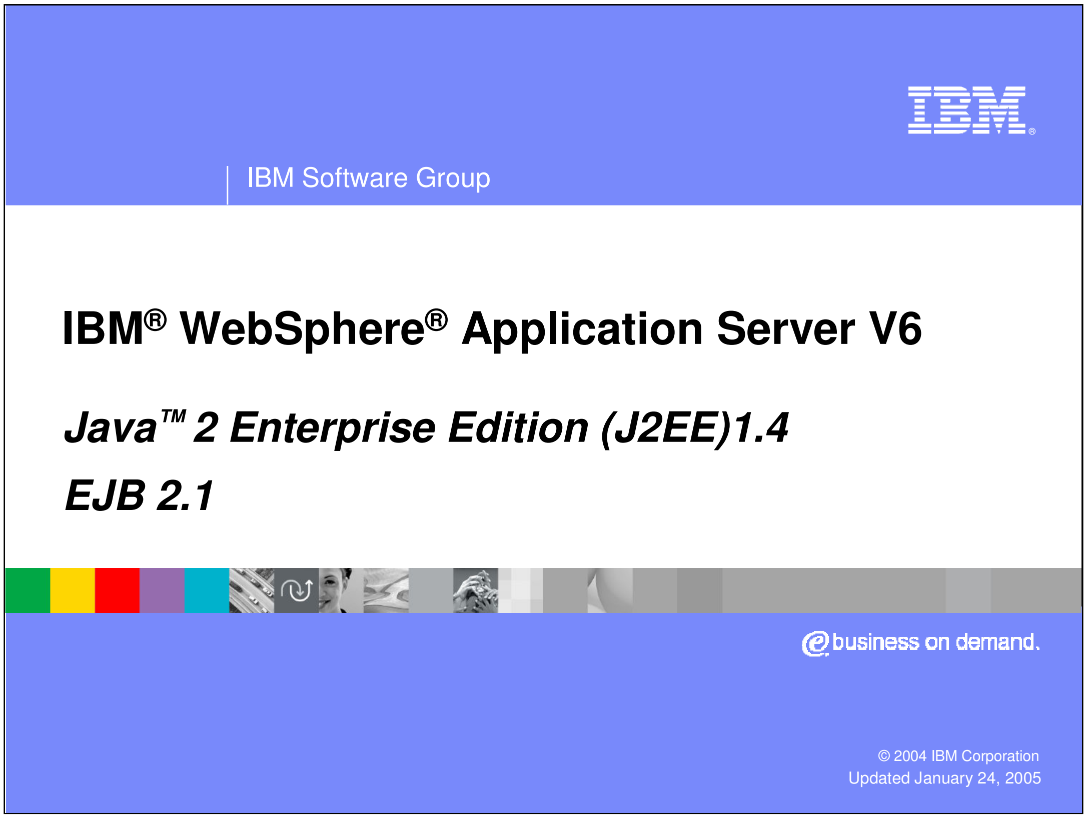


IBM Software Group

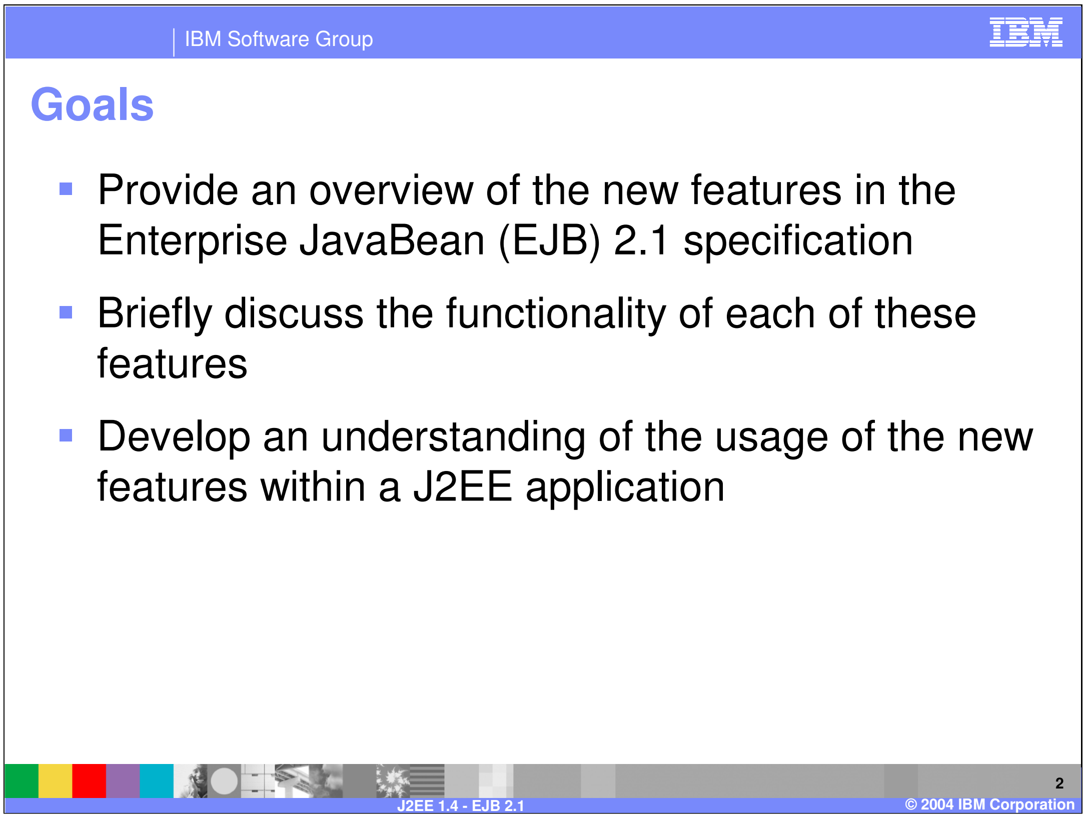
IBM® WebSphere® Application Server V6

Java™ 2 Enterprise Edition (J2EE)1.4

EJB 2.1

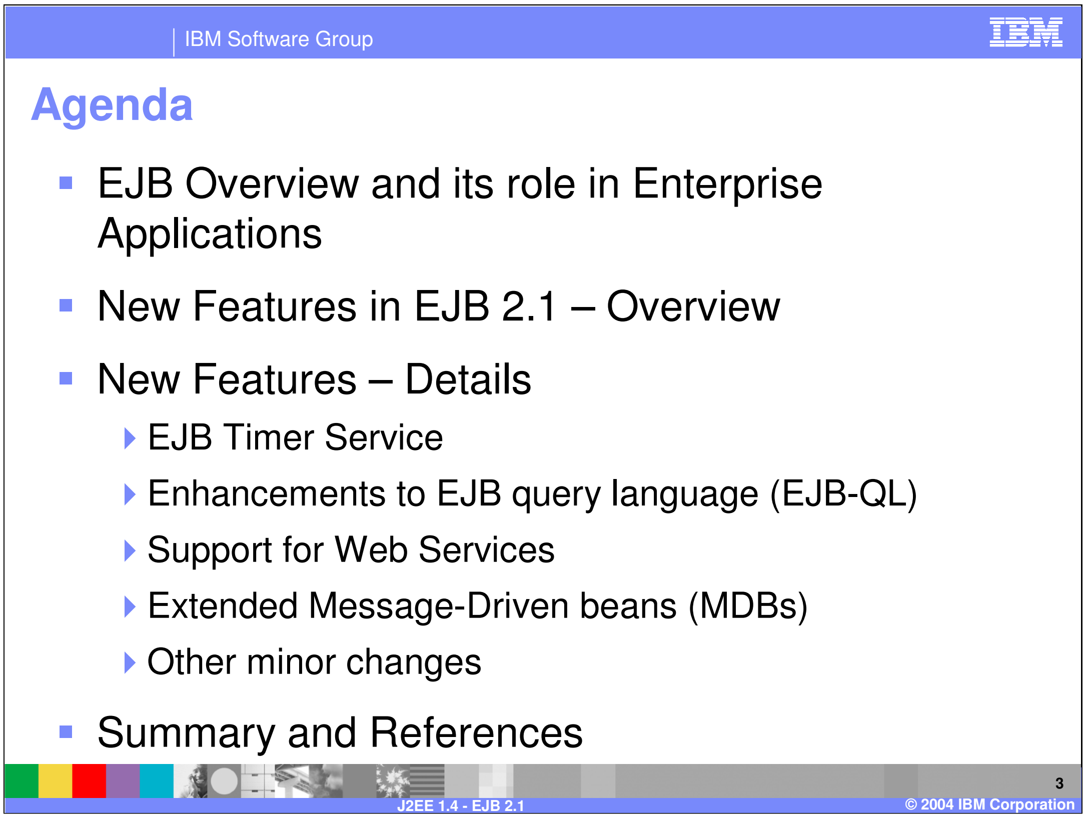
@business on demand.

Updated January 24, 2005



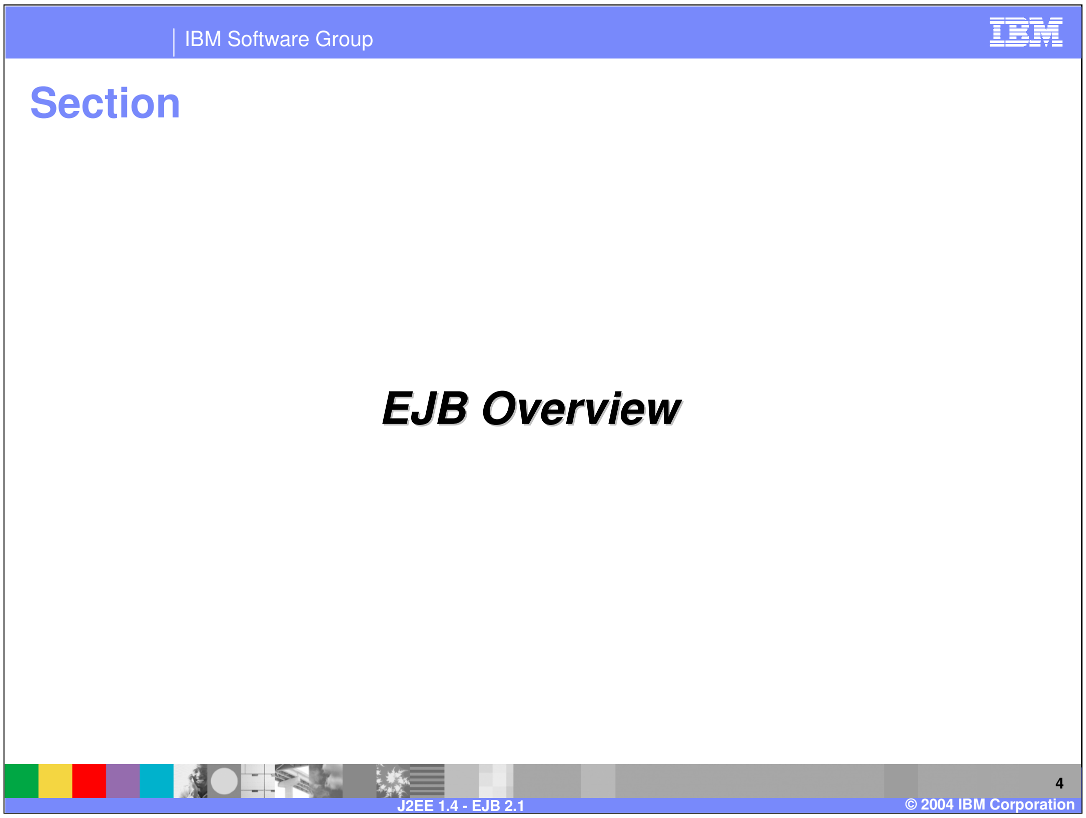
Goals

- Provide an overview of the new features in the Enterprise JavaBean (EJB) 2.1 specification
- Briefly discuss the functionality of each of these features
- Develop an understanding of the usage of the new features within a J2EE application



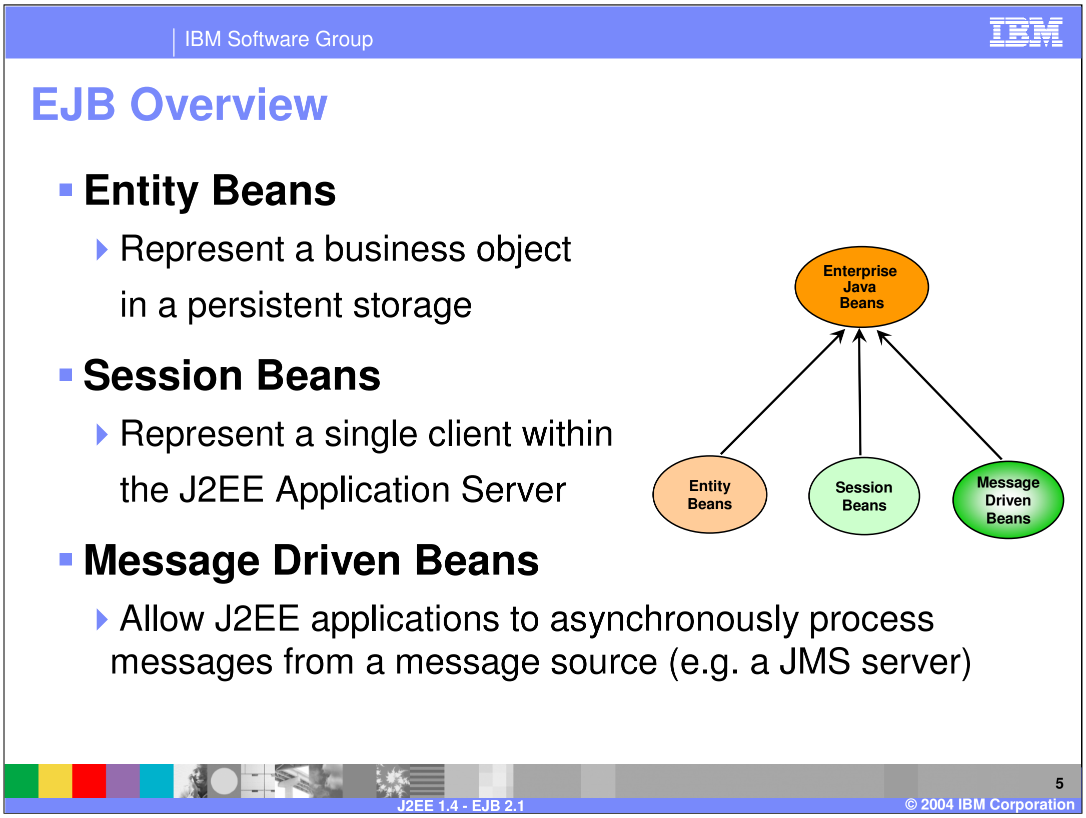
Agenda

- EJB Overview and its role in Enterprise Applications
- New Features in EJB 2.1 – Overview
- New Features – Details
  - EJB Timer Service
  - Enhancements to EJB query language (EJB-QL)
  - Support for Web Services
  - Extended Message-Driven beans (MDBs)
  - Other minor changes
- Summary and References



