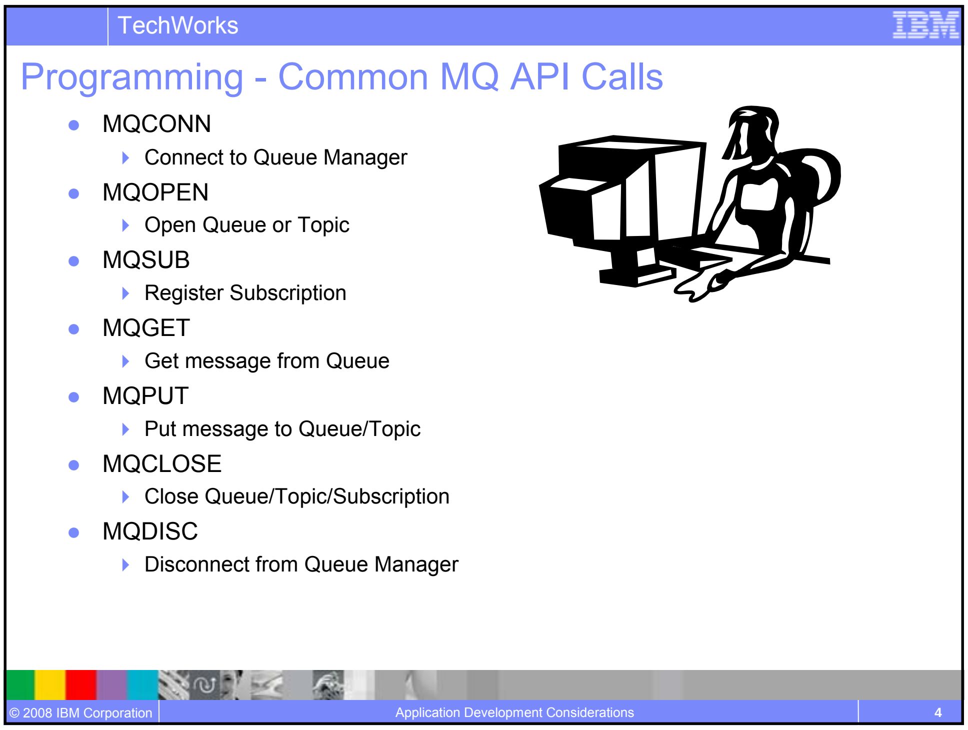
© 2008 IBM Corporation | Application Development Considerations | 2



TechWorks

Using the Native WebSphere MQ API

© 2008 IBM Corporation | Application Development Considerations | 3

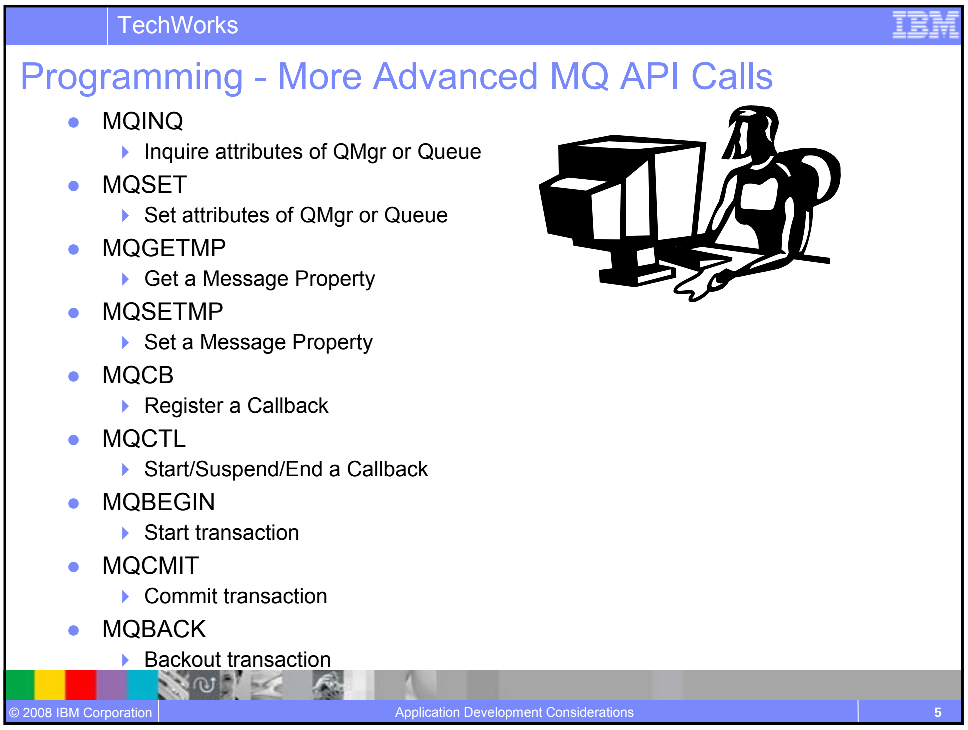


TechWorks

Programming - Common MQ API Calls

- MQCONN
  - Connect to Queue Manager
- MQOPEN
  - Open Queue or Topic
- MQSUB
  - Register Subscription
- MQGET
  - Get message from Queue
- MQPUT
  - Put message to Queue/Topic
- MQCLOSE
  - Close Queue/Topic/Subscription
- MQDISC
  - Disconnect from Queue Manager

© 2008 IBM Corporation | Application Development Considerations | 4

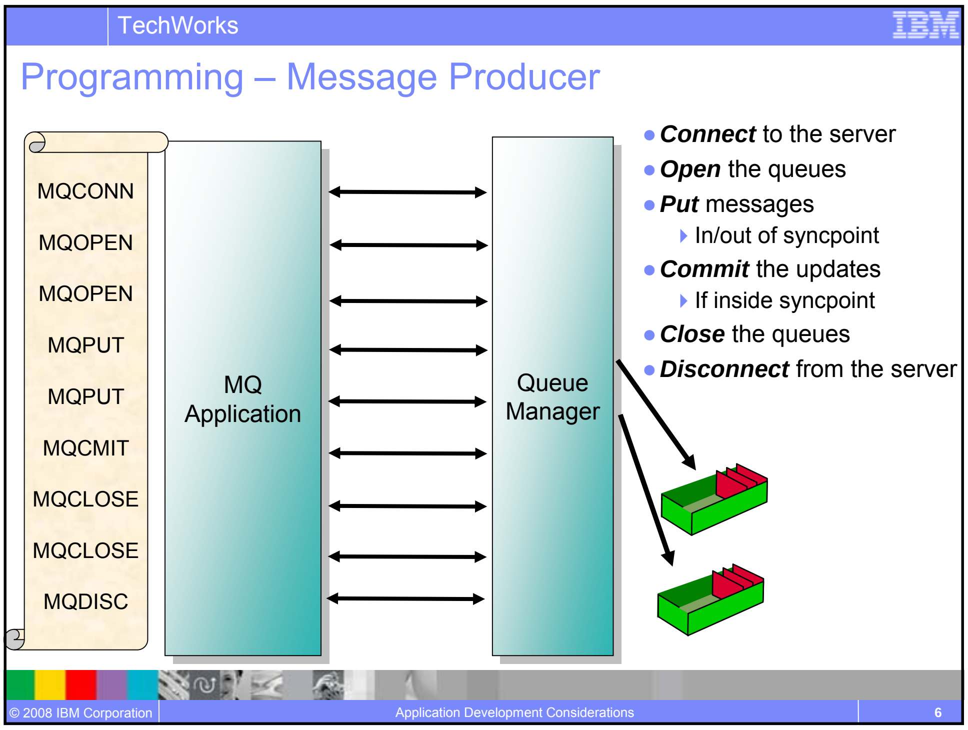


TechWorks

Programming - More Advanced MQ API Calls

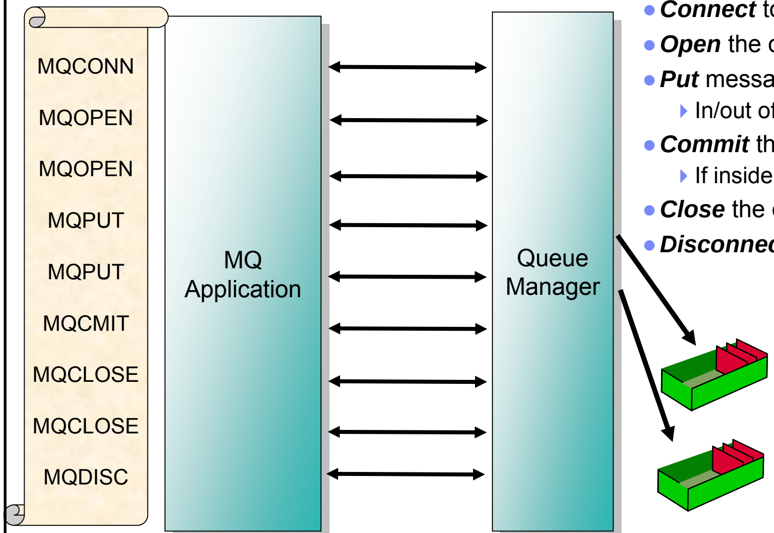
- MQINQ
  - Inquire attributes of QMgr or Queue
- MQSET
  - Set attributes of QMgr or Queue
- MQGETMP
  - Get a Message Property
- MQSETMP
  - Set a Message Property
- MQCB
  - Register a Callback
- MQCTL
  - Start/Suspend/End a Callback
- MQBEGIN
  - Start transaction
- MQCMIT
  - Commit transaction
- MQBACK
  - Backout transaction

