Routing mechanisms in WebSphere Message Broker V7

Ribu Rajan
Ritesh Srivastava

July 06, 2011

One of the core capabilities of WebSphere Message Broker is message routing, which enables messages to be routed from sender to receiver based on message content. This article describes the content-based routing mechanisms available in WebSphere Message Broker, including the Route node, RoutetoLabel node, Label node, Filter node, Compute node, and JavaCompute node.

Introduction

This article describes the various routing mechanisms available in IBM® WebSphere® Message Broker (hereafter called Message Broker). Messages can be routed through a message flow or broker by using either the built-in routing nodes, or a publish/subscribe application. This article shows you how to use the routing nodes. To benefit from this article you should have basic knowledge of:

- WebSphere Message Broker
- WebSphere Message Broker Toolkit
- Message flow development
- Routing

Routing using nodes

Message Broker provides several built-in nodes that you can use in different ways to control the path that a message takes through a message flow. You can route messages dynamically using the structure or content of a message. This article shows you how to configure and use the following Message Broker nodes:

- Route node
- RoutetoLabel and Label node
- Filter node
- Compute node

© Copyright IBM Corporation 2011

- JavaCompute node

The Route, RoutetoLabel, and Label nodes are non-programming nodes, and you can use them in a message flow without writing any processing code, such as ESQL or Java. You can use XPath expressions to quickly manipulate messages using these nodes.

The Route, RoutetoLabel, Label, and Filter nodes are specifically used for routing purposes. The Compute and JavaCompute nodes are generic programming nodes that you can extend in order to perform sophisticated message routing. This article shows you how to perform content-based routing can be achieved using all five of these nodes.

Route node

This section describes the Route node and its capabilities and usage patterns. The Route node is contained in the Routing drawer of the Message Flow Node palette and is represented in the WebSphere Message Broker Toolkit by the following icon:

Figure 1. Route node icon

Node capabilities

The Route node can be used in message flows to route messages that meet certain criteria down different paths of a message flow. As mentioned earlier, it is a non-programming node, and the criteria for routing messages is set using XPath filter expressions.

Terminals

The Route node has one input terminal and at least three output terminals: Match, Default, and Failure. The Default and Failure output terminals are static, so they are always present on the node. The Match terminal is dynamic and is created automatically each time a new Route node is selected and used in the Message Flow Editor:

Terminals of Route node

Terminal	Description
In	Accepts a message to process by the node
Match	A dynamic output terminal to which the original message can be routed when processing completes successfully
Default	The static output terminal to which the message is routed if no filter expression resolves to true
Failure	The static output terminal to which the message is routed if no filter expression resolves to true

The Route node can have further dynamic output terminals. Not all dynamic output terminals that are created on a Route node need to be mapped to an expression in the filter table. For unmapped dynamic output terminals, messages are never propagated to them. Several expressions can map to the same single dynamic output terminal. No static output terminal exists to which the message is passed straight through. The Route node determines the order in which the terminals are driven. The node always propagates messages to the terminals in the order in which they appear in the filter table.

Properties

Filter Table -- A table containing all the criteria for processing messages by this node. Configure this table with the XPath expressions. Its a mandatory field and must contain at least one row representing a routing decision. Each row specifies a user-defined routing expression and a dynamic terminal for the node to propagate the incoming message based on the result of filter expression. The filter expression result is cast as Boolean value.

All XPath expressions must start with $Root, $Body, $Properties, $LocalEnvironment, $DestinationList, $ExceptionList, or $Environment. Expressions are evaluated in the order in which they appear in the table. To improve performance, specify the expressions that are satisfied most frequently at the top of the filter table. Typically, you specify a unique terminal name for each XPath expression.

Distribution Mode -- By default the value of this field is ALL. When this property is set to ALL, every row in the table is guaranteed to be processed synchronously in the order given. A copy of the node's input message assembly is propagated down each row's user-defined dynamic output terminal, where its routing expressions resolve to true. If the routing expression resolve to false, then the node's input message assembly is propagated down the static output terminal named Default.

When the property is set to FIRST, each row in the table is processed synchronously in the order given until a routing decision is met. For routing expressions resolving to true, a copy of the node's input message assembly is propagated down the row's user-defined dynamic output terminal. Otherwise, the node's input message assembly is propagated down the static output terminal named Default.