Section

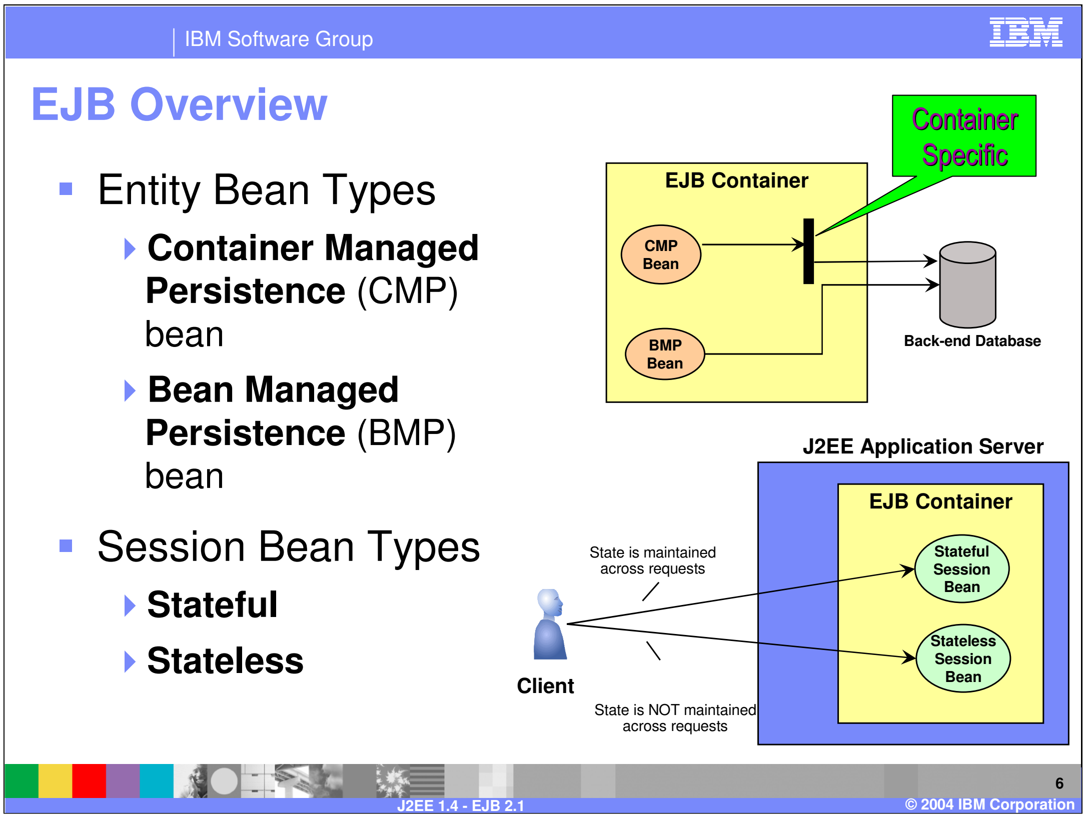
EJB Overview



EJB Overview

- **Entity Beans**
  - Represent a business object in a persistent storage
- **Session Beans**
  - Represent a single client within the J2EE Application Server
- **Message Driven Beans**
  - Allow J2EE applications to asynchronously process messages from a message source (e.g. a JMS server)

Enterprise Java Beans
Entity Beans
Session Beans
Message Driven Beans



IBM Software Group

EJB Overview

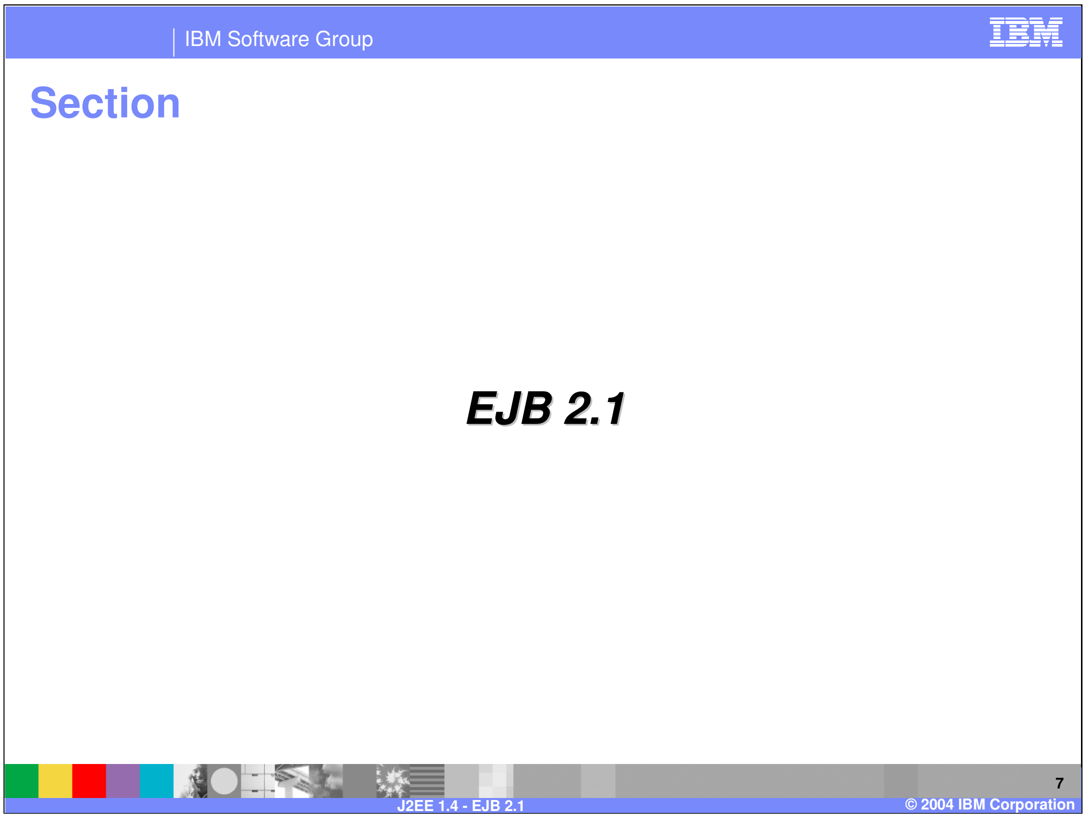
- Entity Bean Types
  - **Container Managed Persistence** (CMP) bean
  - **Bean Managed Persistence** (BMP) bean
- Session Bean Types
  - **Stateful**
  - **Stateless**

J2EE 1.4 - EJB 2.1

6

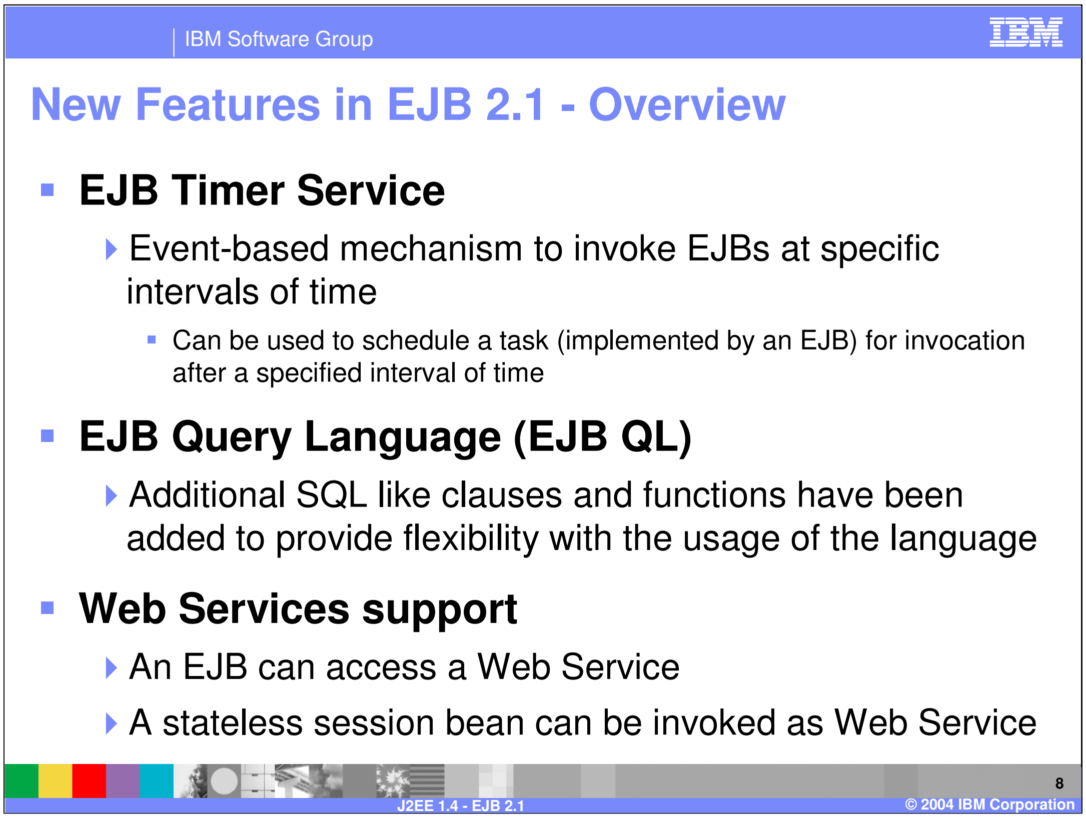
© 2004 IBM Corporation

When you use the forward() method to a servlet, the servlet container changes the target servlet's path environment as if it were the first servlet being invoked. The methods getRequestURI(), getContextPath(), getServletPath(), getPathInfo(), and getQueryString() all return information based on the URI (Uniform Resource Identifier) passed to the getRequestDispatcher() method. However, sometimes an advanced forward() target servlet might like to know the true original request URI. Servlet 2.4 adds five new request attributes to provide extra information during a RequestDispatcher forward() call.



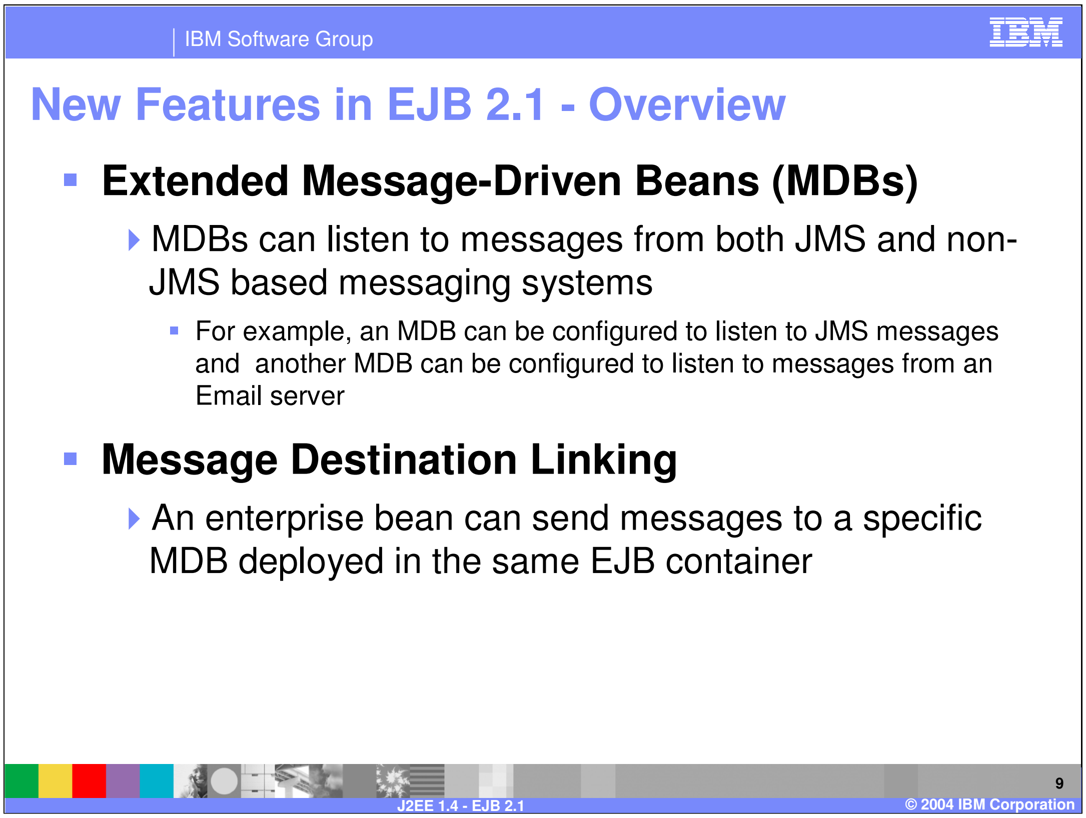
Section

EJB 2.1



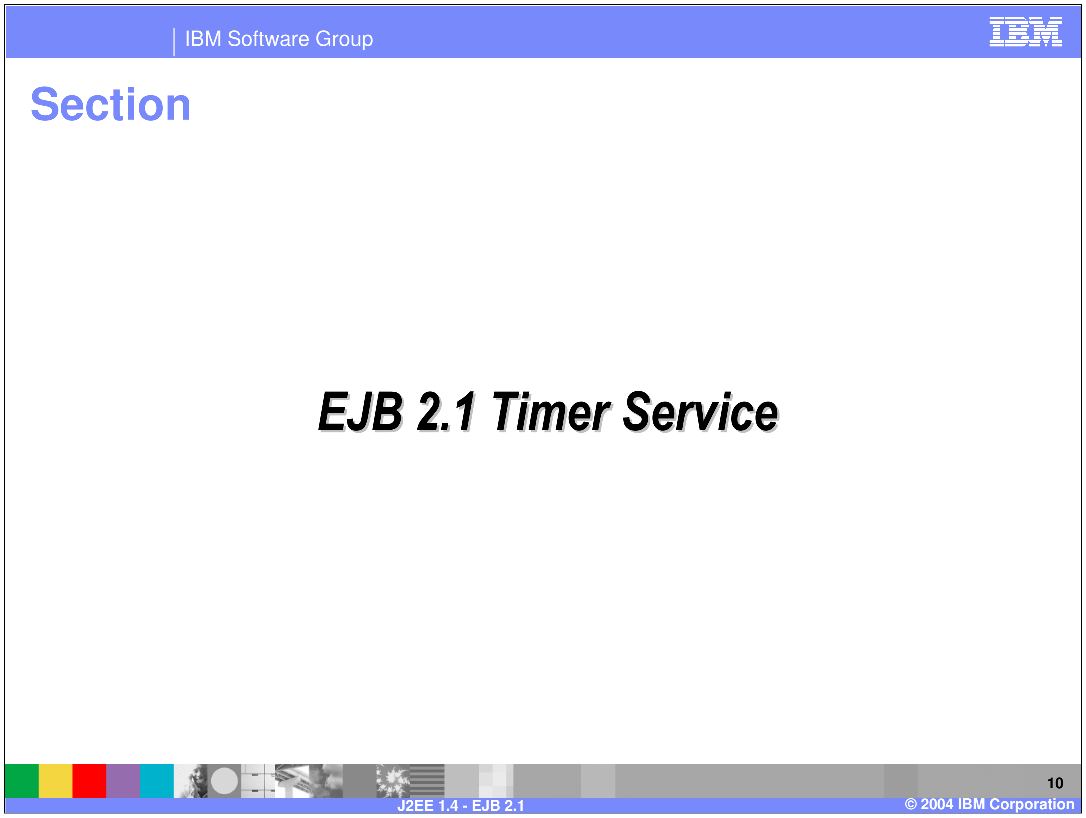
New Features in EJB 2.1 - Overview

- **EJB Timer Service**
  - Event-based mechanism to invoke EJBs at specific intervals of time
    - Can be used to schedule a task (implemented by an EJB) for invocation after a specified interval of time
- **EJB Query Language (EJB QL)**
  - Additional SQL like clauses and functions have been added to provide flexibility with the usage of the language
- **Web Services support**
  - An EJB can access a Web Service
  - A stateless session bean can be invoked as Web Service