© 2008 IBM Corporation | Application Development Considerations | 5



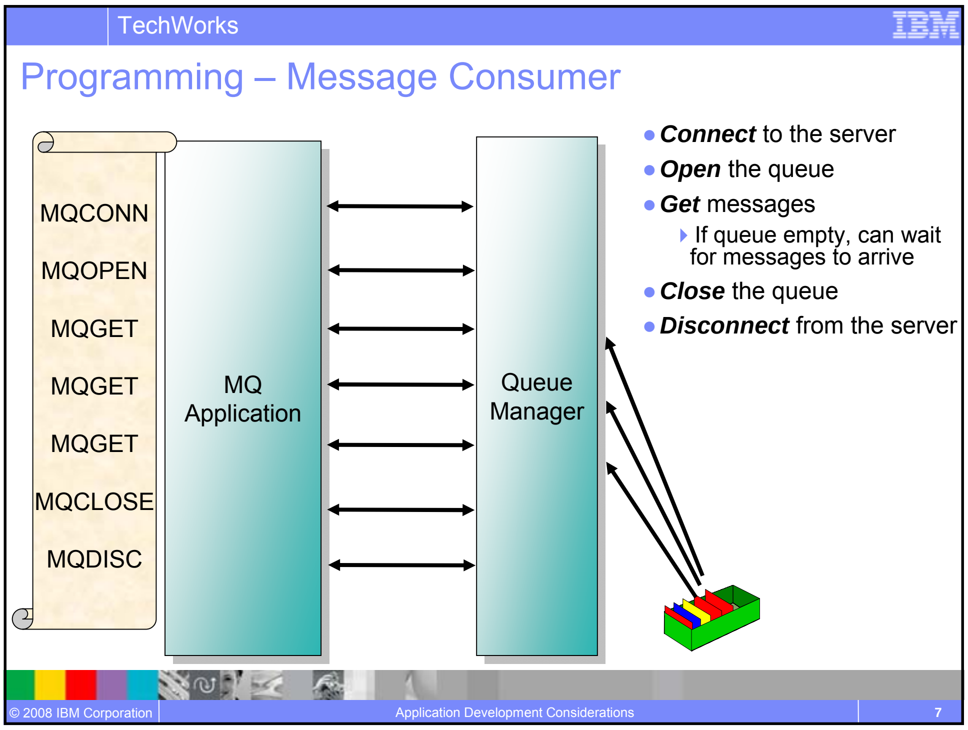
TechWorks

Programming – Message Producer



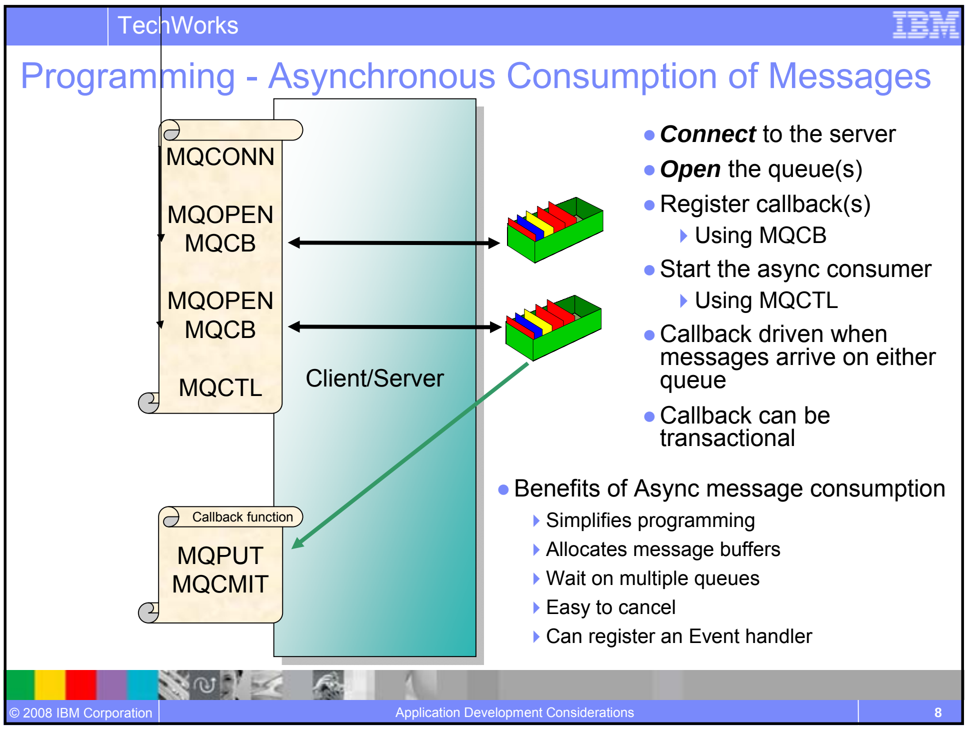
- ***Connect*** to the server
- ***Open*** the queues
- ***Put*** messages
  - In/out of syncpoint
- ***Commit*** the updates
  - If inside syncpoint
- ***Close*** the queues
- ***Disconnect*** from the server

© 2008 IBM Corporation | Application Development Considerations | 6



Programming – Message Consumer

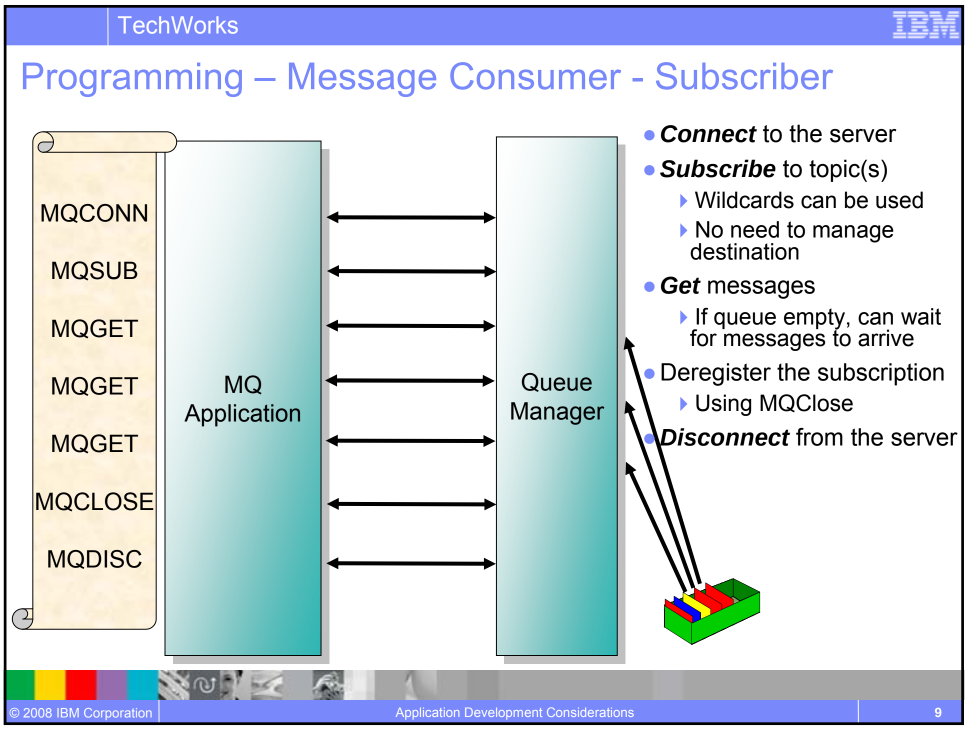
- ***Connect*** to the server
- ***Open*** the queue
- ***Get*** messages
  - If queue empty, can wait for messages to arrive
- ***Close*** the queue
- ***Disconnect*** from the server