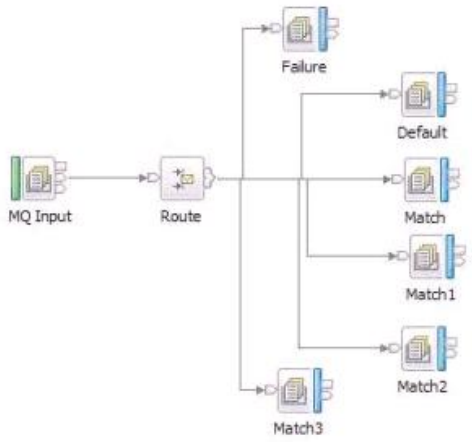
The monitoring properties of the Route node are described below:

Events -- Events that you have defined for the node are displayed on this tab. By default, no monitoring events are defined on any node in a message flow. Use Add, Edit, and Delete to create, change, or delete monitoring events for the node.

Usage scenario

The Route node directs the messages depending on the filter expressions resolving to True. The filter expression always evaluates to produce a True or False condition. Based on the final result, the message is be propagated to the matching output terminal. Figure 2 shows the Route node example using different XPath expressions to route the message to the appropriate queue:

Figure 2. Route node flow



The Output nodes are named according to the terminals connected to it from the Route node. By default, the Route node has one input terminal and three output terminals (Failure, Default, and Match). Since this node can have dynamic terminals, Figure 2 shows additional terminals being added to the Route node, with six nodes wired with Route node to provide six branches of execution within the flow. The routing of the messages depends on the different XPath expressions mentioned in the Filter Table, as shown in Figure 3 below.

In this case, based on the XPath Expression defined under Filter table, consider two different messages:

Message1

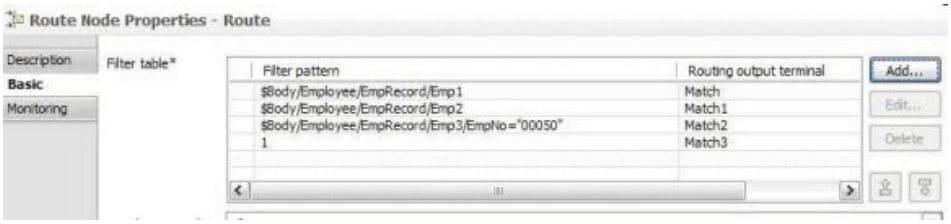
```
<Employee>
   <EmpRecord>
      <Emp1>
         <EmpNo>00051</EmpNo>
         <FirstName>DAVID</FirstName>
         <LastName>BROWN</LastName>
      </Emp1>
      <Emp2>
         <EmpNo>00052</EmpNo>
         <FirstName>MARIE</FirstName>
         <LastName>PHIL</LastName>
      </Emp2>
   </EmpRecord>
</Employee>
```

Message2

```
<Employee>
   <EmpRecord>
      <Emp3>
         <EmpNo>00053</EmpNo>
         <FirstName>DAVID</FirstName>
         <LastName>BROWN</LastName>
      </Emp3>
  </EmpRecord>
</Employee>
```

The message is routed based on the filter table below:

Figure 3. XPath expression for the Route node



If none of the XPath expressions match the message content, then by default, the message is routed to the Default terminal of the Route node. If an exception is detected during processing of the message, then the message will be routed to the Failure terminal. If the Failure Terminal is not wired to a node, the message is routed to the Dead Letter Queue.

For the message to be routed to the Match, Match1, Match2, or Match3 terminals, there should be at least one XPath Expression in the filter table resolving to True.

While processing the input message, Message Broker checks for the result of each expression in the filter table and routes the message to the corresponding terminal.

With Message1 as input, filter expression $Body/Employee/EmpRecord/Emp1 evaluates to True and the message is routed to the Match terminal. Similarly, $Body/Employee/EmpRecord/Emp2 evaluates to True and is be routed to the Match1 Terminal.

With Message 2 as input, filter expression $Body/Employee/EmpRecord/Emp3/EmpNo="00050" never evaluates to True and is be routed to the Default terminal.

Regardless of the input messages, the filter expression 1 is always cast as Boolean 1 (True), and messages are always routed to its corresponding terminal.