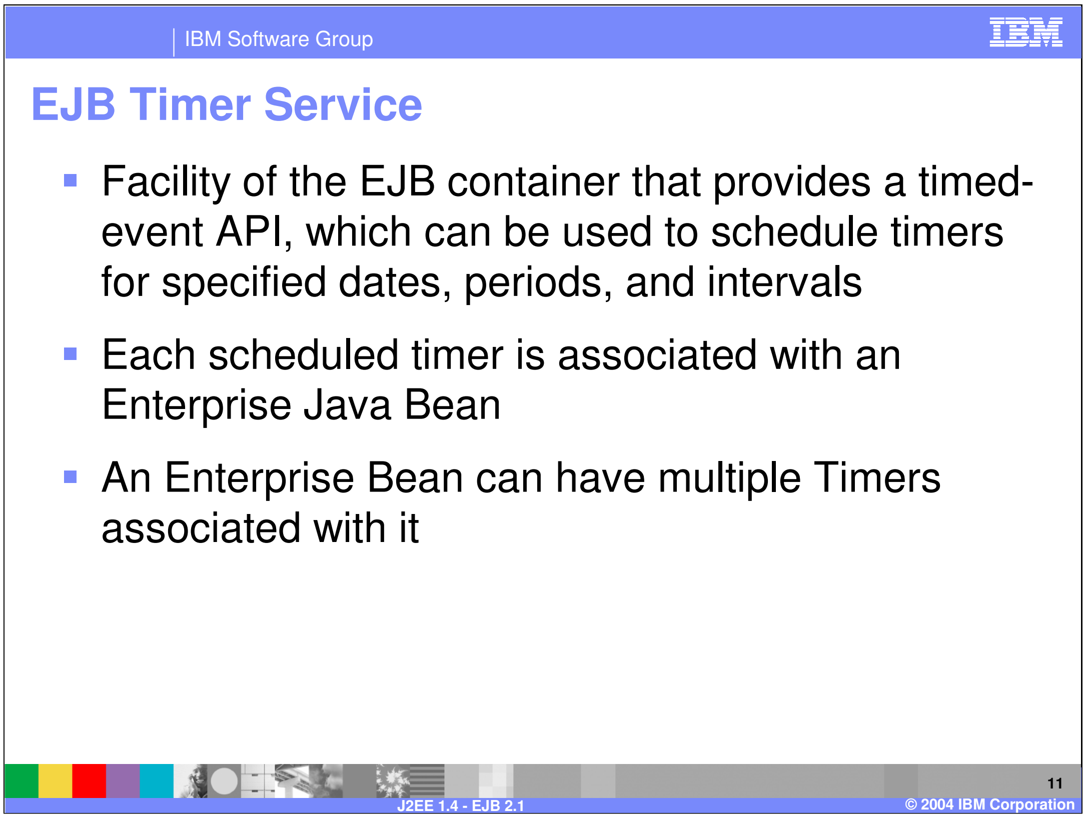
New Features in EJB 2.1 - Overview

- **Extended Message-Driven Beans (MDBs)**
  - MDBs can listen to messages from both JMS and non-JMS based messaging systems
    - For example, an MDB can be configured to listen to JMS messages and another MDB can be configured to listen to messages from an Email server
- **Message Destination Linking**
  - An enterprise bean can send messages to a specific MDB deployed in the same EJB container



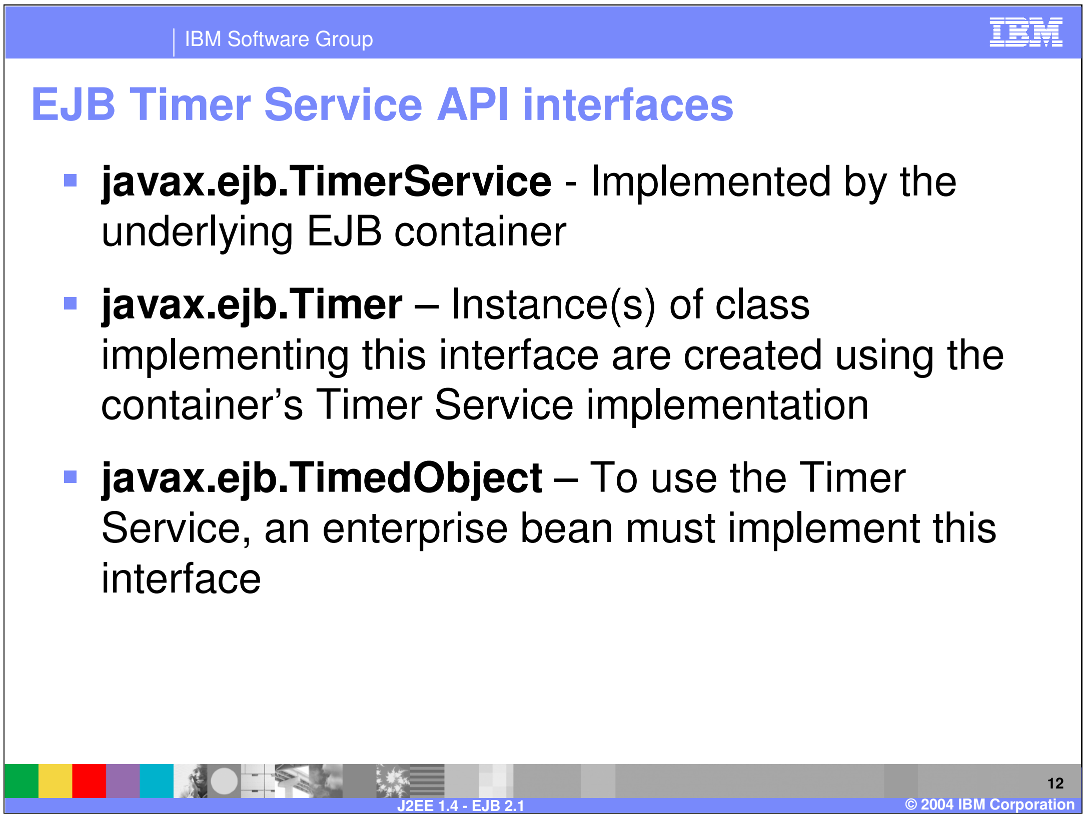
Section

EJB 2.1 Timer Service



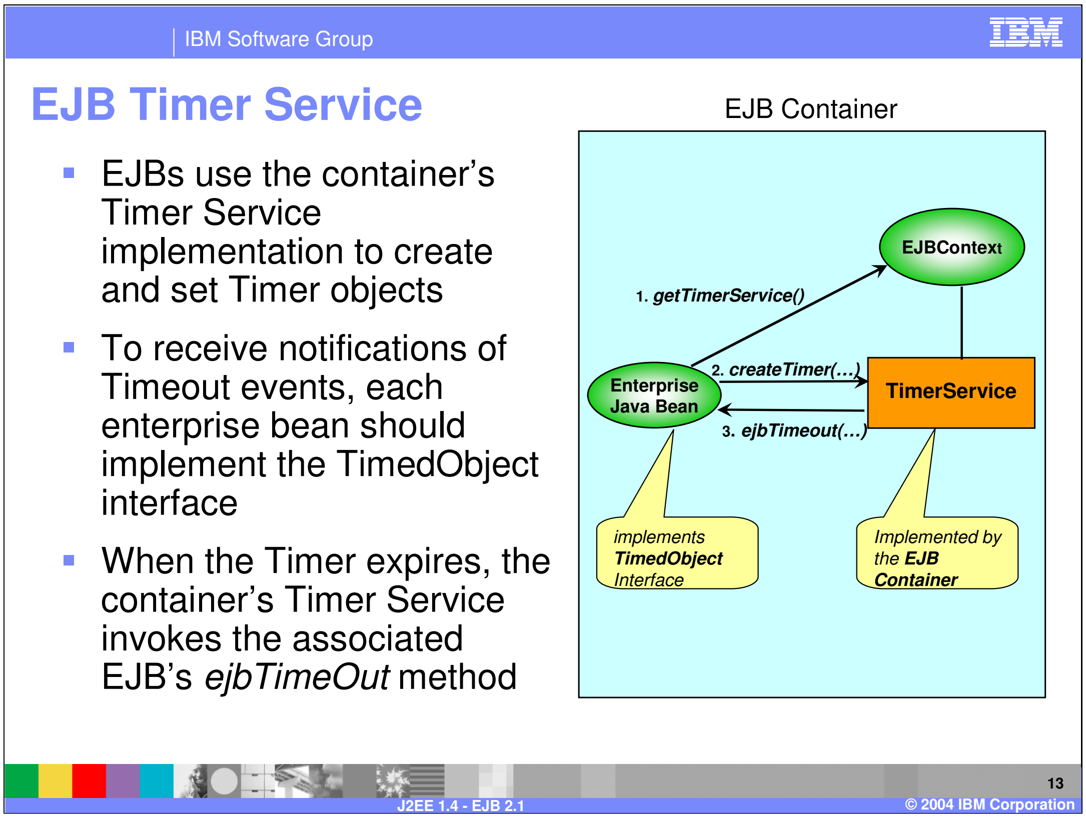
EJB Timer Service

- Facility of the EJB container that provides a timed-event API, which can be used to schedule timers for specified dates, periods, and intervals
- Each scheduled timer is associated with an Enterprise Java Bean
- An Enterprise Bean can have multiple Timers associated with it



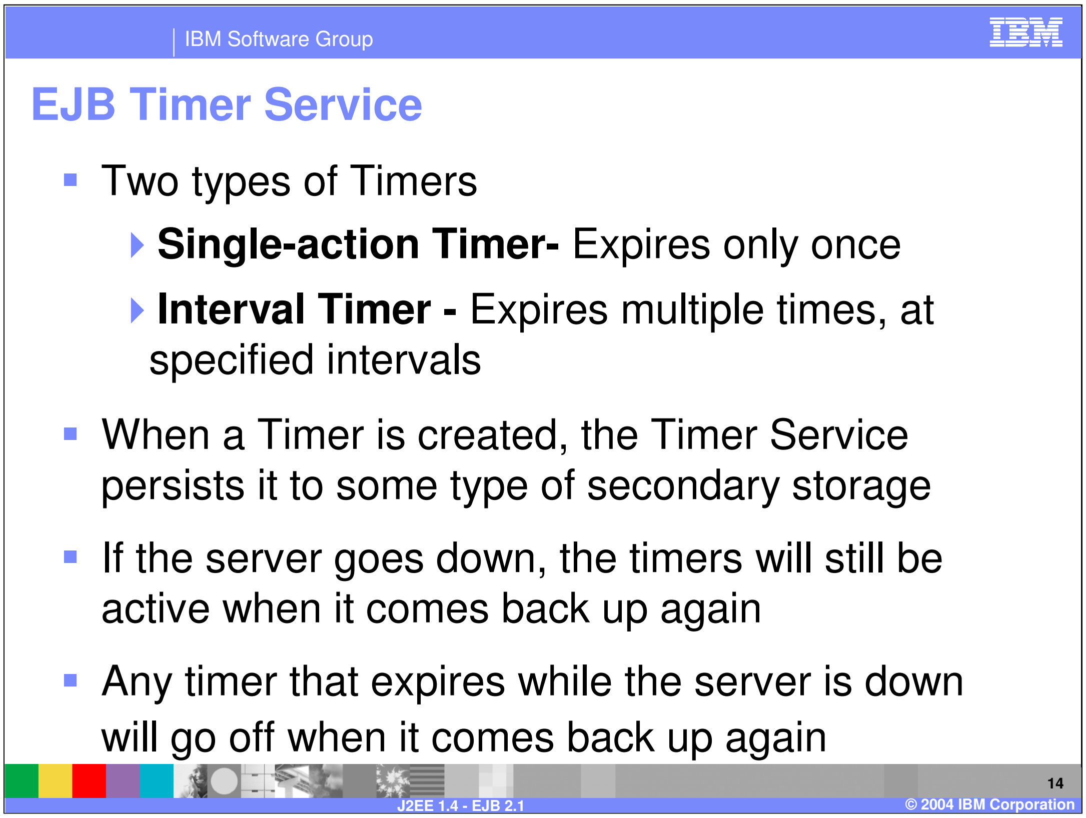
EJB Timer Service API interfaces

- **javax.ejb.TimerService** - Implemented by the underlying EJB container
- **javax.ejb.Timer** – Instance(s) of class implementing this interface are created using the container’s Timer Service implementation
- **javax.ejb.TimedObject** – To use the Timer Service, an enterprise bean must implement this interface



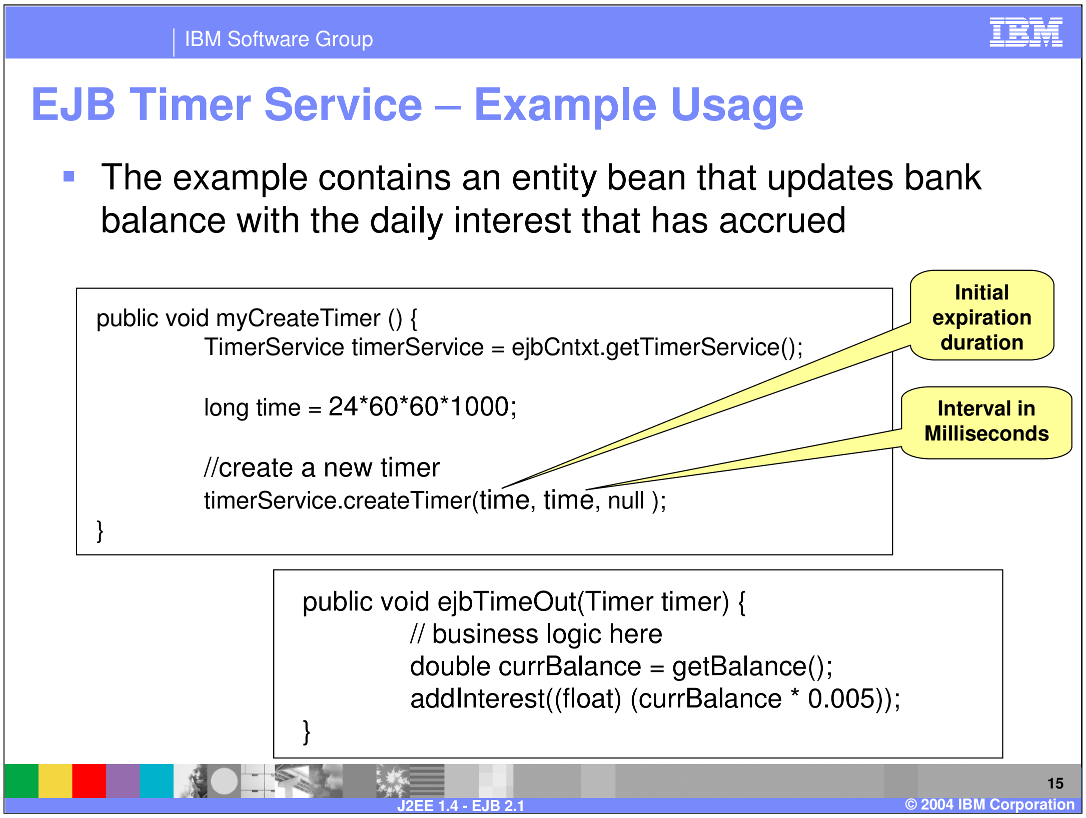
EJB Timer Service

- EJBs use the container's Timer Service implementation to create and set Timer objects
- To receive notifications of Timeout events, each enterprise bean should implement the TimedObject interface
- When the Timer expires, the container's Timer Service invokes the associated EJB's *ejbTimeOut* method



EJB Timer Service

- Two types of Timers
  - **Single-action Timer-** Expires only once
  - **Interval Timer -** Expires multiple times, at specified intervals
- When a Timer is created, the Timer Service persists it to some type of secondary storage
- If the server goes down, the timers will still be active when it comes back up again
- Any timer that expires while the server is down will go off when it comes back up again