Programming - Asynchronous Consumption of Messages

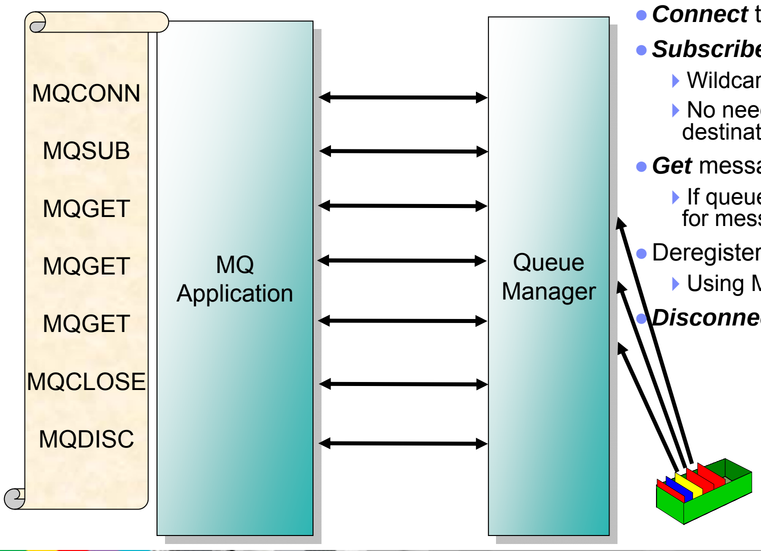
- ***Connect*** to the server
- ***Open*** the queue(s)
- Register callback(s)
  - Using MQCB
- Start the async consumer
  - Using MQCTL
- Callback driven when messages arrive on either queue
- Callback can be transactional

- Benefits of Async message consumption
  - Simplifies programming
  - Allocates message buffers
  - Wait on multiple queues
  - Easy to cancel
  - Can register an Event handler



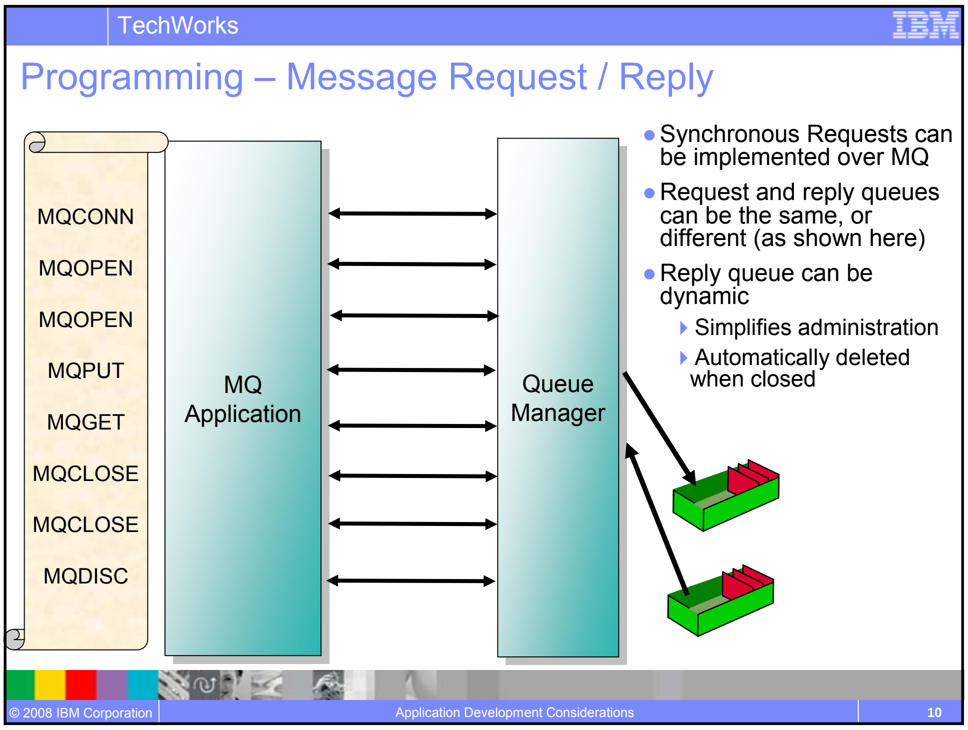
TechWorks

Programming – Message Consumer - Subscriber



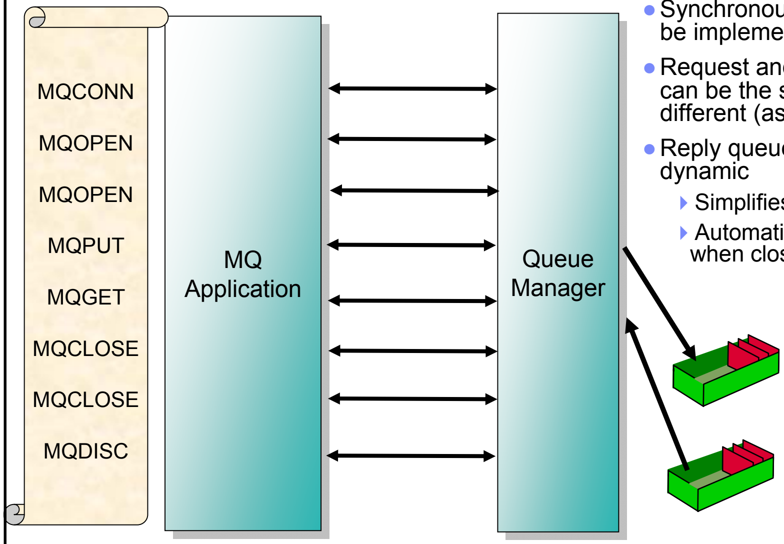
- ***Connect*** to the server
- ***Subscribe*** to topic(s)
  - Wildcards can be used
  - No need to manage destination
- ***Get*** messages
  - If queue empty, can wait for messages to arrive
- Deregister the subscription
  - Using MQClose
- ***Disconnect*** from the server

© 2008 IBM Corporation | Application Development Considerations | 9



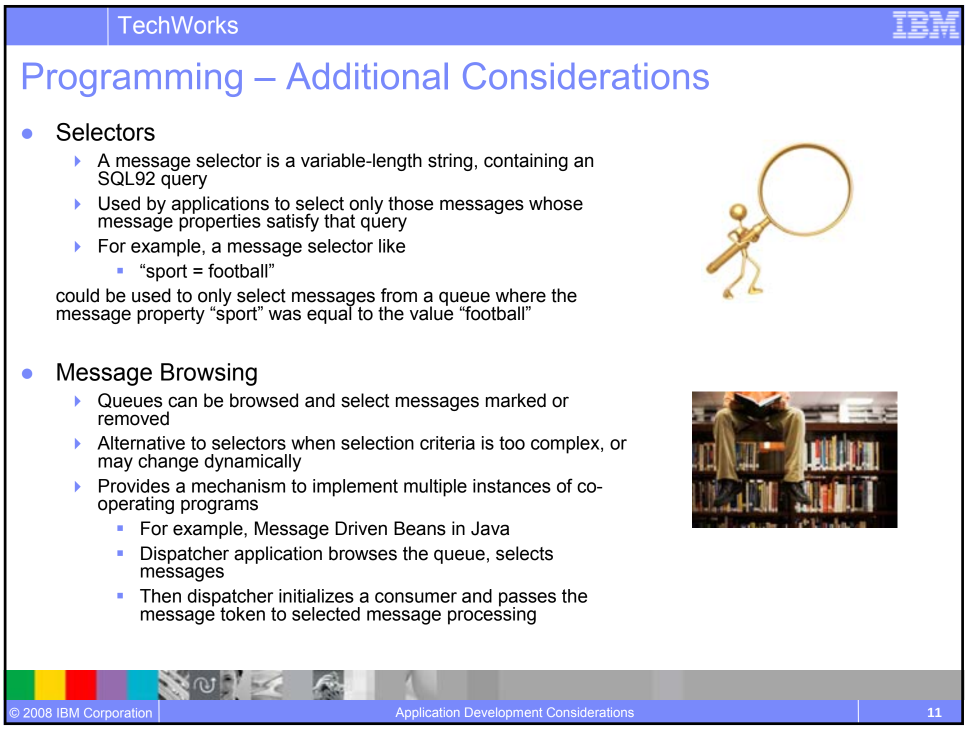
TechWorks

Programming – Message Request / Reply



- Synchronous Requests can be implemented over MQ
- Request and reply queues can be the same, or different (as shown here)
- Reply queue can be dynamic
  - Simplifies administration
  - Automatically deleted when closed

