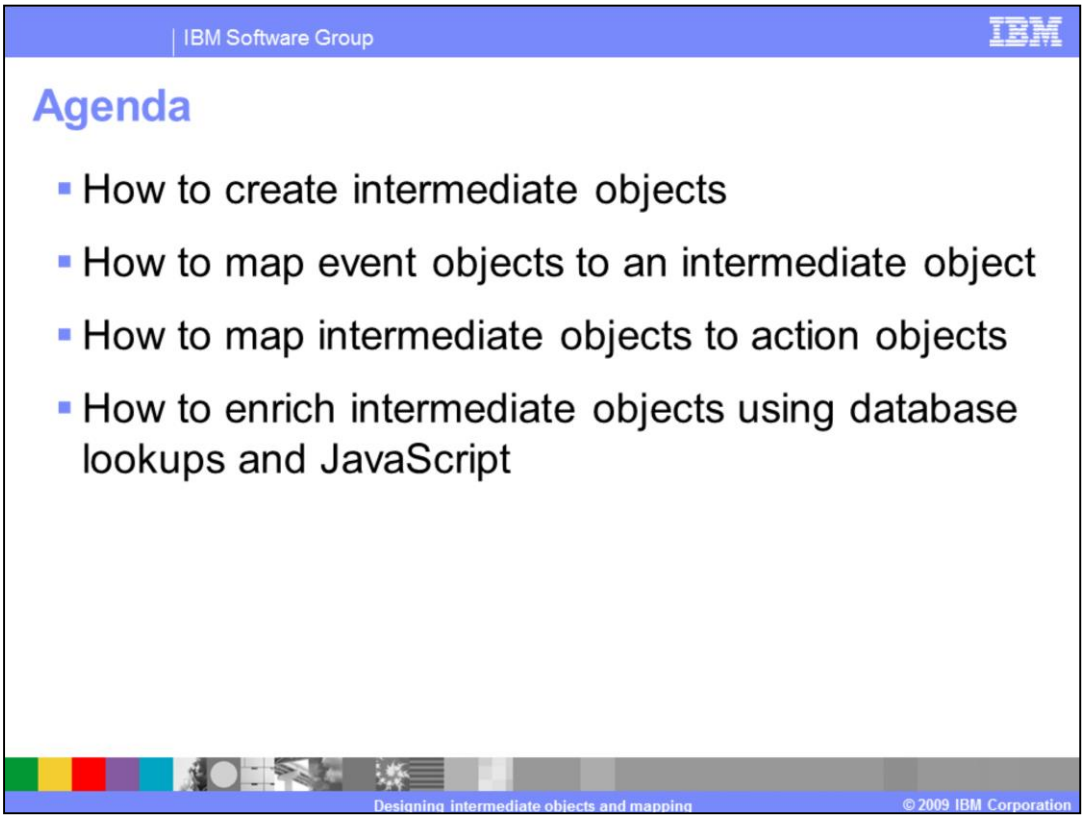
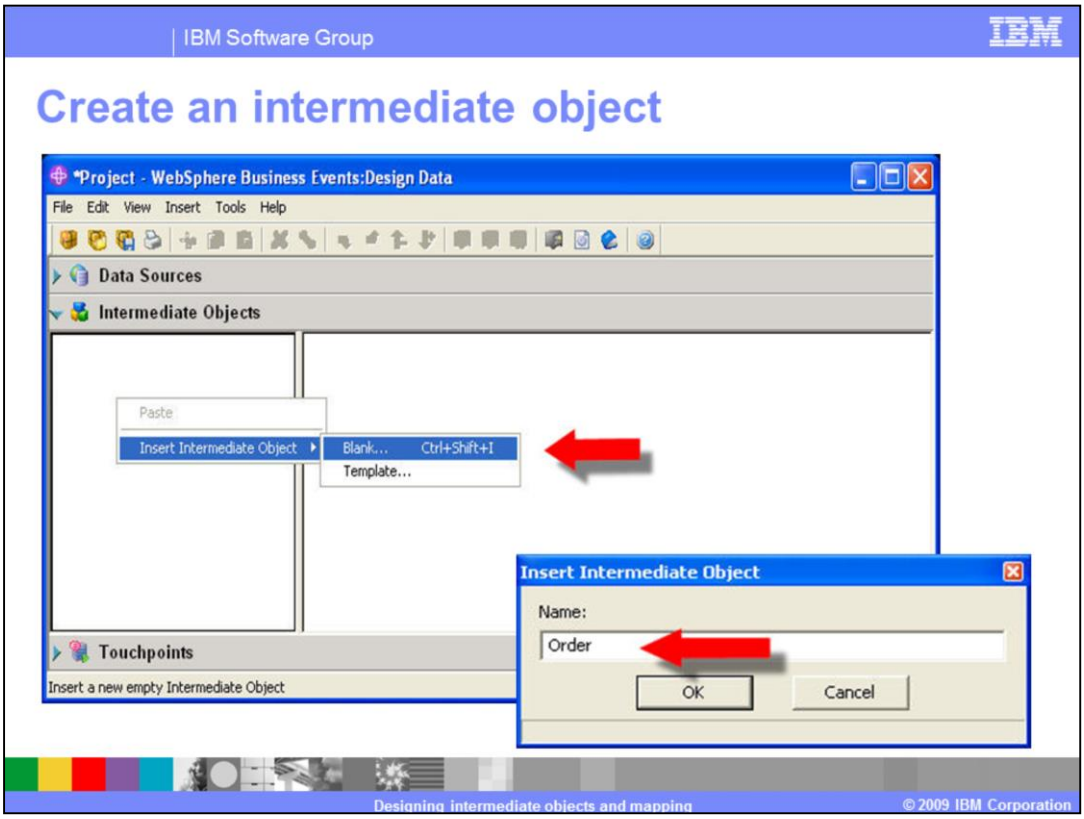
In this presentation you will see how intermediate objects are created and how the data within these objects is mapped to events and actions. Additionally you will see how JavaScript™, constants and database look-ups can be used to enrich WebSphere® Business Events event and action content.