EJB Timer Service – Example Usage

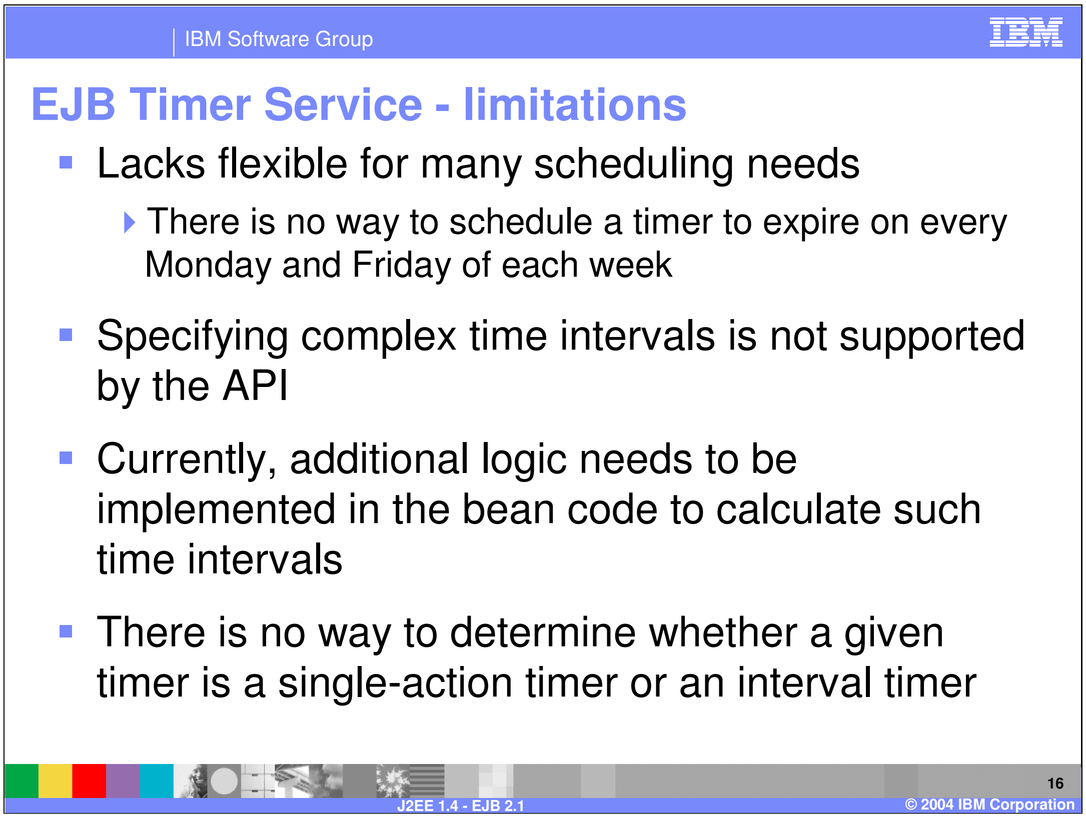
- The example contains an entity bean that updates bank balance with the daily interest that has accrued

```
public void myCreateTimer () {
		TimerService timerService = ejbCntxt.getTimerService();

		long time = 24*60*60*1000;

		//create a new timer
		timerService.createTimer(time, time, null );
}
```

Initial expiration duration

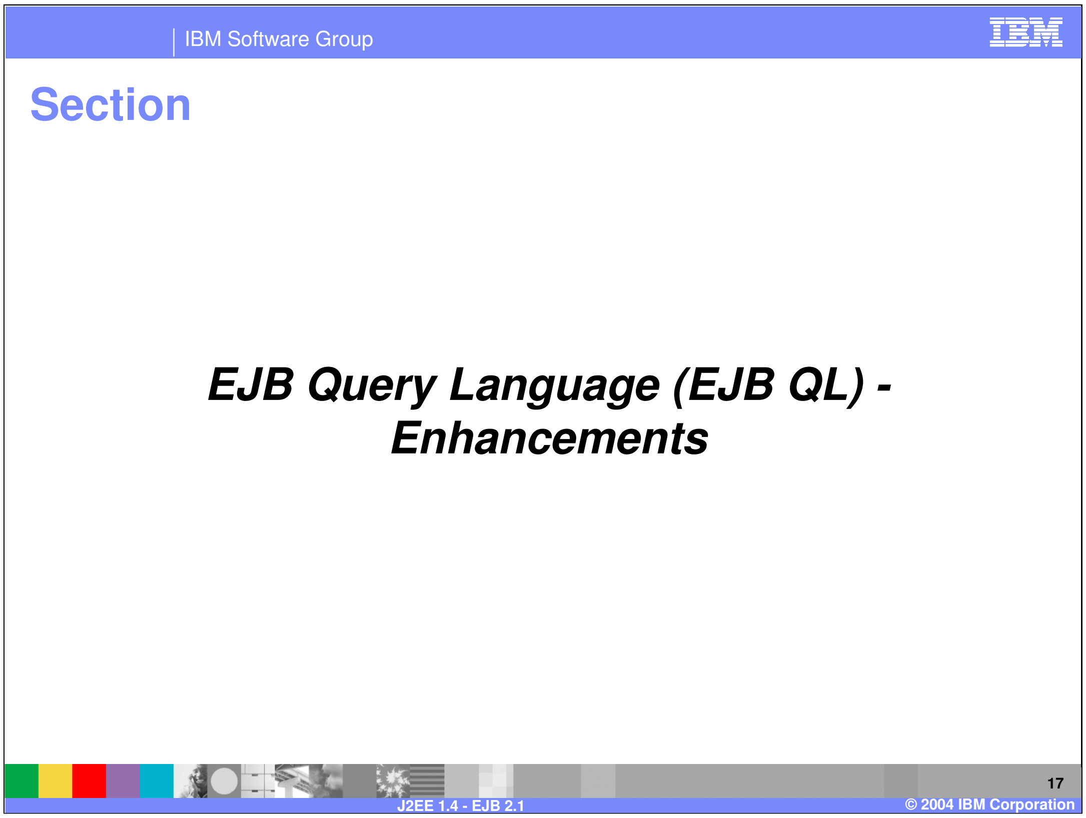
Interval in Milliseconds

```
public void ejbTimeOut(Timer timer) {
		// business logic here
		double currBalance = getBalance();
		addInterest((float) (currBalance * 0.005));
}
```



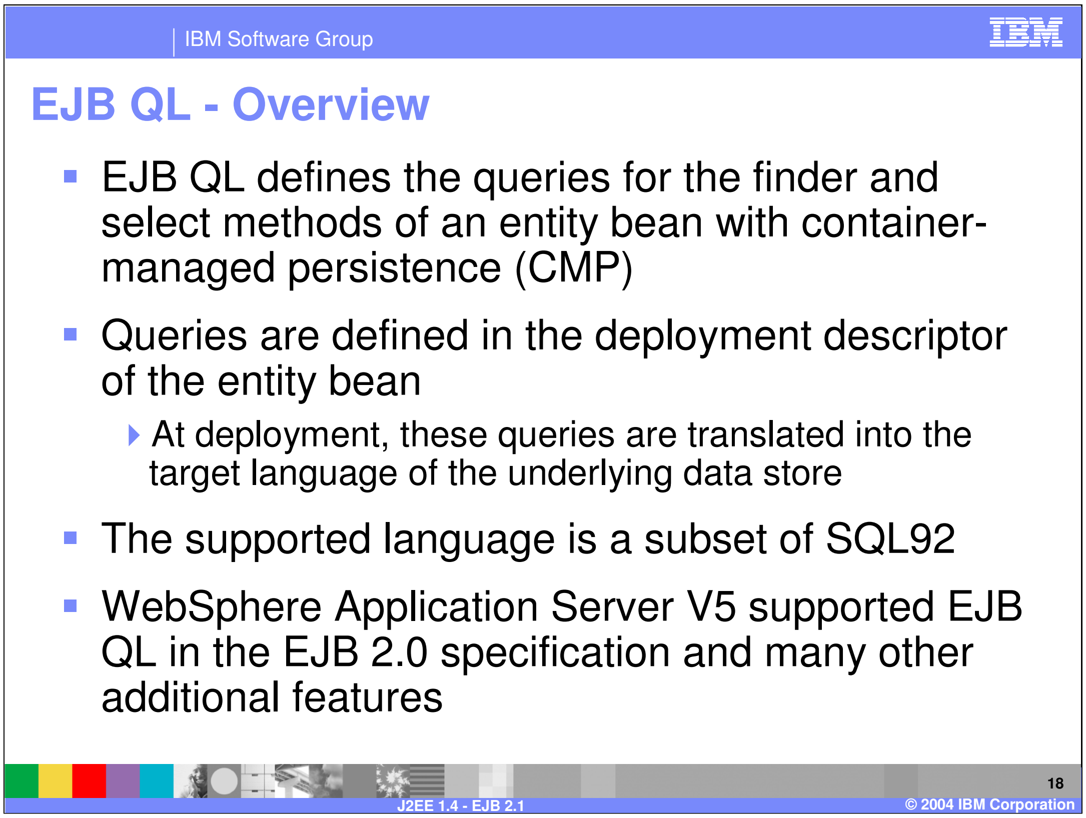
EJB Timer Service - limitations

- Lacks flexible for many scheduling needs
  - There is no way to schedule a timer to expire on every Monday and Friday of each week
- Specifying complex time intervals is not supported by the API
- Currently, additional logic needs to be implemented in the bean code to calculate such time intervals
- There is no way to determine whether a given timer is a single-action timer or an interval timer



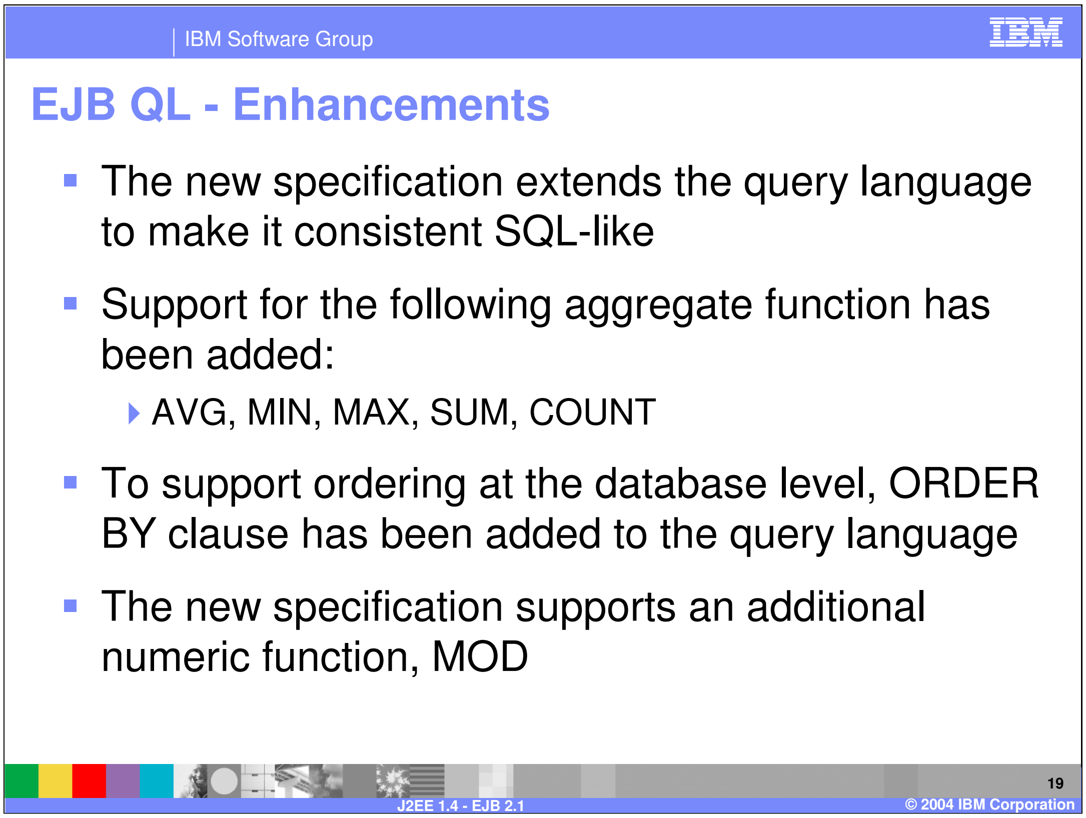
Section

EJB Query Language (EJB QL) - Enhancements



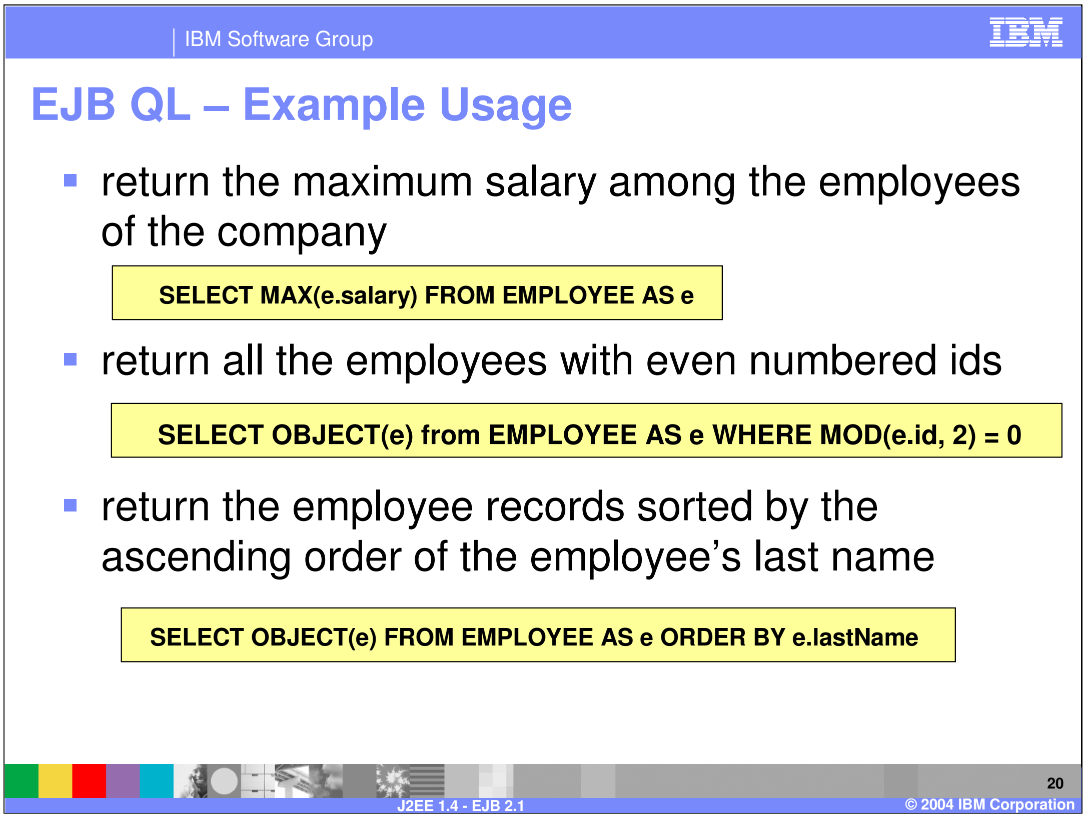
EJB QL - Overview

- EJB QL defines the queries for the finder and select methods of an entity bean with container-managed persistence (CMP)
- Queries are defined in the deployment descriptor of the entity bean
  - At deployment, these queries are translated into the target language of the underlying data store
- The supported language is a subset of SQL92
- WebSphere Application Server V5 supported EJB QL in the EJB 2.0 specification and many other additional features