Filter node

This section describes the Filter node and its capabilities and usage patterns. The Filter node is contained in the Routing drawer of the Message Flow Node Palette and is represented in the Message Broker Toolkit by the following icon:

Figure 4. Filter node icon

Node Capabilities

The Filter node is used for content-based routing. Unlike the Route node, the Filter expression can be defined using ESQL. Although you use ESQL statements for the Filter node, the usage is restricted until the point of routing, and therefore the processed message remains unchanged at the output terminal. The Routing depends only on the expression evaluating to True, False, or Unknown:

Terminals of Filter node

Terminal	Description
In	Accepts a message for further processing.
Unknown	If the expression evaluates to Unknown or a null value, the message is routed to Unknown terminal.
True	The output terminal to which the message is routed if the filter expression evaluates to True.
False	The output terminal to which the message is routed if the filter expression evaluates to False.
Failure	The output terminal to which the message is routed in case of any failure during the computation.

Properties of Filter node

Property	Description
Data Source	This property refers to the ODBC datasource name that you refer to in the ESQL associated with the node.
Transaction	There are two transaction modes: With Automatic (the default), the commit depends on the success or failure of the entire message flow. If there is a failure, the entire process including the database operations are rolled back. With Commit, the database changes are committed regardless of success or failure.
Filter Expression	Refers to the name of the ESQL module to which the filter node is associated. If there are multiple modules, use the Browse button to select the appropriate modules.
Treat warnings as error	By default, the checkbox is unchecked. In this case, regardless of warnings from the database, the message is be propagated normally. However, if it is checked, the output message is propagated from the node to the Failure terminal.
Throw exception on database error	By default, the checkbox is checked. In this case, if there are database errors, then the node throws an exception. If the check box is unchecked, you must include ESQL to handle any database error.

Monitoring Events can also be configured for this node using the Events property.

Usage scenario

Consider a scenario where an organization is trying to filter skill sets based on their database record. The records are in XML and serve as input to the message flow. Consider the messages below:

Message1

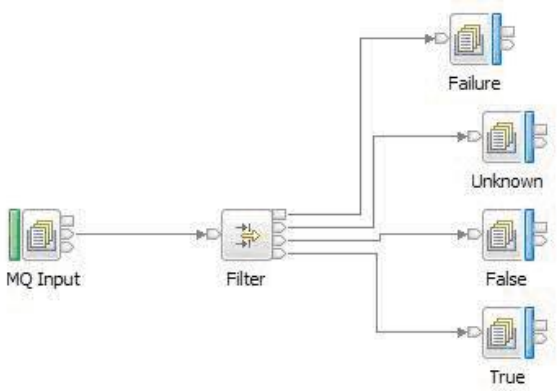
```
<Organisation>
   <OrgDetails>
      <OrgName>IBM</OrgName>
   </OrgDetails>
   <ListOfEmployees>
      <EmployeeDetails><FirstName>Mary</FirstName><LastName>Smith</LastName>
         </EmployeeDetails>
      <EmployeeDetails><FirstName>Diane</FirstName><LastName>Rose</LastName>
         </EmployeeDetails>
   </ListOfEmployees>
   <Skill>WMB</Skill>
</Organisation>
```

Message2

```
<Organisation>
   <OrgDetails>
      <OrgName>IBM</OrgName>
   </OrgDetails>
   <ListOfEmployees>
      <EmployeeDetails><FirstName>Debra</FirstName><LastName>Wiess</LastName>
         </EmployeeDetails>
      <EmployeeDetails><FirstName>Atila</FirstName><LastName>Wiess</LastName>
         </EmployeeDetails>
   </ListOfEmployees>
   <Skill>WMQ</Skill>
</Organisation>
```

These messages are used as input tor the message flow below:

Figure 5. Filter node flow



Output nodes are named according to the terminals connected to them.

ESQL for Filter node in Figure 5

```
CREATE FILTER MODULE Filter_Samp_Filter
CREATE FUNCTION Main() RETURNS BOOLEAN
BEGIN
   IF Root.XMLNSC.Organisation.Skill = 'WMB' THEN
      RETURN TRUE;
   END IF;
   IF Root.XMLNSC.Organisation.Skill ='WMQ' THEN
      RETURN FALSE;
   END IF;

   RETURN UNKNOWN;
END;
END MODULE;
```

The above ESQL code filters the Employee details based on the skill set and routes the message to the corresponding terminals.