© 2008 IBM Corporation | Application Development Considerations | 10



TechWorks

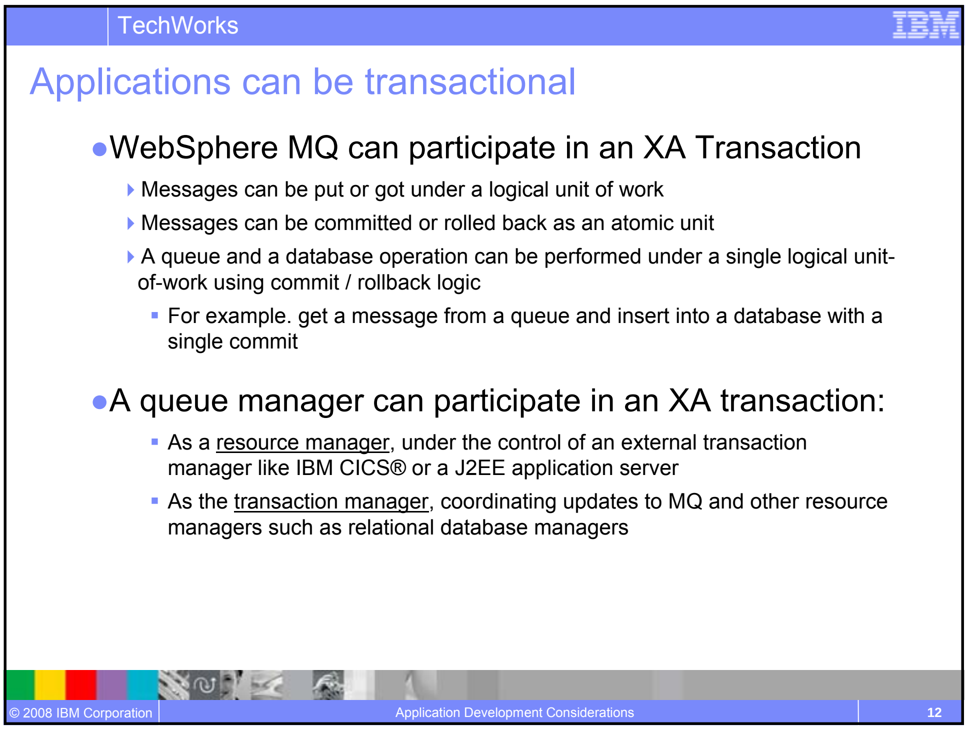
Programming – Additional Considerations

- Selectors
  - A message selector is a variable-length string, containing an SQL92 query
  - Used by applications to select only those messages whose message properties satisfy that query
  - For example, a message selector like
    - “sport = football”

  could be used to only select messages from a queue where the message property “sport” was equal to the value “football”

- Message Browsing
  - Queues can be browsed and select messages marked or removed
  - Alternative to selectors when selection criteria is too complex, or may change dynamically
  - Provides a mechanism to implement multiple instances of co-operating programs
    - For example, Message Driven Beans in Java
    - Dispatcher application browses the queue, selects messages
    - Then dispatcher initializes a consumer and passes the message token to selected message processing

Application Development Considerations 11

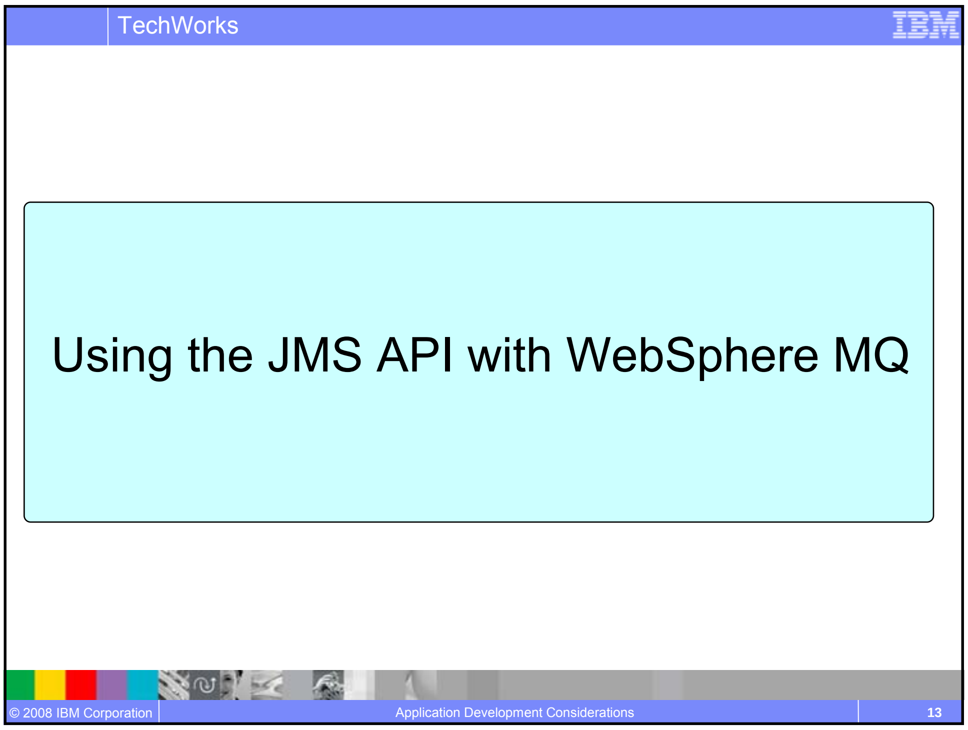


TechWorks

Applications can be transactional

- WebSphere MQ can participate in an XA Transaction
  - Messages can be put or got under a logical unit of work
  - Messages can be committed or rolled back as an atomic unit
  - A queue and a database operation can be performed under a single logical unit-of-work using commit / rollback logic
    - For example. get a message from a queue and insert into a database with a single commit
- A queue manager can participate in an XA transaction:
  - As a <u>resource manager</u>, under the control of an external transaction manager like IBM CICS® or a J2EE application server
  - As the <u>transaction manager</u>, coordinating updates to MQ and other resource managers such as relational database managers

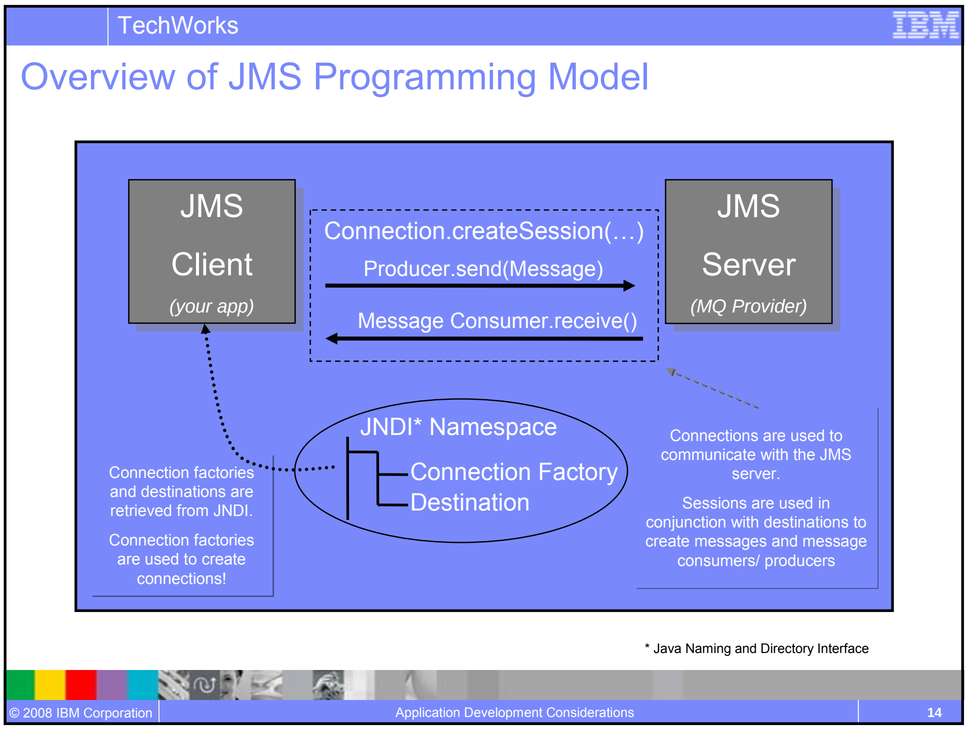
Application Development Considerations 12