EJB QL - Enhancements

- The new specification extends the query language to make it consistent SQL-like
- Support for the following aggregate function has been added:
  - AVG, MIN, MAX, SUM, COUNT
- To support ordering at the database level, ORDER BY clause has been added to the query language
- The new specification supports an additional numeric function, MOD



EJB QL – Example Usage

- return the maximum salary among the employees of the company

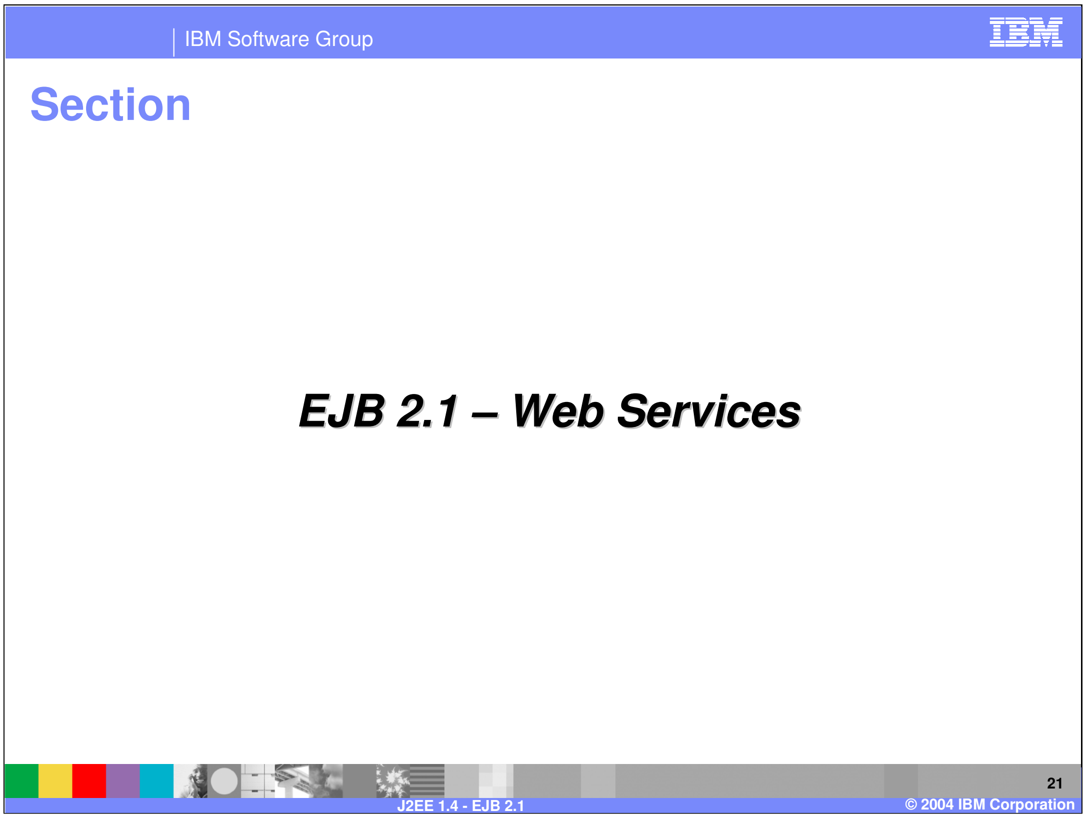
```
SELECT MAX(e.salary) FROM EMPLOYEE AS e
```

- return all the employees with even numbered ids

```
SELECT OBJECT(e) from EMPLOYEE AS e WHERE MOD(e.id, 2) = 0
```

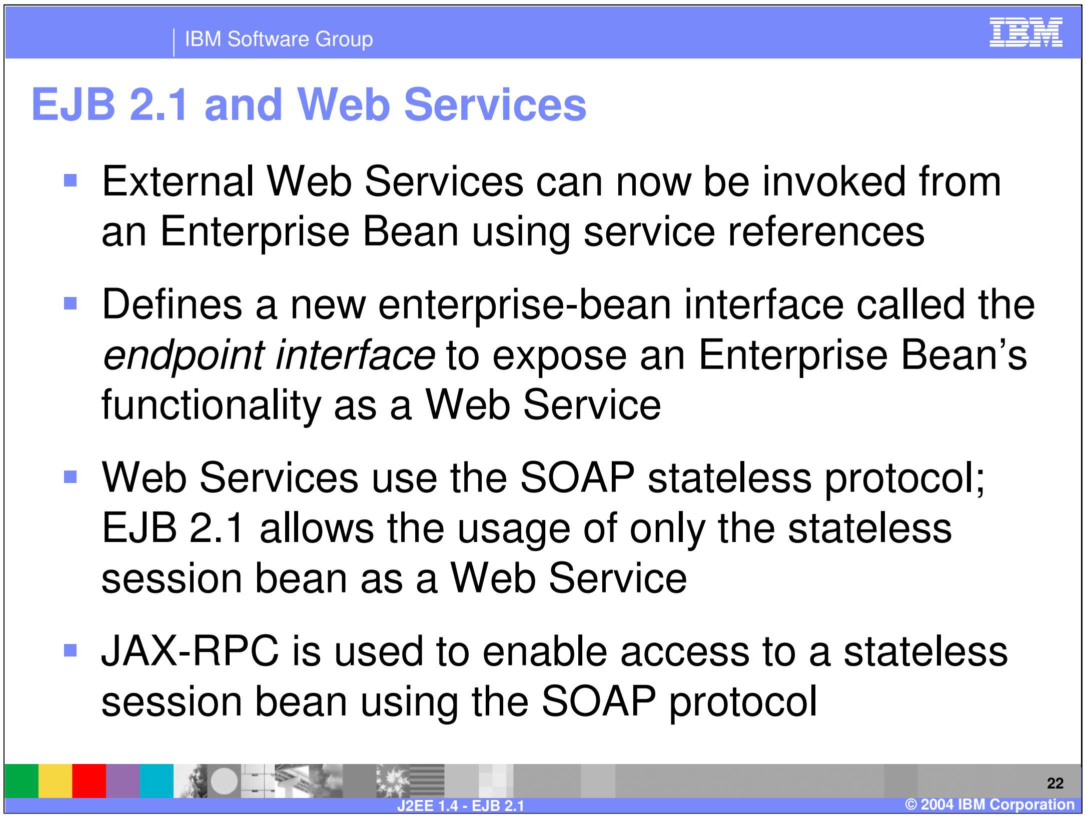
- return the employee records sorted by the ascending order of the employee’s last name

```
SELECT OBJECT(e) FROM EMPLOYEE AS e ORDER BY e.lastName
```



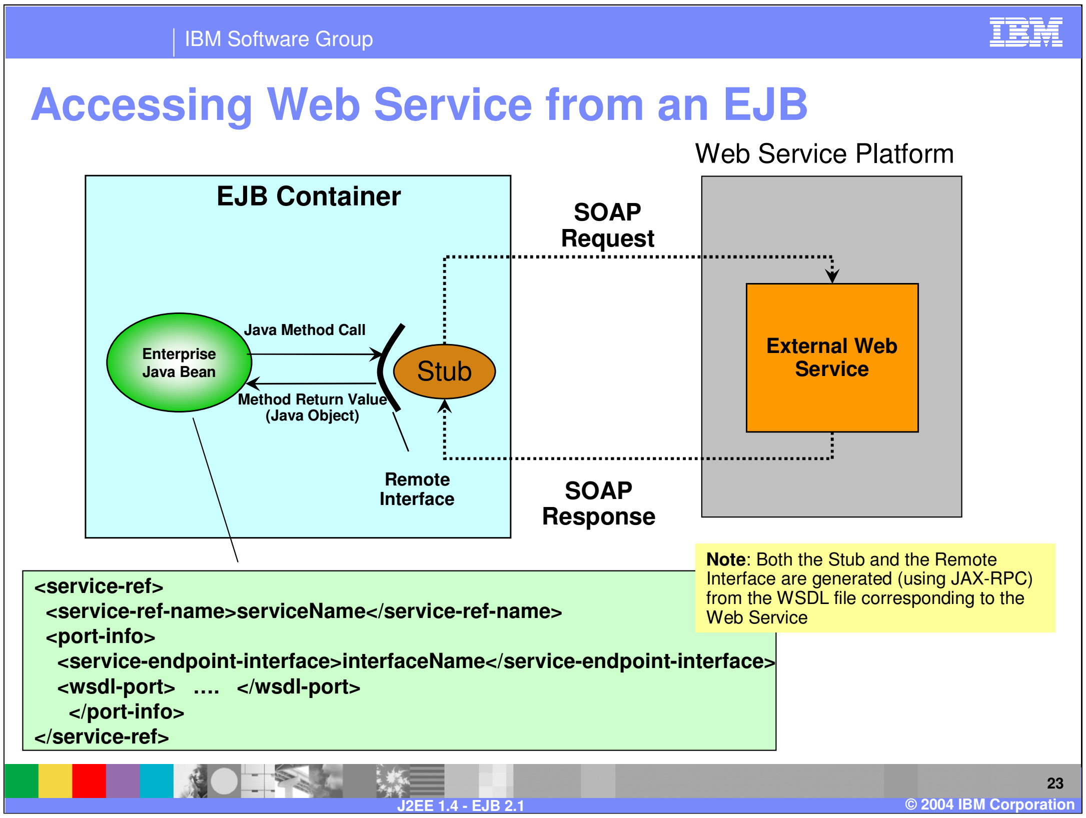
Section

EJB 2.1 – Web Services



EJB 2.1 and Web Services

- External Web Services can now be invoked from an Enterprise Bean using service references
- Defines a new enterprise-bean interface called the *endpoint interface* to expose an Enterprise Bean’s functionality as a Web Service
- Web Services use the SOAP stateless protocol; EJB 2.1 allows the usage of only the stateless session bean as a Web Service
- JAX-RPC is used to enable access to a stateless session bean using the SOAP protocol

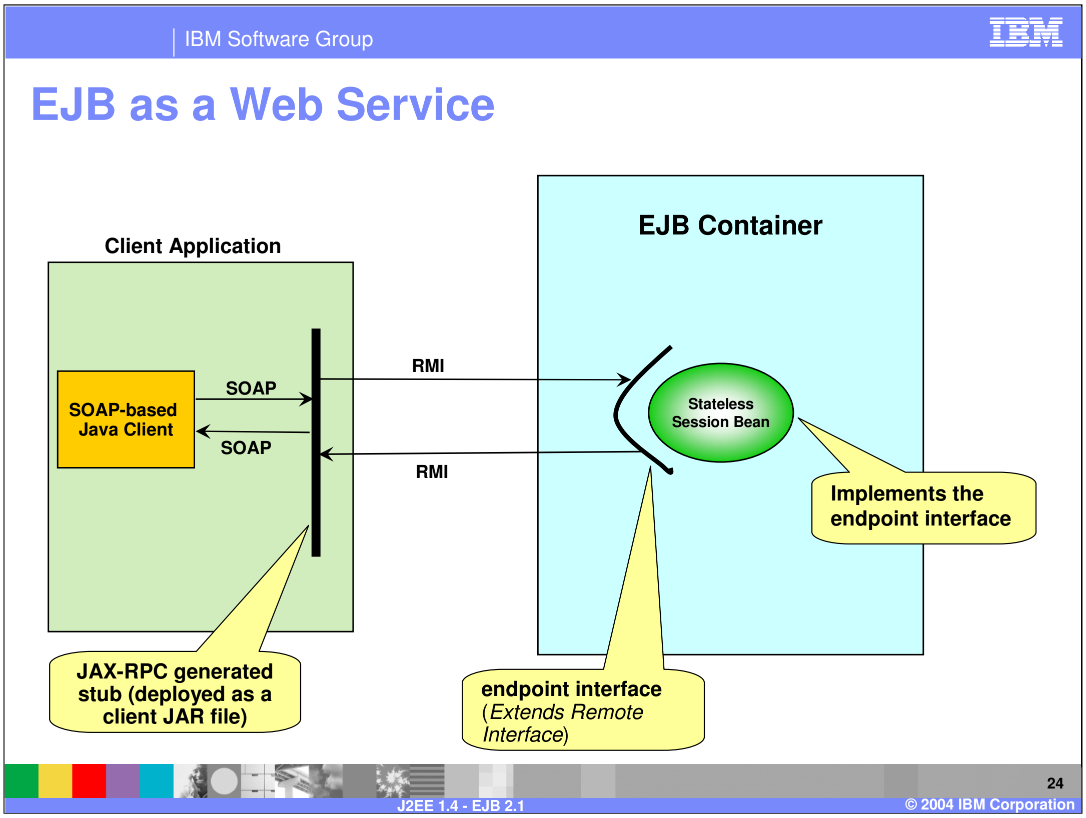


Accessing Web Service from an EJB

```
<service-ref>
  <service-ref-name>serviceName</service-ref-name>
  <port-info>
    <service-endpoint-interface>interfaceName</service-endpoint-interface>
    <wsdl-port>   ….   </wsdl-port>
      </port-info>
</service-ref>
```

Note: Both the Stub and the Remote Interface are generated (using JAX-RPC) from the WSDL file corresponding to the Web Service

Once the interface and stub have been generated and bound to the JNDI ENC, they can be used at run time to invoke operations on the Web service



EJB as a Web Service

Client Application

SOAP-based Java Client

SOAP

SOAP

RMI

RMI

EJB Container

Stateless Session Bean

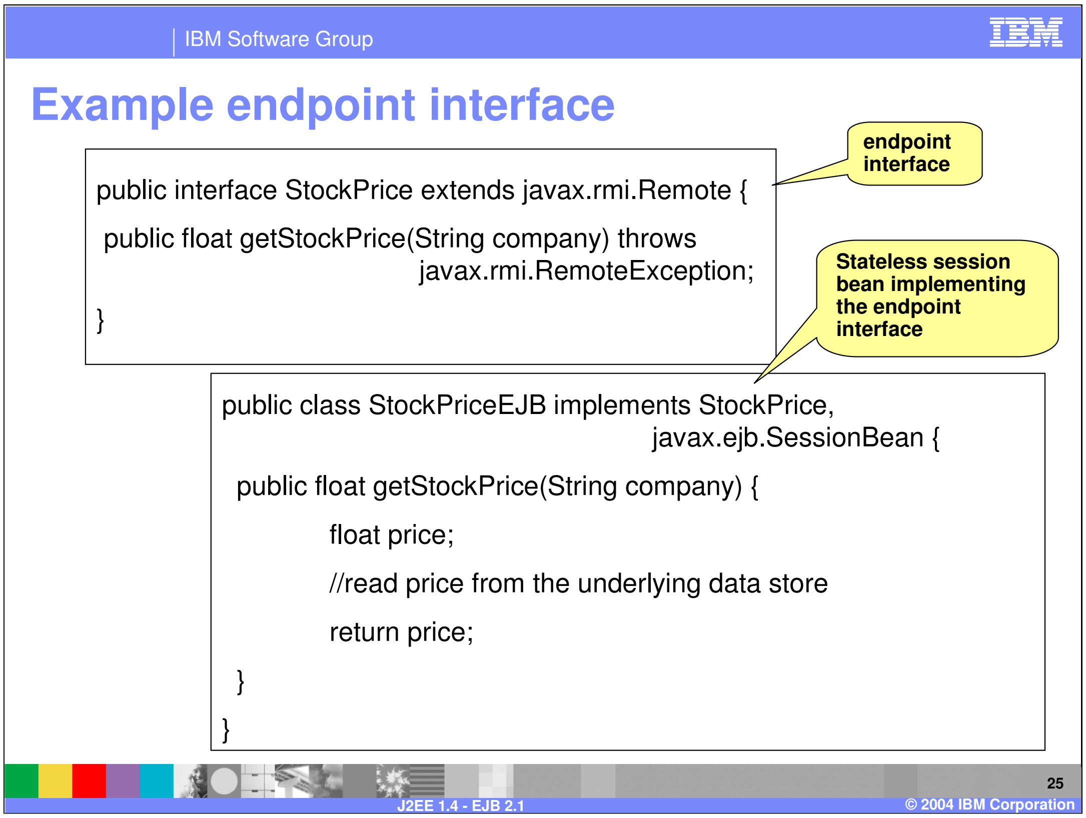
Implements the endpoint interface

JAX-RPC generated stub (deployed as a client JAR file)

endpoint interface
(*Extends Remote Interface*)