For Message1, Root.XMLNSC.Organisation.Skill = WMB evaluates to True, the corresponding Return statement is executed, and the message is routed to the True terminal.

For Message 2, Root.XMLNSC.Organisation.Skill = WMQ evaluates to True, the corresponding Return statement is executed, and the message is routed to the False terminal.

The RETURN UNKNOWN statement in the above ESQL is not mandatory and by default. is executed for a message that doesn't satisfy either of the ESQL conditions, with the message routed to the Unknown terminal.

In the case of an ESQL computation exception, the message is routed to the Failure node. If the Failure node is not connected, the message will is routed to the Dead Letter Queue.

RoutetoLabel and Label nodes

This section describes RoutetoLabel and Label nodes, their capabilities, and usage patterns. The RoutetoLabel and Label nodes are contained in the Routing drawer of the Message Flow Node Palette and represented in the WebSphere Message Broker Toolkit by the following icon :

Figure 6. RoutetoLabel node icon

Node capabilities

The RoutetoLabel node provides a dynamic routing facility based on the contents of the LocalEnvironment associated with the message. LocalEnvironment contains the identity of one or more target Label nodes, identified by their Label Name property (not the node name).

The Label node is a named destination for a message processed by a RoutetoLabel node. The Label node is identified by an entry in the LocalEnvironment of the message when it is processed by a RoutetoLabel node.

Using the RoutetoLabel and Label nodes in combination provides dynamic routing within a message flow, because you can determine the destination from the contents of the LocalEnvironment within the message itself -- the RoutetoLabel node checks the local environment of the message to determine the identifier of the Label node to which to route the message. By itself, the RoutetoLabel node cannot generate the LocalEnvironment for a message, and therefore the RoutetoLabel node must always be preceded by a Compute node, JavaCompute node, or HTTPInput node (provided that the Set destination list property is checked) to populate the local environment of the message with the identifiers of one or more Label nodes. The Label names can be any string value, and can be specified explicitly in the Compute node, taken or cast from any field in the message, or retrieved from a database. A label name in the local environment must match the Label Name property of a corresponding Label node.

Design your message flow so that one or more Label nodes logically follows a RoutetoLabel node in a message flow. The RoutetoLabel node cannot be physically wired to a Label node. The connection is made by the Broker, when required, according to the contents of LocalEnvironment. For the LocalEnvironment variables to be available in the Compute node, change the Compute Mode property from Message to LocalEnvironment.

Terminals

The RoutetoLabel node has an In terminal to accept a message for processing and a Failure node to which the message is routed if a failure is detected during processing.

Label node has only an Out terminal to which the incoming messages are routed.

Properties

RoutetoLabel node

Property	Description
Mode	Mandatory field that controls how the RoutetoLabel node processes the items in the local environment associated with the current message. It has two options: 1. Route To First -- Removes the first item from the local environment. The current message is routed to the Label node identified by LabelName in that list item. 2. Route To Last (default) -- Removes the last item from the local environment. The current message is routed to the Label node identified by LabelName in that list item.
Events	Monitoring properties of the node are mentioned here. Events that you have defined for the node are displayed on this tab. By default, no monitoring events are defined on any node in a message flow. Use Add, Edit, and Delete to create, change, or delete monitoring events for the node.

Label node

Property	Description
Label Name	This required field serves as an identifier for the node. It is used as a target for a message that is routed by a RoutetoLabel node. The Label Name must not be the same as the name of the instance of the node itself, and it must be unique in the message flow in which it appears.
Events	Monitoring properties of the node are mentioned here. Events that you have defined for the node are displayed on this tab. By default, no monitoring events are defined on any node in a message flow. Use Add, Edit, or Delete to create, change, or delete monitoring events for the node

Usage scenario

Considering the same scenario used above with the Filter node. Filter node can be used to separate two queries, but if you have more than two queries, you need different filters. But using RoutetoLabel node you only have to add a new Label identifier and a new condition. To understand the scenario better, consider the messages below:

Message1

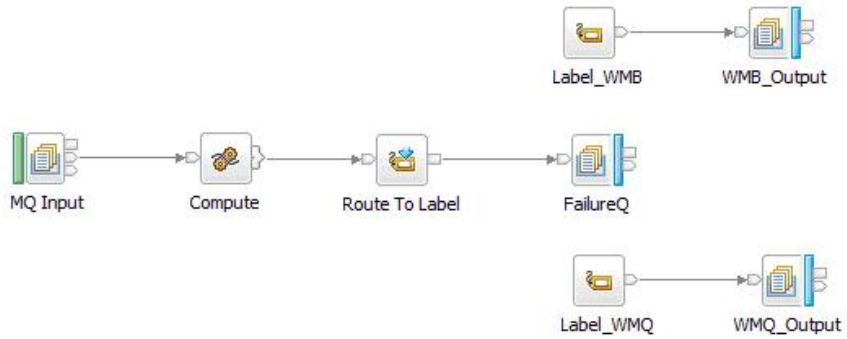
```
<Organisation>
   <OrgDetails>
      <OrgName>IBM</OrgName>
   </OrgDetails>
   <ListOfEmployees>
      <EmployeeDetails><FirstName>Mary</FirstName><LastName>Smith</LastName>
         </EmployeeDetails>
      <EmployeeDetails><FirstName>Diane</FirstName><LastName>Rose</LastName>
         </EmployeeDetails>
   </ListOfEmployees>
   <Skill>WMB</Skill>
</Organisation>
```

Message2

```
<Organisation>
   <OrgDetails>
      <OrgName>IBM</OrgName>
   </OrgDetails>
   <ListOfEmployees>
      <EmployeeDetails><FirstName>Debra</FirstName><LastName>Wiess</LastName>
         </EmployeeDetails>
      <EmployeeDetails><FirstName>Atila</FirstName><LastName>Wiess</LastName>
         </EmployeeDetails>
   </ListOfEmployees>
   <Skill>WMQ</Skill>
</Organisation>
```

These messages are used as input for the message flow below, as shown in Figure 7:

Figure 7. RoutetoLabel and Label node flow



ESQL for the Compute node in Figure 7

```
CREATE COMPUTE MODULE DecideOnQuery
CREATE FUNCTION Main() RETURNS BOOLEAN
BEGIN
   SET OutputRoot = InputRoot;
   IF InputRoot.XMLNSC.Organisation.Skill ='WMB' THEN
      Set OutputLocalEnvironment.Destination.RouterList.DestinationData[1]
         .labelname = 'WMB';
   ELSE
      Set OutputLocalEnvironment.Destination.RouterList.DestinationData[1]
         .labelname = 'WMQ';
   END IF;
RETURN TRUE;
END;
END MODULE;
```

The above ESQL code in the Compute node filters the Employee details based on the skill set mentioned in the input message.

For Message1, InputRoot.XMLNSC.Organisation.Skill = WMB evaluates to True, the corresponding statement is executed, and LocalEnvironment is set to a label with identifier WMB.

For Message 2, InputRoot.XMLNSC.Organisation.Skill = WMQ evaluates to True, the corresponding statement is executed, and LocalEnvironment is set to a label with identifier WMQ.

Then, for each message reaching the RoutetoLabel node, the node checks the label identifier in the LocalEnvironment of the message and routes the message to the corresponding label. If a failure occurs while processing the message, it is routed to the failure terminal of the RoutetoLabel node and hits the FailureQ.

Compute node

You can use the Compute node to construct one or more new messages using ESQL, and perform the following tasks:

1. Build a new message using a set of assignment statements
2. Copy messages between parsers
3. Convert messages from one code set to another
4. Transform messages from one format to another

Terminal

Terminals of Filter node

Terminal	Description
In	The input terminal that accepts a message for processing by the node.
Failure	The output terminal to which the input message is routed if an unhandled exception occurs during the computation.
Out	The output terminal to which the transformed message is routed when processing in the node is completed. The transformed message may also be routed to this terminal by a PROPAGATE statement.
Out1	The first alternative output terminal to which the transformed message might be routed by a PROPAGATE statement.
Out2	The second alternative output terminal to which the transformed message might be routed by a PROPAGATE statement.
Out3	The third alternative output terminal to which the transformed message might be routed by a PROPAGATE statement.
Out4	The fourth alternative output terminal to which the transformed message might be routed by a PROPAGATE statement.

For more information, see Using the JavaCompute node in the Message Broker V7 information center.

Conclusion

This article described the various routing mechanisms in WebSphere Message Broker V7, and usage scenarios for the Route, Route to Label, Label, Filter, Compute, and JavaCompute nodes. These nodes support multiple languages, including XPath, ESQL, and Java, so that you can use the one you prefer for the routing logic.