TechWorks

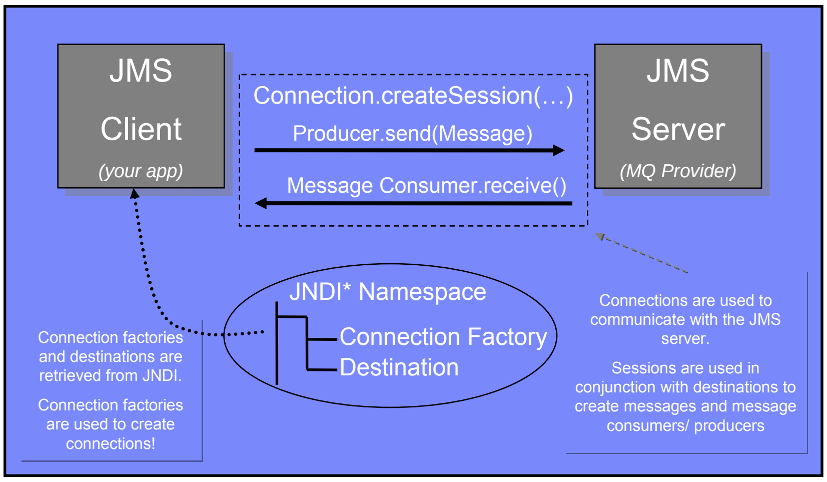
Using the JMS API with WebSphere MQ

Application Development Considerations 13



TechWorks

Overview of JMS Programming Model



Connection factories and destinations are retrieved from JNDI.

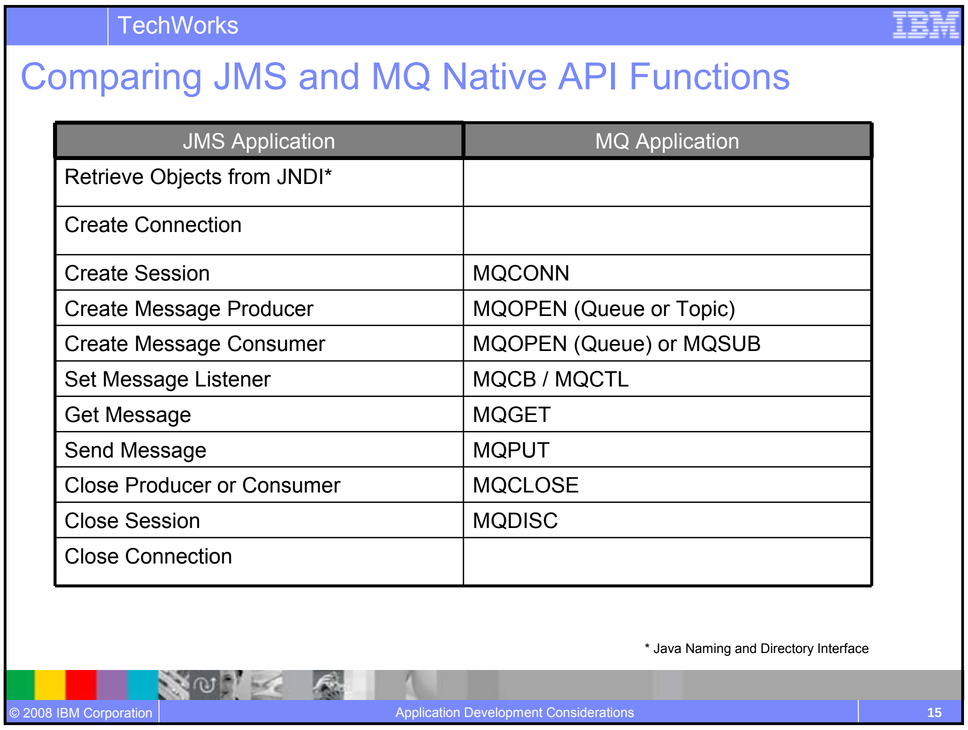
Connection factories are used to create connections!

Connections are used to communicate with the JMS server.

Sessions are used in conjunction with destinations to create messages and message consumers/ producers

* Java Naming and Directory Interface

Application Development Considerations 14



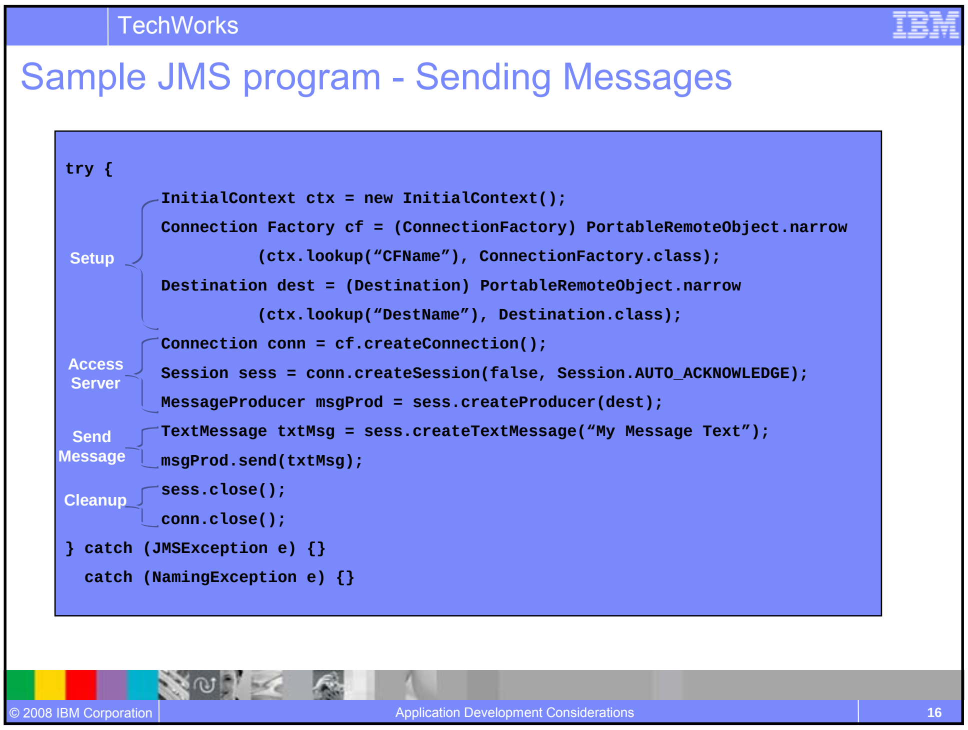
TechWorks

Comparing JMS and MQ Native API Functions

JMS Application	MQ Application
Retrieve Objects from JNDI*	
Create Connection	
Create Session	MQCONN
Create Message Producer	MQOPEN (Queue or Topic)
Create Message Consumer	MQOPEN (Queue) or MQSUB
Set Message Listener	MQCB / MQCTL
Get Message	MQGET
Send Message	MQPUT
Close Producer or Consumer	MQCLOSE
Close Session	MQDISC
Close Connection	