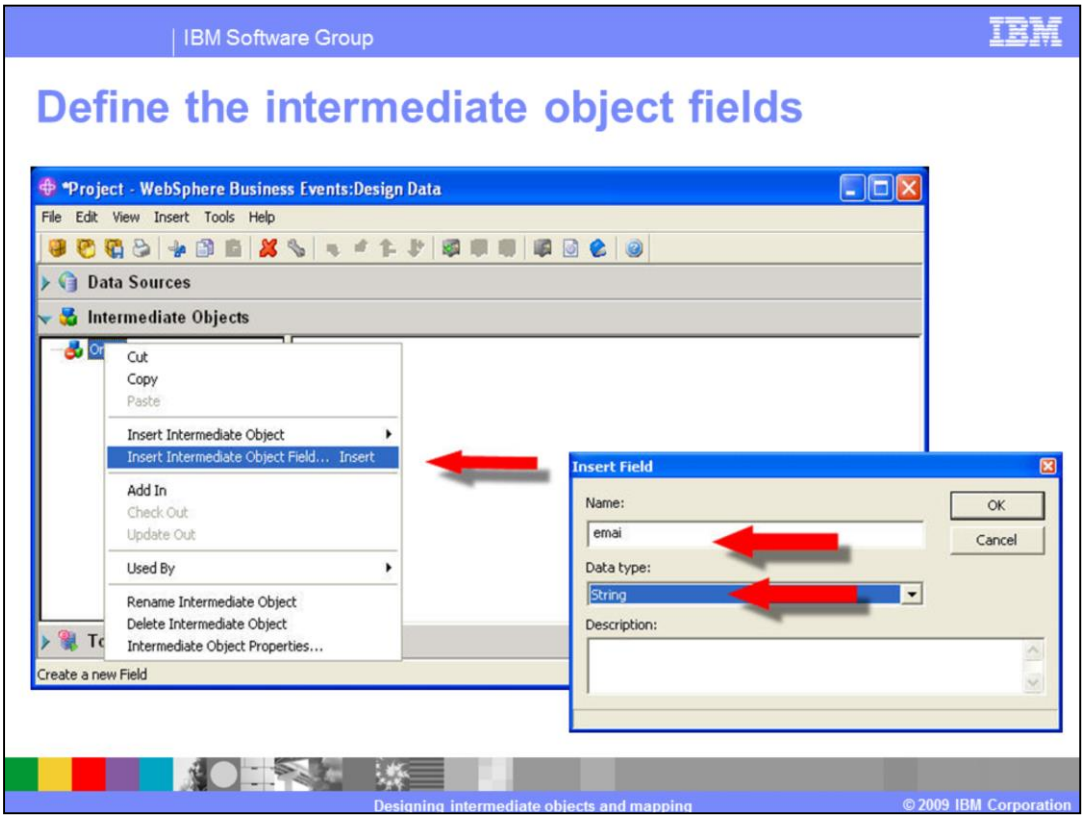
During this presentation you will use the concepts you learned about intermediate objects in the earlier presentations. This presentation will show you how to build an intermediate object using the WebSphere Business Events: Design Data tool. You will learn how to implement the mapping between event objects and intermediate objects; also the mapping between intermediate objects and action objects. Also, you will see how to enrich intermediate objects by using constants, database look-ups and JavaScript. Parts of this presentation assume that you understand concepts introduced in earlier presentations.