Related topics

- **WebSphere Message Broker resources**
  - WebSphere Message Broker V7 information centre
    A single Web portal to all WebSphere Message Broker V7 documentation, with conceptual, task, and reference information on installing, configuring, and using your WebSphere Message Broker environment.
  - WebSphere Message Broker Route node
    Node description, purpose, and configuration from the product information centre.
  - WebSphere Message Broker RoutetoLabel node
    Node description, purpose, and configuration from the product information centre.
  - WebSphere Message Broker Filter node
    Node description, purpose, and configuration from the product information centre.
  - WebSphere Message Broker Compute node
    Node description, purpose, and configuration from the product information centre.
  - WebSphere Message Broker developer resources page
    Technical resources to help you use WebSphere Message Broker for connectivity, universal data transformation, and enterprise-level integration of disparate services, applications, and platforms to power your SOA.
  - WebSphere Message Broker product page
    Product descriptions, product news, training information, support information, and more.
  - What's new in WebSphere Message Broker V7
    WebSphere Message Broker V7 provides universal connectivity with its ability to route and transform messages from anywhere to anywhere. Through its simple programming model and a powerful operational management interface, it makes complex application integration solutions much easier to develop, deploy, and maintain. This article describes the major enhancements in V7.
  - Download free trial version of WebSphere Message Broker V7
    WebSphere Message Broker V7 is an ESB built for universal connectivity and transformation in heterogeneous IT environments. It distributes information and data generated by business events in real time to people, applications, and devices throughout your extended enterprise and beyond.
  - WebSphere Message Broker documentation library
    WebSphere Message Broker specifications and manuals.
  - WebSphere Message Broker forum
    Get answers to your technical questions and share your expertise with other Message Broker users.
  - WebSphere Message Broker support page
    A searchable database of support problems and their solutions, plus downloads, fixes, and problem tracking.
- **WebSphere resources**
  - developerWorks WebSphere developer resources
    Technical information and resources for developers who use WebSphere products. developerWorks WebSphere provides product downloads, how-to information, support

resources, and a free technical library of more than 2000 technical articles, tutorials, best practices, IBM Redbooks, and online product manuals.
- developerWorks WebSphere application connectivity developer resources
  How-to articles, downloads, tutorials, education, product info, and other resources to help you build WebSphere application connectivity and business integration solutions.
- developerWorks WebSphere SOA and Web services developer resources
  How-to articles, downloads, tutorials, education, product info, and other resources to help you design and build WebSphere SOA and Web services solutions.
- Most popular WebSphere trial downloads
  No-charge trial downloads for key WebSphere products.
- WebSphere forums
  Product-specific forums where you can get answers to your technical questions and share your expertise with other WebSphere users.
- WebSphere on-demand demos
  Download and watch these self-running demos, and learn how WebSphere products and technologies can help your company respond to the rapidly changing and increasingly complex business environment.
- developerWorks WebSphere weekly newsletter
  The developerWorks newsletter gives you the latest articles and information only on those topics that interest you. In addition to WebSphere, you can select from Java, Linux, Open source, Rational, SOA, Web services, and other topics. Subscribe now and design your custom mailing.
- WebSphere-related books from IBM Press
  Convenient online ordering through Barnes & Noble.
- WebSphere-related events
  Conferences, trade shows, Webcasts, and other events around the world of interest to WebSphere developers.

- **developerWorks resources**
  - Trial downloads for IBM software products
    No-charge trial downloads for selected IBM® DB2®, Lotus®, Rational®, Tivoli®, and WebSphere® products.
  - developerWorks blogs
    Join a conversation with developerWorks users and authors, and IBM editors and developers.
  - developerWorks tech briefings
    Free technical sessions by IBM experts to accelerate your learning curve and help you succeed in your most challenging software projects. Sessions range from one-hour virtual briefings to half-day and full-day live sessions in cities worldwide.
  - developerWorks podcasts
    Listen to interesting and offbeat interviews and discussions with software innovators.
  - developerWorks on Twitter
    Check out recent Twitter messages and URLs.
  - IBM Education Assistant
    A collection of multimedia educational modules that will help you better understand IBM software products and use them more effectively to meet your business requirements.

© Copyright IBM Corporation 2011
(www.ibm.com/legal/copytrade.shtml)
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)