* Java Naming and Directory Interface

© 2008 IBM Corporation | Application Development Considerations | 15

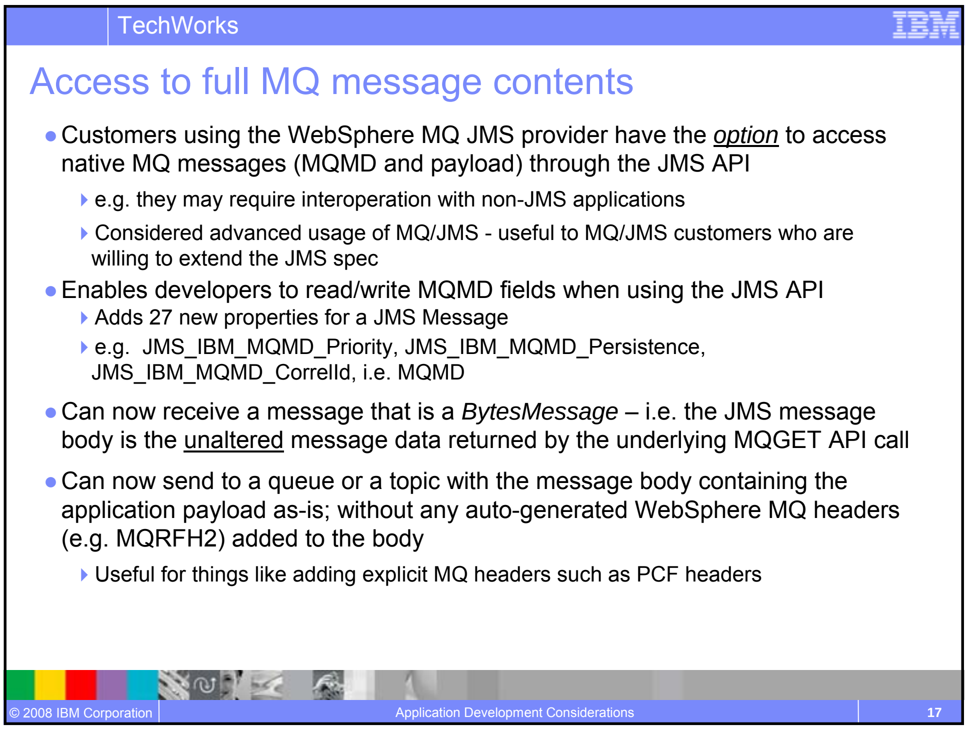


TechWorks

Sample JMS program - Sending Messages

```
try {
            InitialContext ctx = new InitialContext();
            Connection Factory cf = (ConnectionFactory) PortableRemoteObject.narrow
 Setup                (ctx.lookup("CFName"), ConnectionFactory.class);
            Destination dest = (Destination) PortableRemoteObject.narrow
                      (ctx.lookup("DestName"), Destination.class);
            Connection conn = cf.createConnection();
 Access     Session sess = conn.createSession(false, Session.AUTO_ACKNOWLEDGE);
 Server     MessageProducer msgProd = sess.createProducer(dest);
 Send       TextMessage txtMsg = sess.createTextMessage("My Message Text");
 Message    msgProd.send(txtMsg);
            sess.close();
 Cleanup    conn.close();
} catch (JMSException e) {}
  catch (NamingException e) {}
```

© 2008 IBM Corporation | Application Development Considerations | 16

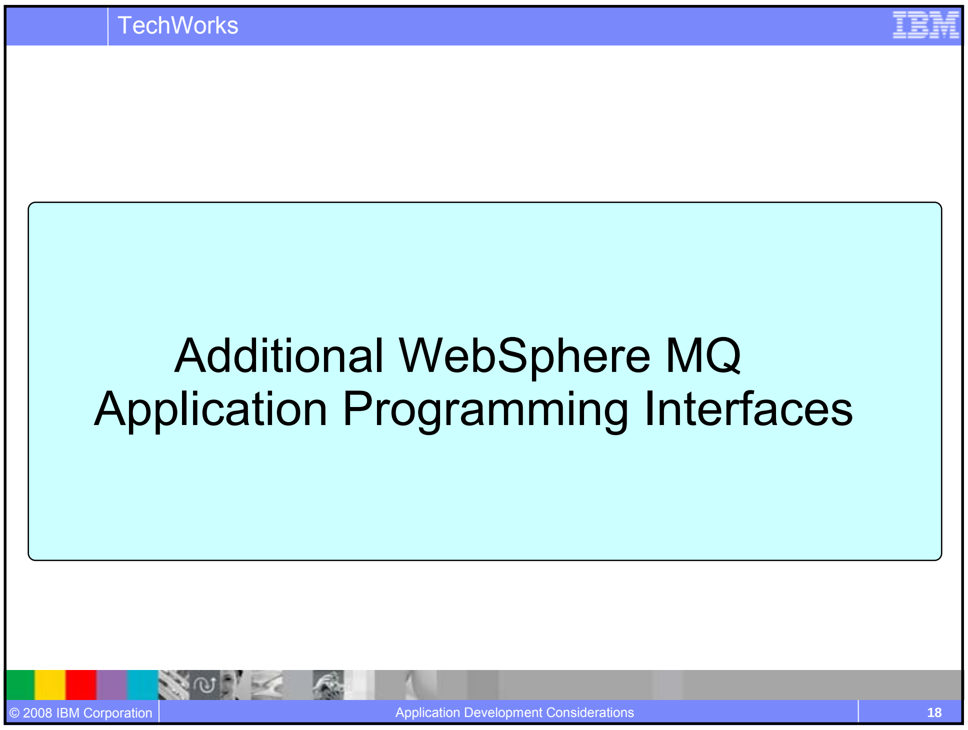


TechWorks

Access to full MQ message contents

- Customers using the WebSphere MQ JMS provider have the *option* to access native MQ messages (MQMD and payload) through the JMS API
  - e.g. they may require interoperation with non-JMS applications
  - Considered advanced usage of MQ/JMS - useful to MQ/JMS customers who are willing to extend the JMS spec
- Enables developers to read/write MQMD fields when using the JMS API
  - Adds 27 new properties for a JMS Message
  - e.g. JMS_IBM_MQMD_Priority, JMS_IBM_MQMD_Persistence, JMS_IBM_MQMD_CorrelId, i.e. MQMD
- Can now receive a message that is a *BytesMessage* – i.e. the JMS message body is the unaltered message data returned by the underlying MQGET API call
- Can now send to a queue or a topic with the message body containing the application payload as-is; without any auto-generated WebSphere MQ headers (e.g. MQRFH2) added to the body
  - Useful for things like adding explicit MQ headers such as PCF headers

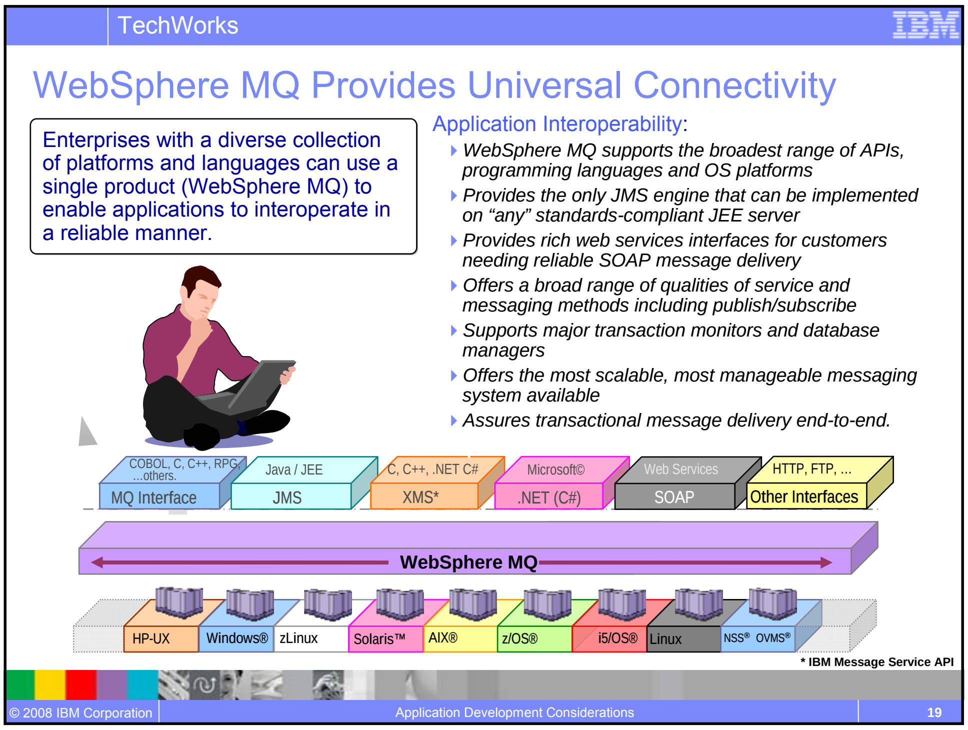
© 2008 IBM Corporation | Application Development Considerations | 17



TechWorks

Additional WebSphere MQ Application Programming Interfaces

© 2008 IBM Corporation | Application Development Considerations | 18



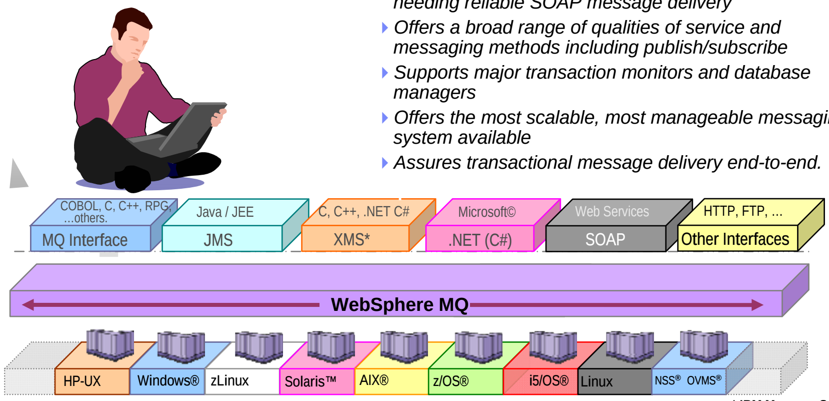
TechWorks

WebSphere MQ Provides Universal Connectivity

Enterprises with a diverse collection of platforms and languages can use a single product (WebSphere MQ) to enable applications to interoperate in a reliable manner.