•After defining the endpoint interface of the stateless session bean, a stub is generated by the EJB Container's JAX-RPC implementation

•The generated stub is packaged into a client JAR that can be used by external SOAP-based Java applications to access the EJB methods



Example endpoint interface

```
public interface StockPrice extends javax.rmi.Remote {
 public float getStockPrice(String company) throws
                                   javax.rmi.RemoteException;
}
```

endpoint interface

Stateless session bean implementing the endpoint interface

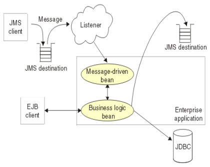
```
public class StockPriceEJB implements StockPrice,
                                   javax.ejb.SessionBean {
  public float getStockPrice(String company) {
          float price;
          //read price from the underlying data store
          return price;
  }
}
```

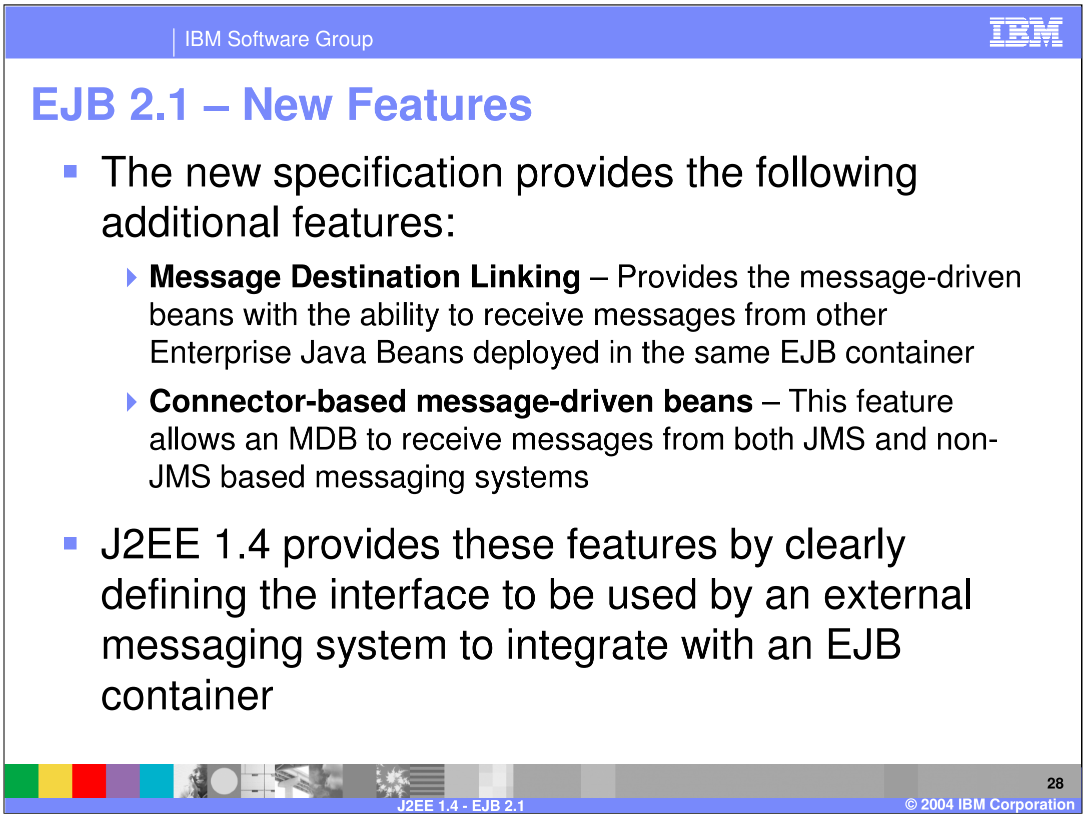
Section

Extended Message Driven Beans (MDBs)

MDB - Overview

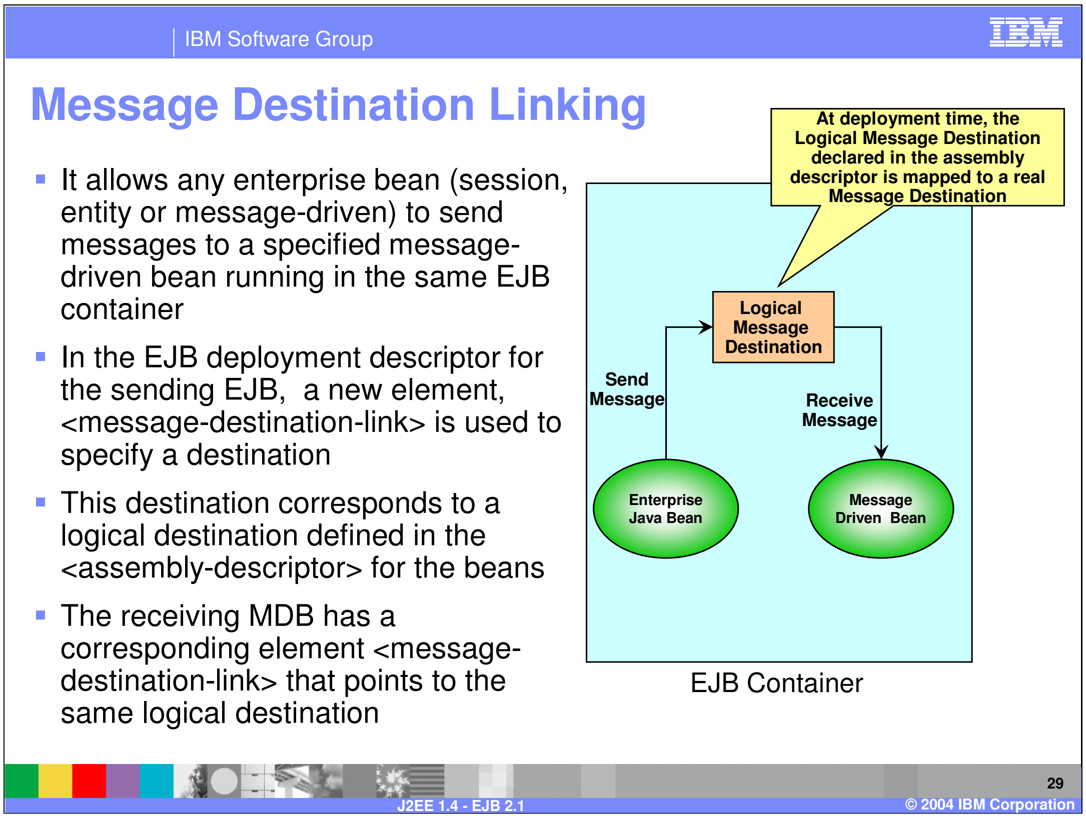
- MDB is an enterprise bean that allows J2EE applications to asynchronously process messages
- As of EJB 2.0 specification, MDBs can receive messages only from JMS-based messaging systems
- An MDB can be configured to listen to messages from either a Queue or a Topic





EJB 2.1 – New Features

- The new specification provides the following additional features:
  - **Message Destination Linking** – Provides the message-driven beans with the ability to receive messages from other Enterprise Java Beans deployed in the same EJB container
  - **Connector-based message-driven beans** – This feature allows an MDB to receive messages from both JMS and non-JMS based messaging systems
- J2EE 1.4 provides these features by clearly defining the interface to be used by an external messaging system to integrate with an EJB container

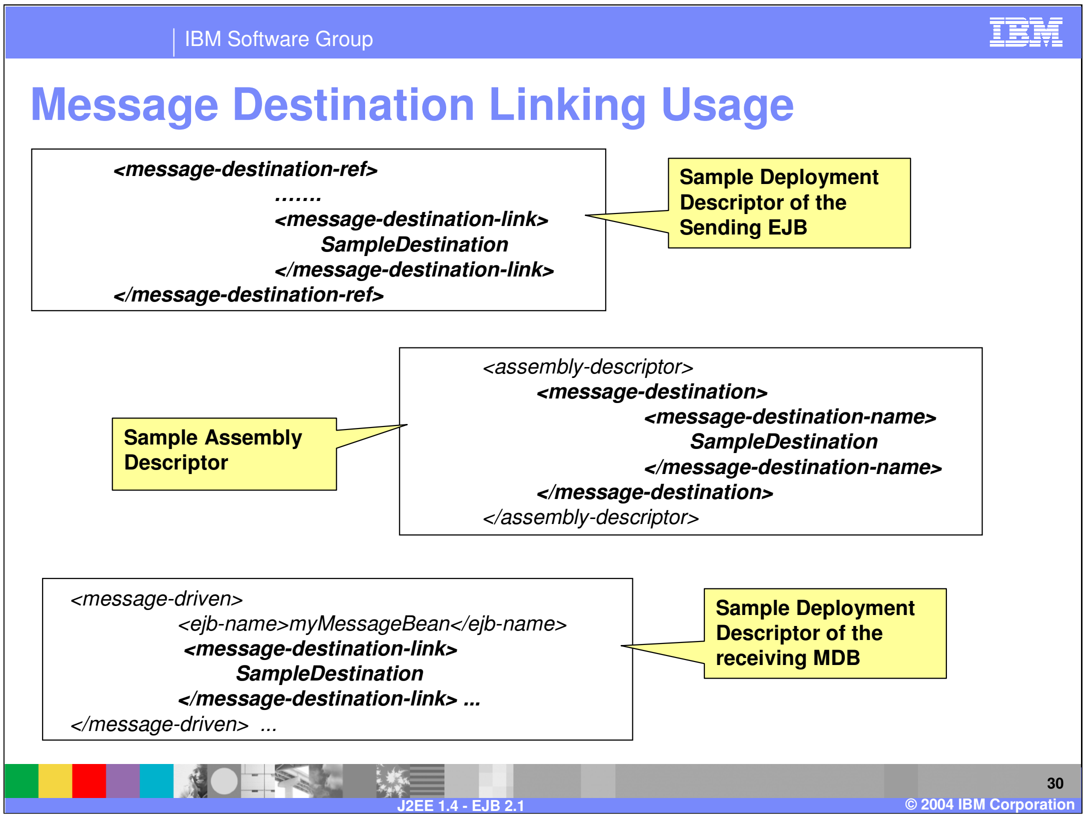


Message Destination Linking

- It allows any enterprise bean (session, entity or message-driven) to send messages to a specified message-driven bean running in the same EJB container
- In the EJB deployment descriptor for the sending EJB, a new element, <message-destination-link> is used to specify a destination
- This destination corresponds to a logical destination defined in the <assembly-descriptor> for the beans
- The receiving MDB has a corresponding element <message-destination-link> that points to the same logical destination

At deployment time, the Logical Message Destination declared in the assembly descriptor is mapped to a real Message Destination

EJB Container



Message Destination Linking Usage

```
        <message-destination-ref>
                        …….
                        <message-destination-link>
                            SampleDestination
                        </message-destination-link>
        </message-destination-ref>
```

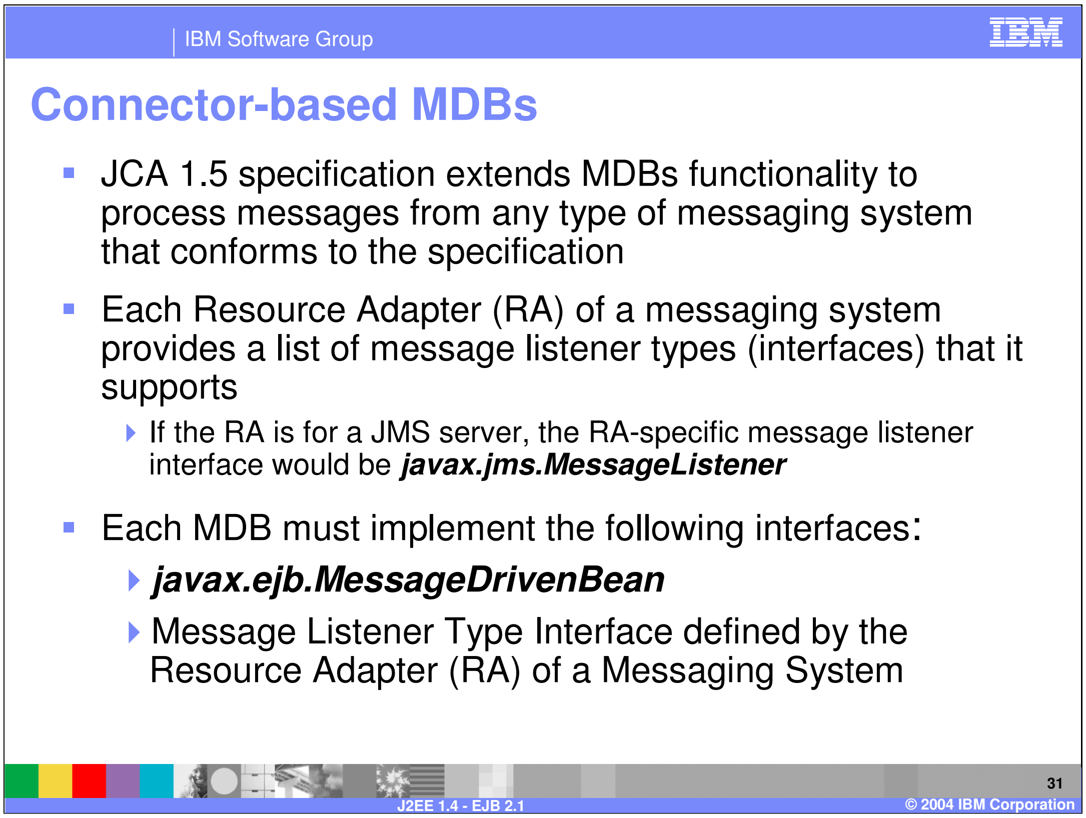
Sample Deployment Descriptor of the Sending EJB

Sample Assembly Descriptor

```
        <assembly-descriptor>
            <message-destination>
                    <message-destination-name>
                        SampleDestination
                    </message-destination-name>
            </message-destination>
        </assembly-descriptor>
```