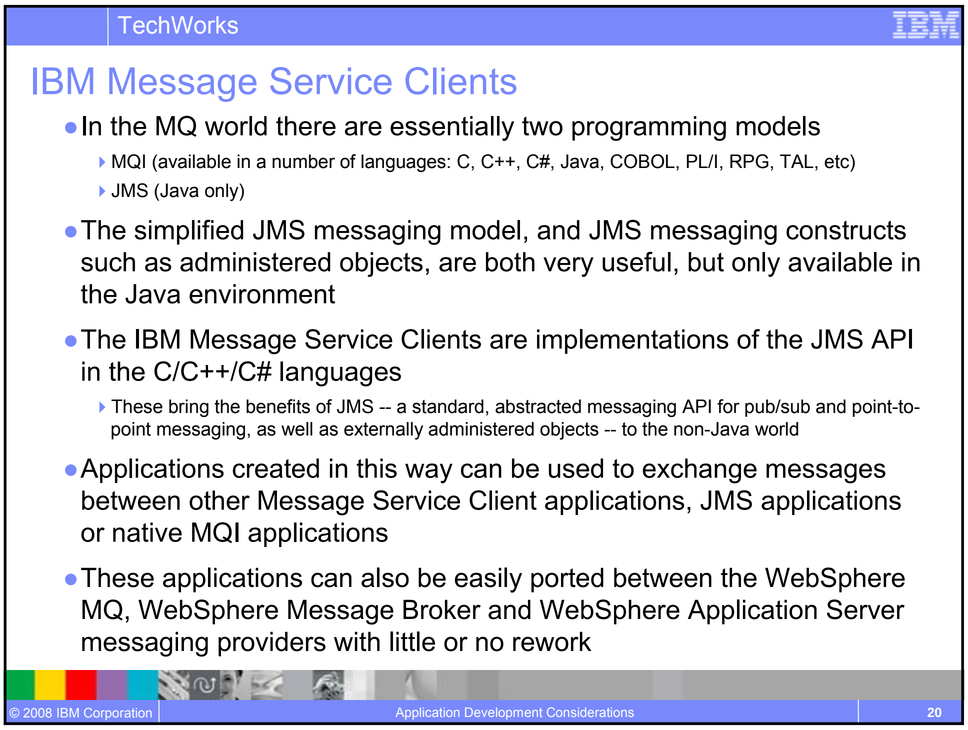
Application Interoperability:

- *WebSphere MQ supports the broadest range of APIs, programming languages and OS platforms*
- *Provides the only JMS engine that can be implemented on "any" standards-compliant JEE server*
- *Provides rich web services interfaces for customers needing reliable SOAP message delivery*
- *Offers a broad range of qualities of service and messaging methods including publish/subscribe*
- *Supports major transaction monitors and database managers*
- *Offers the most scalable, most manageable messaging system available*
- *Assures transactional message delivery end-to-end.*



* IBM Message Service API

© 2008 IBM Corporation | Application Development Considerations | 19

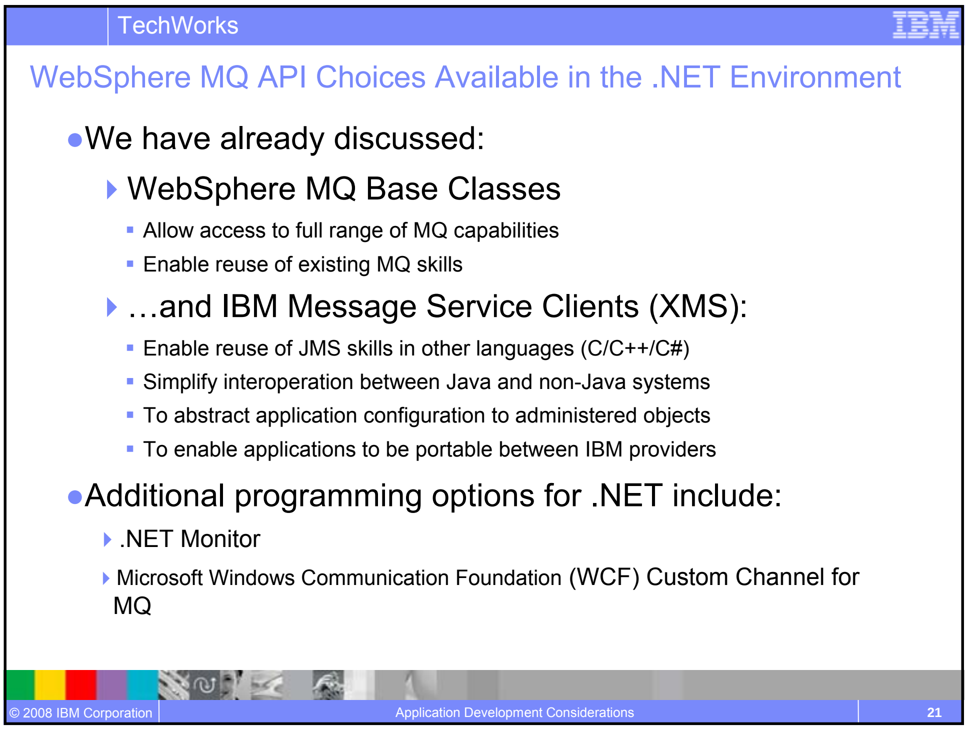


TechWorks

IBM Message Service Clients

- In the MQ world there are essentially two programming models
  - MQI (available in a number of languages: C, C++, C#, Java, COBOL, PL/I, RPG, TAL, etc)
  - JMS (Java only)
- The simplified JMS messaging model, and JMS messaging constructs such as administered objects, are both very useful, but only available in the Java environment
- The IBM Message Service Clients are implementations of the JMS API in the C/C++/C# languages
  - These bring the benefits of JMS -- a standard, abstracted messaging API for pub/sub and point-to-point messaging, as well as externally administered objects -- to the non-Java world
- Applications created in this way can be used to exchange messages between other Message Service Client applications, JMS applications or native MQI applications
- These applications can also be easily ported between the WebSphere MQ, WebSphere Message Broker and WebSphere Application Server messaging providers with little or no rework

© 2008 IBM Corporation | Application Development Considerations | 20

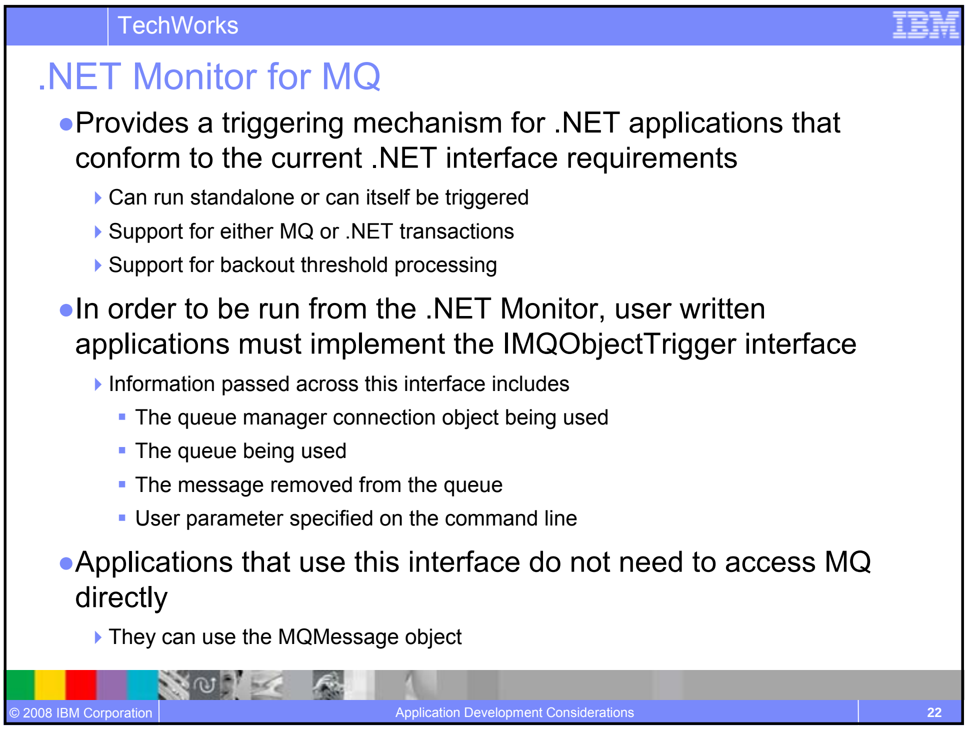


TechWorks

WebSphere MQ API Choices Available in the .NET Environment

- We have already discussed:
  - WebSphere MQ Base Classes
    - Allow access to full range of MQ capabilities
    - Enable reuse of existing MQ skills
  - …and IBM Message Service Clients (XMS):
    - Enable reuse of JMS skills in other languages (C/C++/C#)
    - Simplify interoperation between Java and non-Java systems
    - To abstract application configuration to administered objects
    - To enable applications to be portable between IBM providers
- Additional programming options for .NET include:
  - .NET Monitor
  - Microsoft Windows Communication Foundation (WCF) Custom Channel for MQ

© 2008 IBM Corporation | Application Development Considerations | 21

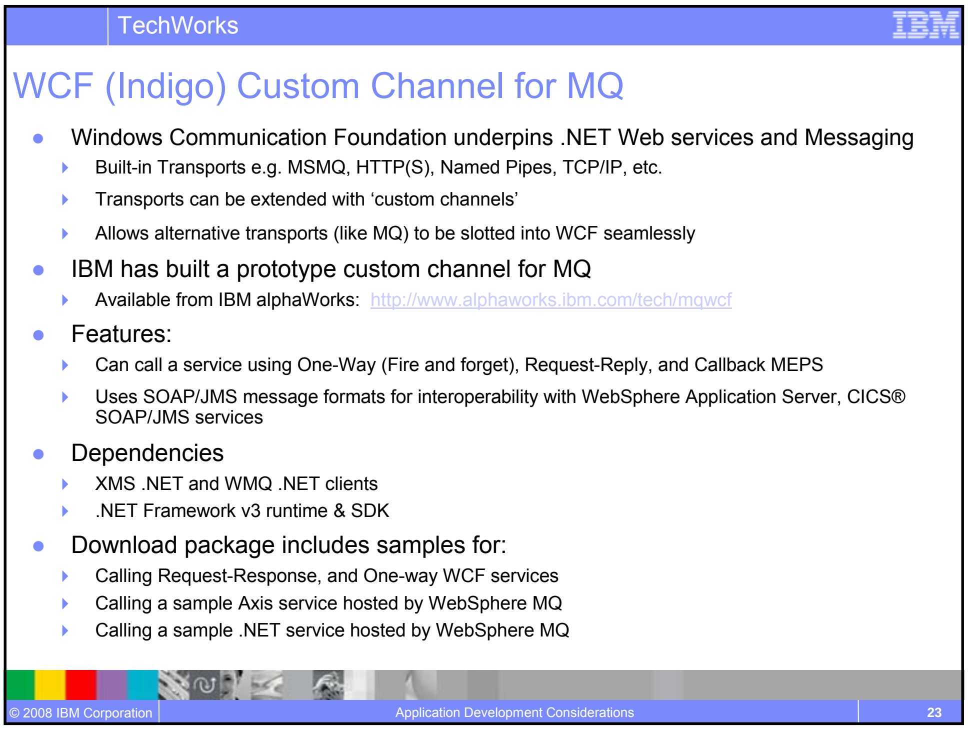


TechWorks

.NET Monitor for MQ

- Provides a triggering mechanism for .NET applications that conform to the current .NET interface requirements
  - Can run standalone or can itself be triggered
  - Support for either MQ or .NET transactions
  - Support for backout threshold processing
- In order to be run from the .NET Monitor, user written applications must implement the IMQObjectTrigger interface
  - Information passed across this interface includes
    - The queue manager connection object being used
    - The queue being used
    - The message removed from the queue
    - User parameter specified on the command line
- Applications that use this interface do not need to access MQ directly
  - They can use the MQMessage object

© 2008 IBM Corporation | Application Development Considerations | 22

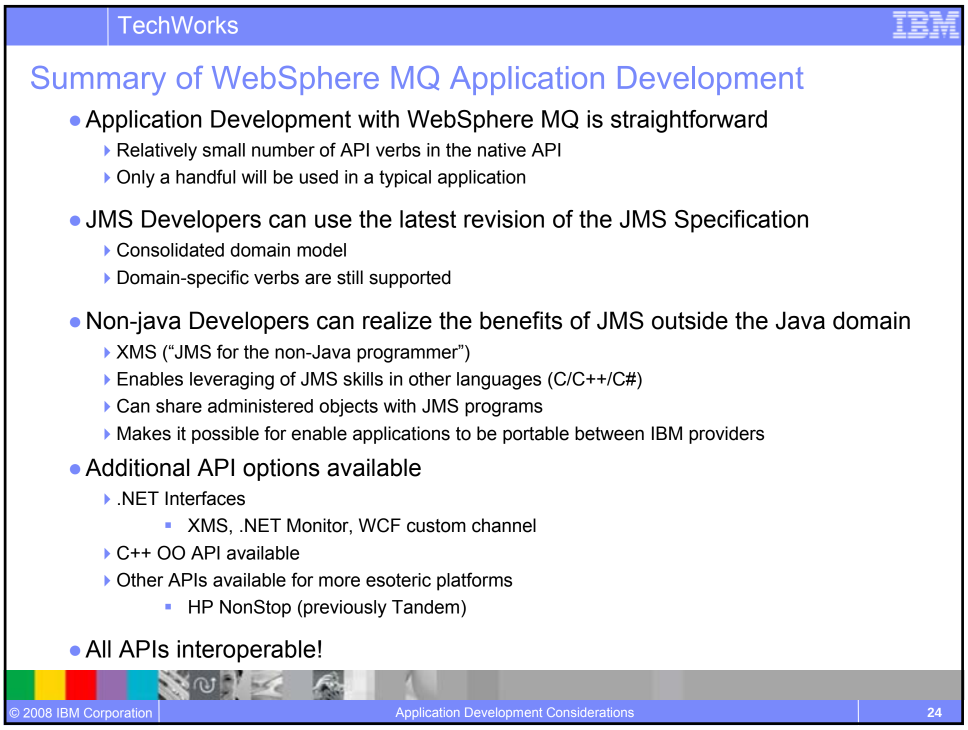


TechWorks

WCF (Indigo) Custom Channel for MQ

- Windows Communication Foundation underpins .NET Web services and Messaging
  - Built-in Transports e.g. MSMQ, HTTP(S), Named Pipes, TCP/IP, etc.
  - Transports can be extended with 'custom channels'
  - Allows alternative transports (like MQ) to be slotted into WCF seamlessly
- IBM has built a prototype custom channel for MQ
  - Available from IBM alphaWorks: http://www.alphaworks.ibm.com/tech/mqwcf
- Features:
  - Can call a service using One-Way (Fire and forget), Request-Reply, and Callback MEPS
  - Uses SOAP/JMS message formats for interoperability with WebSphere Application Server, CICS® SOAP/JMS services
- Dependencies
  - XMS .NET and WMQ .NET clients
  - .NET Framework v3 runtime & SDK
- Download package includes samples for:
  - Calling Request-Response, and One-way WCF services
  - Calling a sample Axis service hosted by WebSphere MQ
  - Calling a sample .NET service hosted by WebSphere MQ

© 2008 IBM Corporation | Application Development Considerations | 23



TechWorks

Summary of WebSphere MQ Application Development

- Application Development with WebSphere MQ is straightforward
  - Relatively small number of API verbs in the native API
  - Only a handful will be used in a typical application
- JMS Developers can use the latest revision of the JMS Specification
  - Consolidated domain model
  - Domain-specific verbs are still supported
- Non-java Developers can realize the benefits of JMS outside the Java domain
  - XMS ("JMS for the non-Java programmer")
  - Enables leveraging of JMS skills in other languages (C/C++/C#)
  - Can share administered objects with JMS programs
  - Makes it possible for enable applications to be portable between IBM providers
- Additional API options available
  - .NET Interfaces
    - XMS, .NET Monitor, WCF custom channel
  - C++ OO API available
  - Other APIs available for more esoteric platforms
    - HP NonStop (previously Tandem)
- All APIs interoperable!

© 2008 IBM Corporation | Application Development Considerations | 24