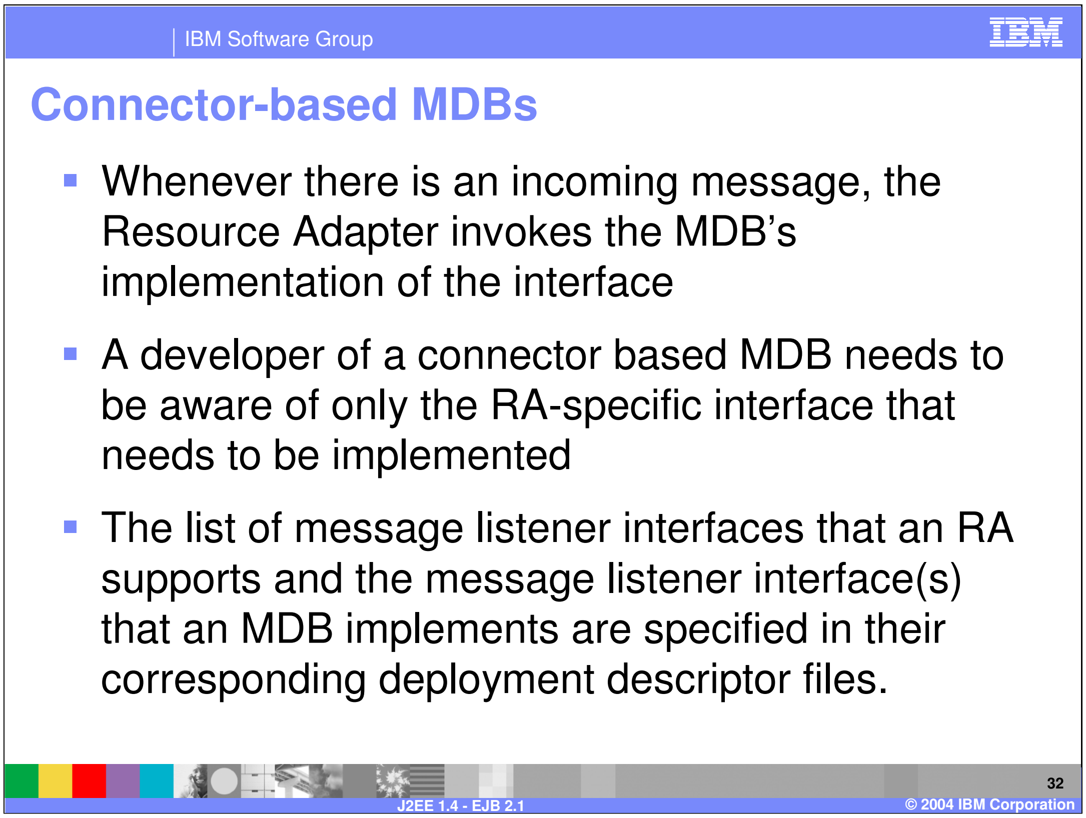
```
    <message-driven>
                <ejb-name>myMessageBean</ejb-name>
                 <message-destination-link>
                     SampleDestination
                </message-destination-link> ...
    </message-driven>  ...
```

Sample Deployment Descriptor of the receiving MDB



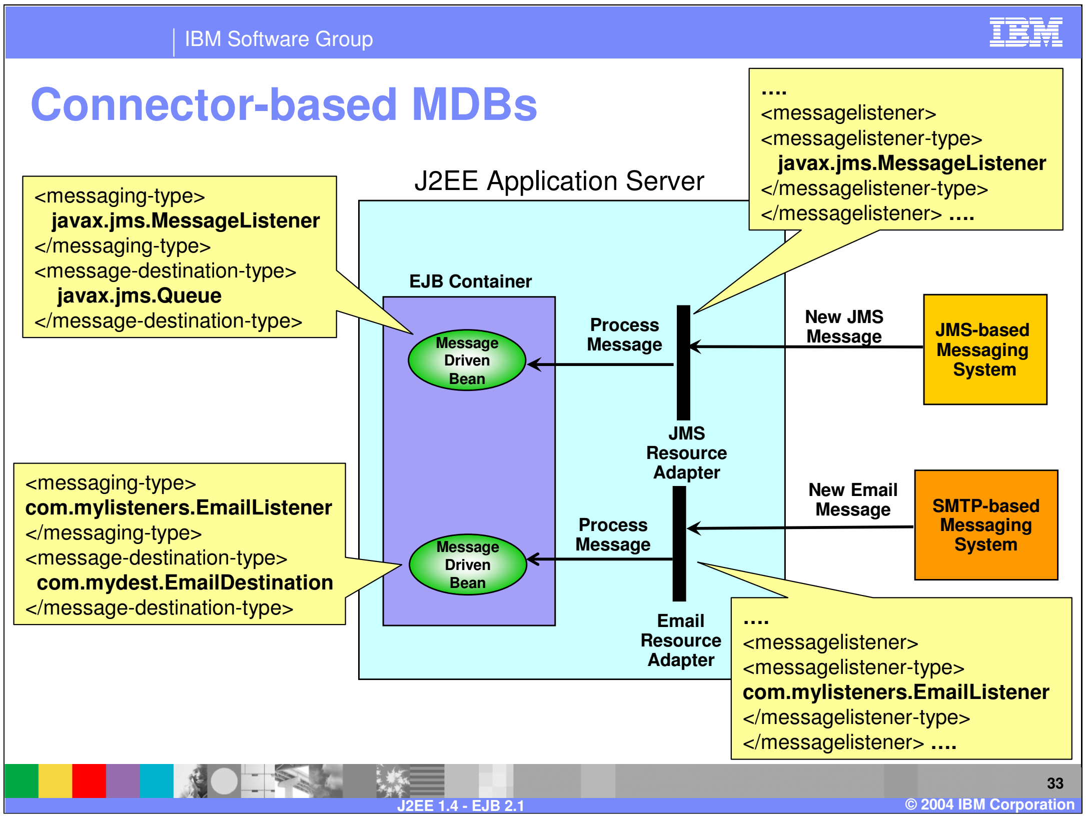
Connector-based MDBs

- JCA 1.5 specification extends MDBs functionality to process messages from any type of messaging system that conforms to the specification
- Each Resource Adapter (RA) of a messaging system provides a list of message listener types (interfaces) that it supports
  - If the RA is for a JMS server, the RA-specific message listener interface would be ***javax.jms.MessageListener***
- Each MDB must implement the following interfaces:
  - ***javax.ejb.MessageDrivenBean***
  - Message Listener Type Interface defined by the Resource Adapter (RA) of a Messaging System



Connector-based MDBs

- Whenever there is an incoming message, the Resource Adapter invokes the MDB’s implementation of the interface
- A developer of a connector based MDB needs to be aware of only the RA-specific interface that needs to be implemented
- The list of message listener interfaces that an RA supports and the message listener interface(s) that an MDB implements are specified in their corresponding deployment descriptor files.



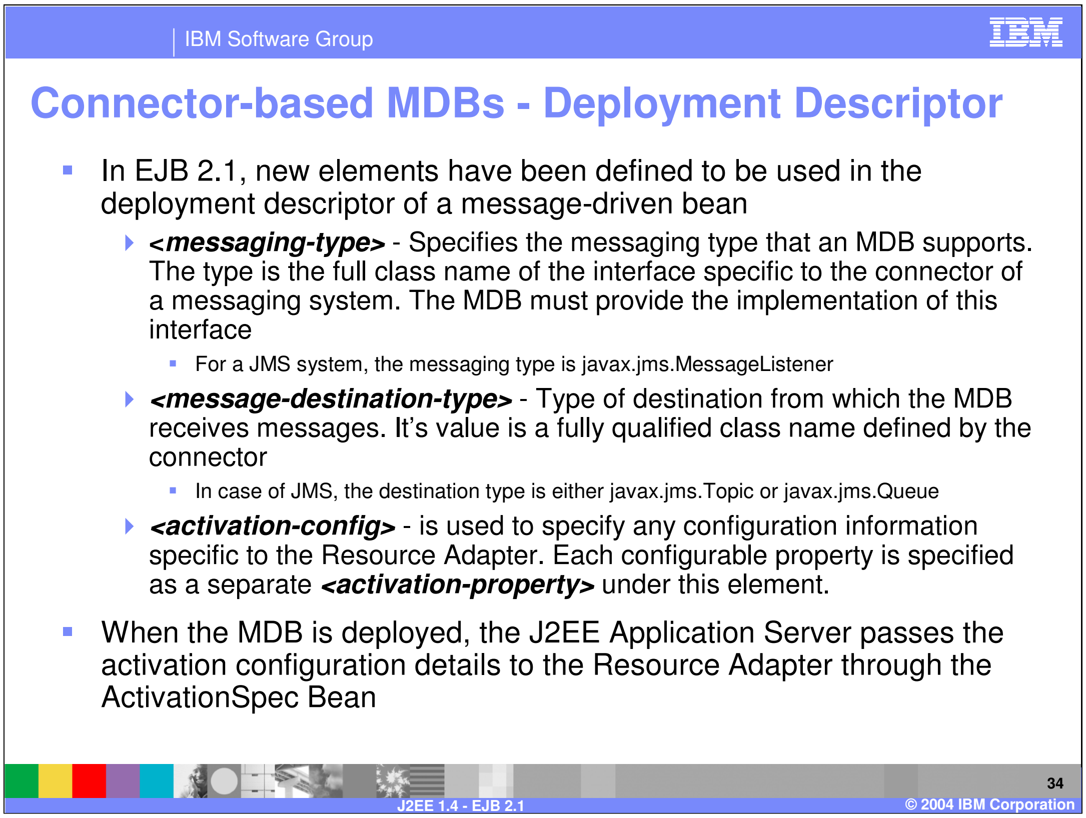
Connector-based MDBs

<messaging-type>
  javax.jms.MessageListener
</messaging-type>
<message-destination-type>
  javax.jms.Queue
</message-destination-type>

<messaging-type>
com.mylisteners.EmailListener
</messaging-type>
<message-destination-type>
  com.mydest.EmailDestination
</message-destination-type>

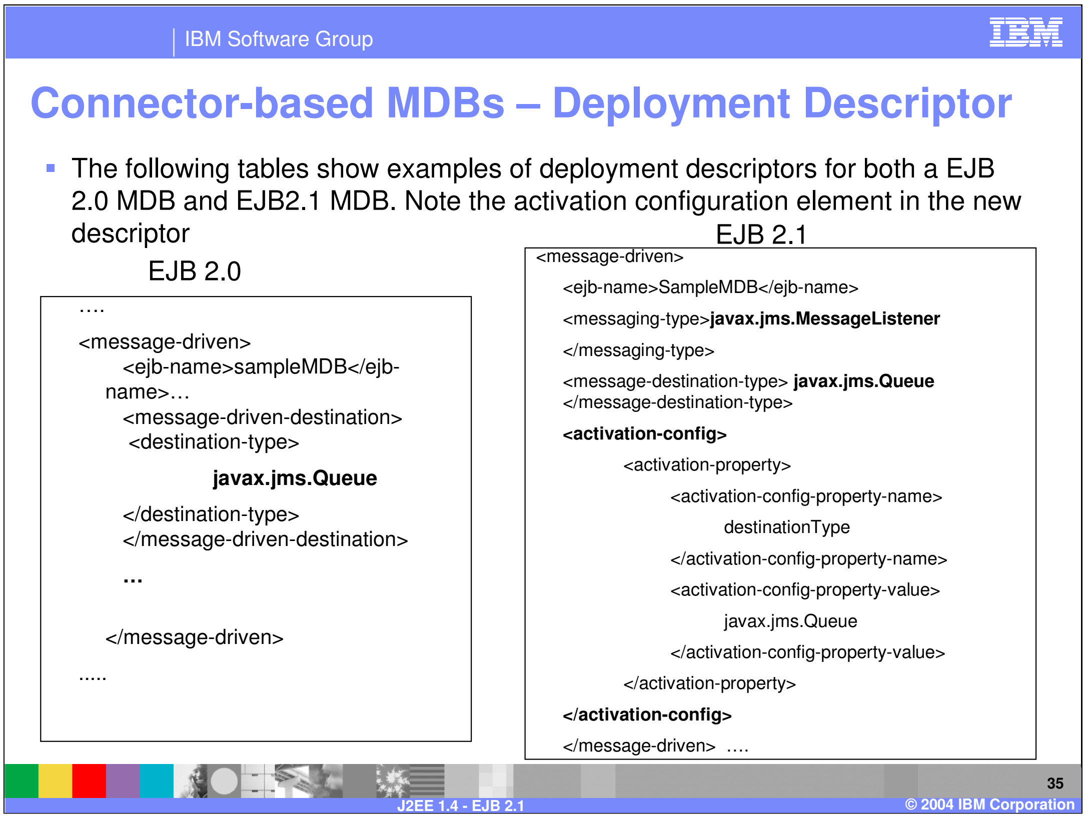
….
<messagelistener>
<messagelistener-type>
  javax.jms.MessageListener
</messagelistener-type>
</messagelistener> **….**

….
<messagelistener>
<messagelistener-type>
com.mylisteners.EmailListener
</messagelistener-type>
</messagelistener> **….**



Connector-based MDBs - Deployment Descriptor

- In EJB 2.1, new elements have been defined to be used in the deployment descriptor of a message-driven bean
  - ***<messaging-type>*** - Specifies the messaging type that an MDB supports. The type is the full class name of the interface specific to the connector of a messaging system. The MDB must provide the implementation of this interface
    - For a JMS system, the messaging type is javax.jms.MessageListener
  - ***<message-destination-type>*** - Type of destination from which the MDB receives messages. It's value is a fully qualified class name defined by the connector
    - In case of JMS, the destination type is either javax.jms.Topic or javax.jms.Queue
  - ***<activation-config>*** - is used to specify any configuration information specific to the Resource Adapter. Each configurable property is specified as a separate ***<activation-property>*** under this element.
- When the MDB is deployed, the J2EE Application Server passes the activation configuration details to the Resource Adapter through the ActivationSpec Bean



Connector-based MDBs – Deployment Descriptor

- The following tables show examples of deployment descriptors for both a EJB 2.0 MDB and EJB2.1 MDB. Note the activation configuration element in the new descriptor

EJB 2.0

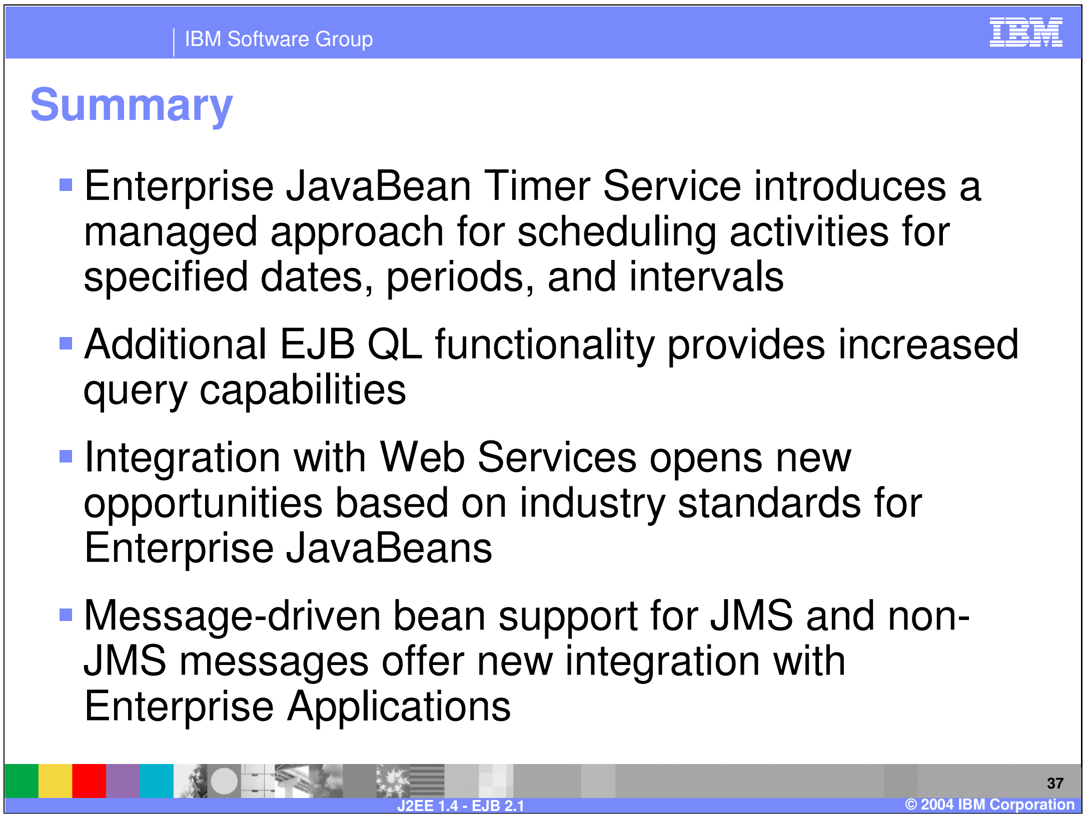
```
 ….
 <message-driven>
       <ejb-name>sampleMDB</ejb-
    name>…
       <message-driven-destination>
        <destination-type>
                    javax.jms.Queue
       </destination-type>
       </message-driven-destination>
       …

    </message-driven>
 …..
```

EJB 2.1

```
<message-driven>
    <ejb-name>SampleMDB</ejb-name>
    <messaging-type>javax.jms.MessageListener
    </messaging-type>
    <message-destination-type> javax.jms.Queue
    </message-destination-type>
    <activation-config>
            <activation-property>
                  <activation-config-property-name>
                         destinationType
                  </activation-config-property-name>
                  <activation-config-property-value>
                         javax.jms.Queue
                  </activation-config-property-value>
            </activation-property>
    </activation-config>
    </message-driven>  ….
```

Section

Summary

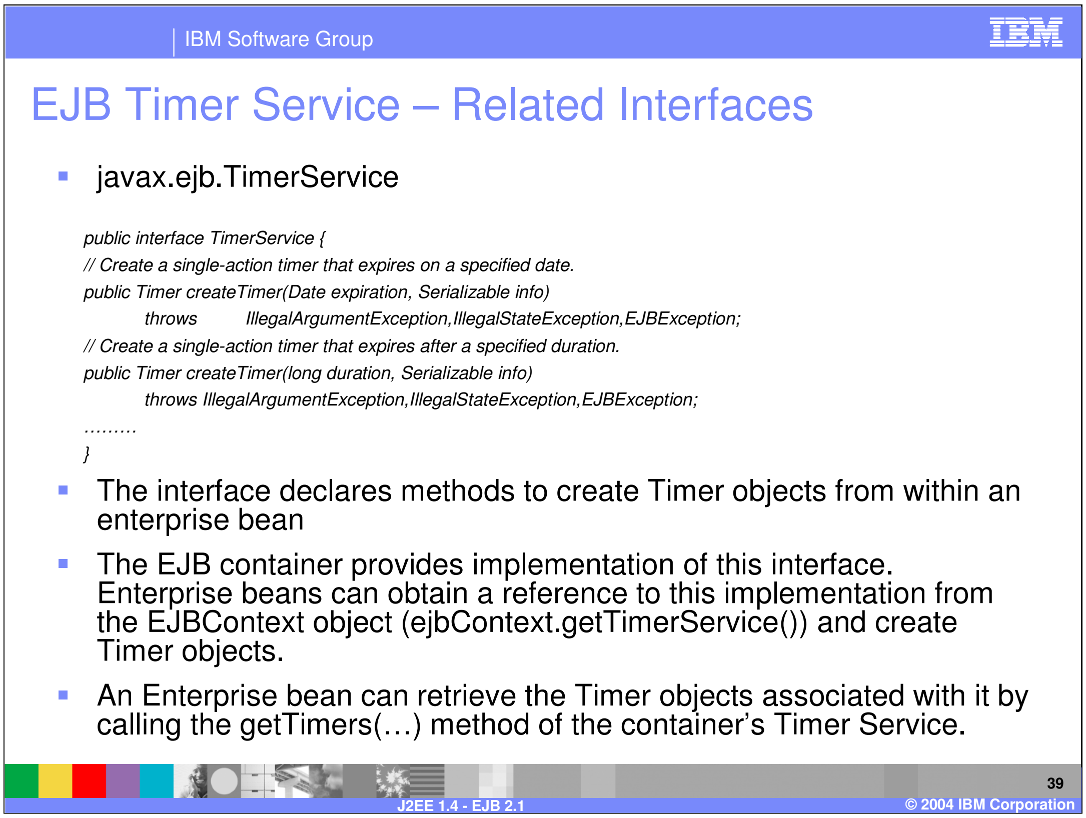


Summary

- Enterprise JavaBean Timer Service introduces a managed approach for scheduling activities for specified dates, periods, and intervals
- Additional EJB QL functionality provides increased query capabilities
- Integration with Web Services opens new opportunities based on industry standards for Enterprise JavaBeans
- Message-driven bean support for JMS and non-JMS messages offer new integration with Enterprise Applications

Section

Appendix

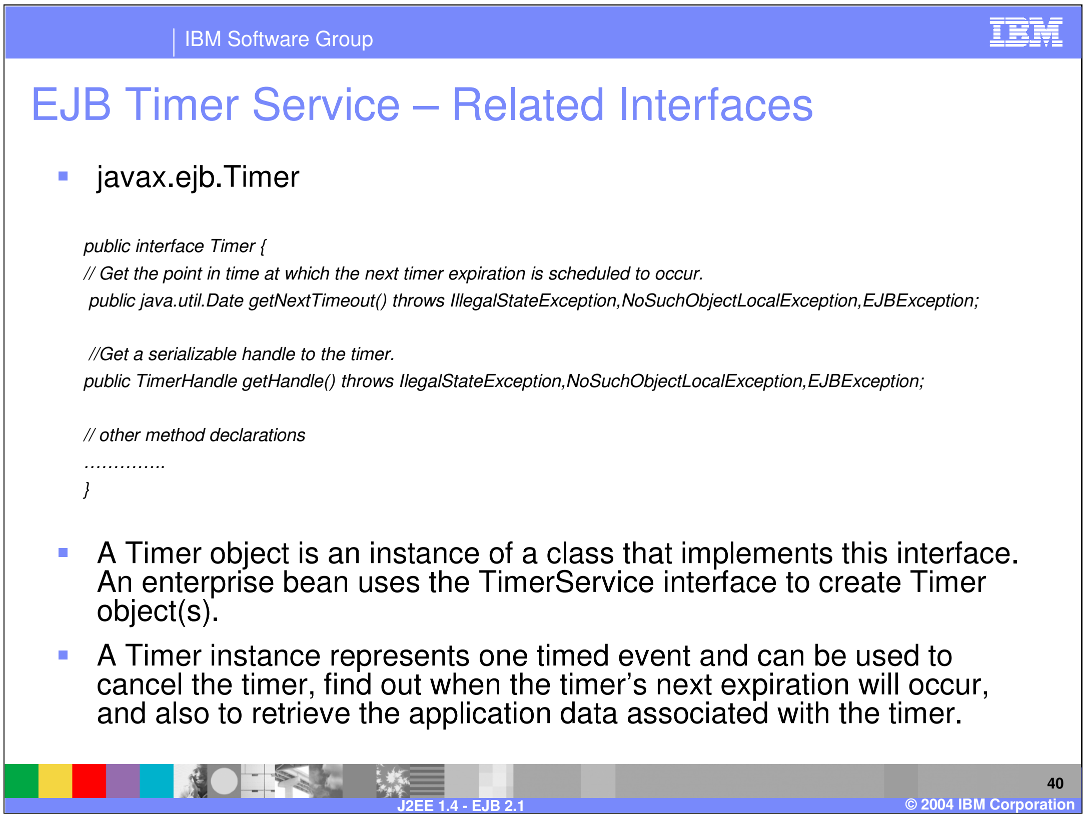


EJB Timer Service – Related Interfaces

- javax.ejb.TimerService

```
public interface TimerService {
// Create a single-action timer that expires on a specified date.
public Timer createTimer(Date expiration, Serializable info)
        throws       IllegalArgumentException,IllegalStateException,EJBException;
// Create a single-action timer that expires after a specified duration.
public Timer createTimer(long duration, Serializable info)
        throws IllegalArgumentException,IllegalStateException,EJBException;
………
}
```

- The interface declares methods to create Timer objects from within an enterprise bean
- The EJB container provides implementation of this interface. Enterprise beans can obtain a reference to this implementation from the EJBContext object (ejbContext.getTimerService()) and create Timer objects.
- An Enterprise bean can retrieve the Timer objects associated with it by calling the getTimers(…) method of the container's Timer Service.

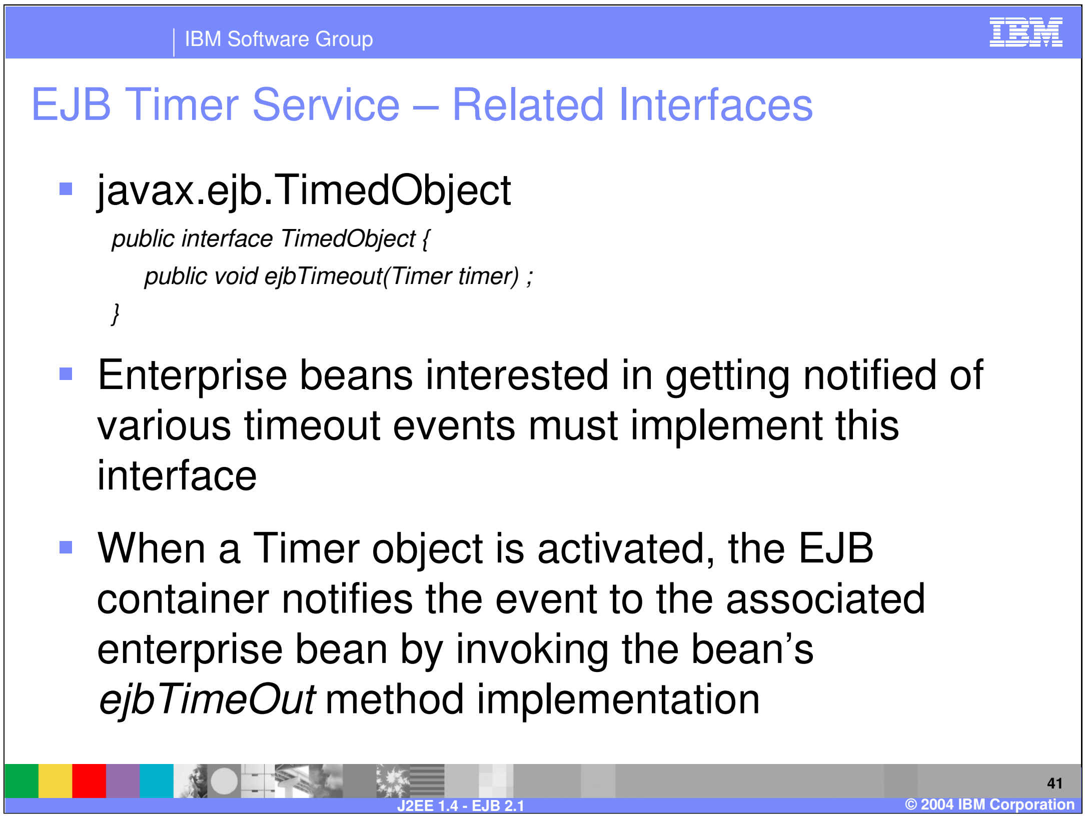


EJB Timer Service – Related Interfaces

- javax.ejb.Timer

```
public interface Timer {
// Get the point in time at which the next timer expiration is scheduled to occur.
 public java.util.Date getNextTimeout() throws IllegalStateException,NoSuchObjectLocalException,EJBException;

 //Get a serializable handle to the timer.
public TimerHandle getHandle() throws IlegalStateException,NoSuchObjectLocalException,EJBException;

// other method declarations
…………..
}
```

- A Timer object is an instance of a class that implements this interface. An enterprise bean uses the TimerService interface to create Timer object(s).
- A Timer instance represents one timed event and can be used to cancel the timer, find out when the timer's next expiration will occur, and also to retrieve the application data associated with the timer.

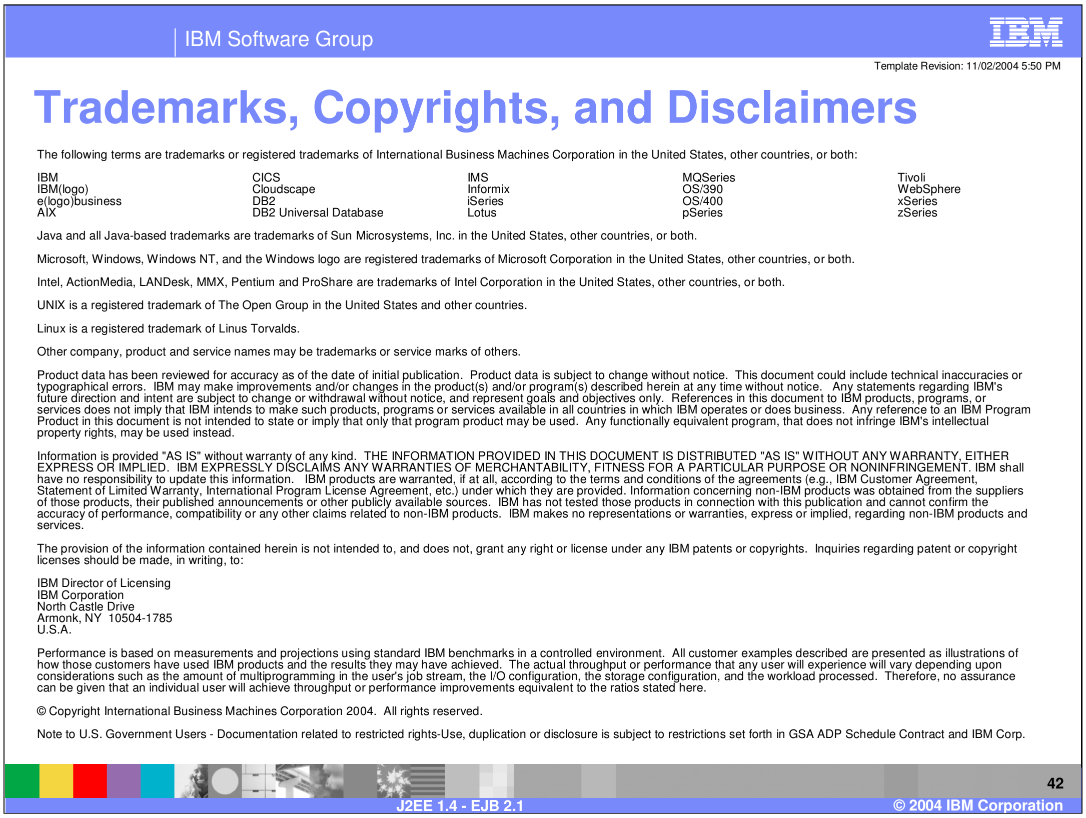


EJB Timer Service – Related Interfaces

- javax.ejb.TimedObject

```
public interface TimedObject {
    public void ejbTimeout(Timer timer) ;
}
```

- Enterprise beans interested in getting notified of various timeout events must implement this interface
- When a Timer object is activated, the EJB container notifies the event to the associated enterprise bean by invoking the bean's *ejbTimeOut* method implementation



Template Revision: 11/02/2004 5:50 PM

Trademarks, Copyrights, and Disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM
IBM(logo)
e(logo)business
AIX

CICS
Cloudscape
DB2
DB2 Universal Database

IMS
Informix
iSeries
Lotus

MQSeries
OS/390
OS/400
pSeries

Tivoli
WebSphere
xSeries
zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.