In order to create an intermediate object you will use the WebSphere Business Events: Design Data tool.

Having started the Design Data tool and expanded the intermediate objects bar, create an intermediate object by right-clicking in the intermediate objects pane and selecting ‘Insert Intermediate Object’, then ‘Blank’ from the pop-up menu.

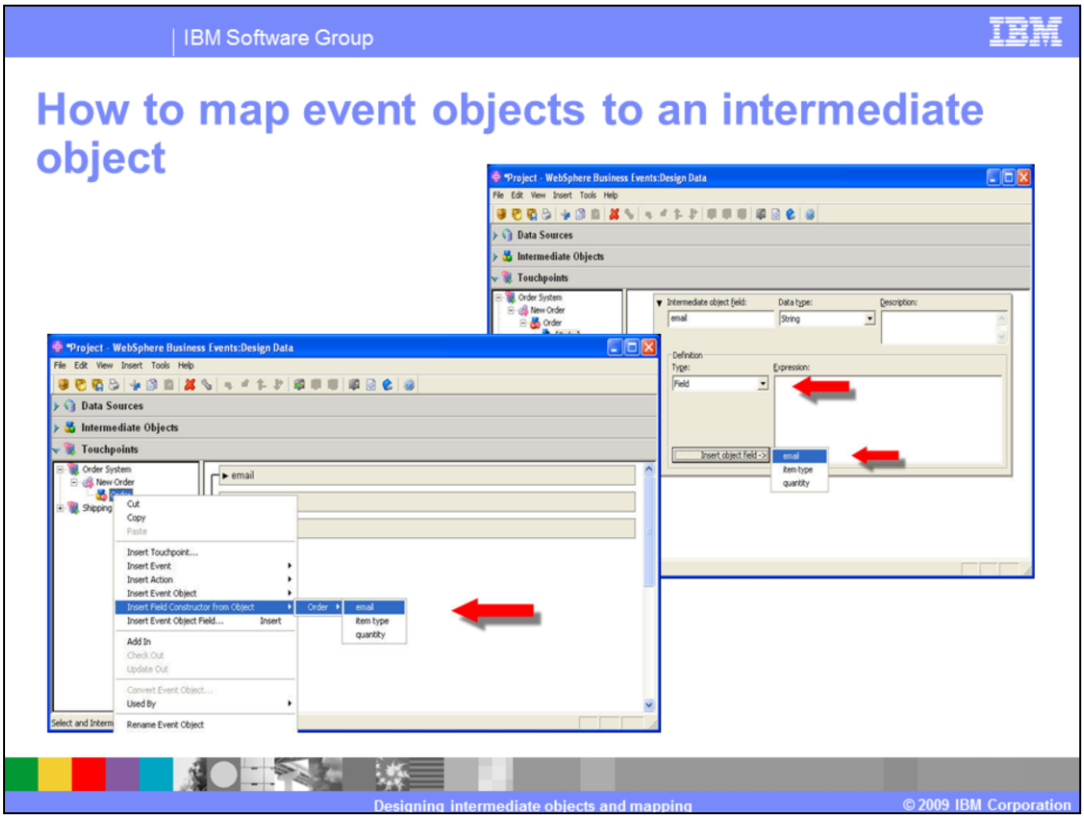
The intermediate object must then be given a name.



An intermediate object needs fields associated with it. These fields define the potential content that the intermediate object can contain. A field is a combination of a name and a data type.

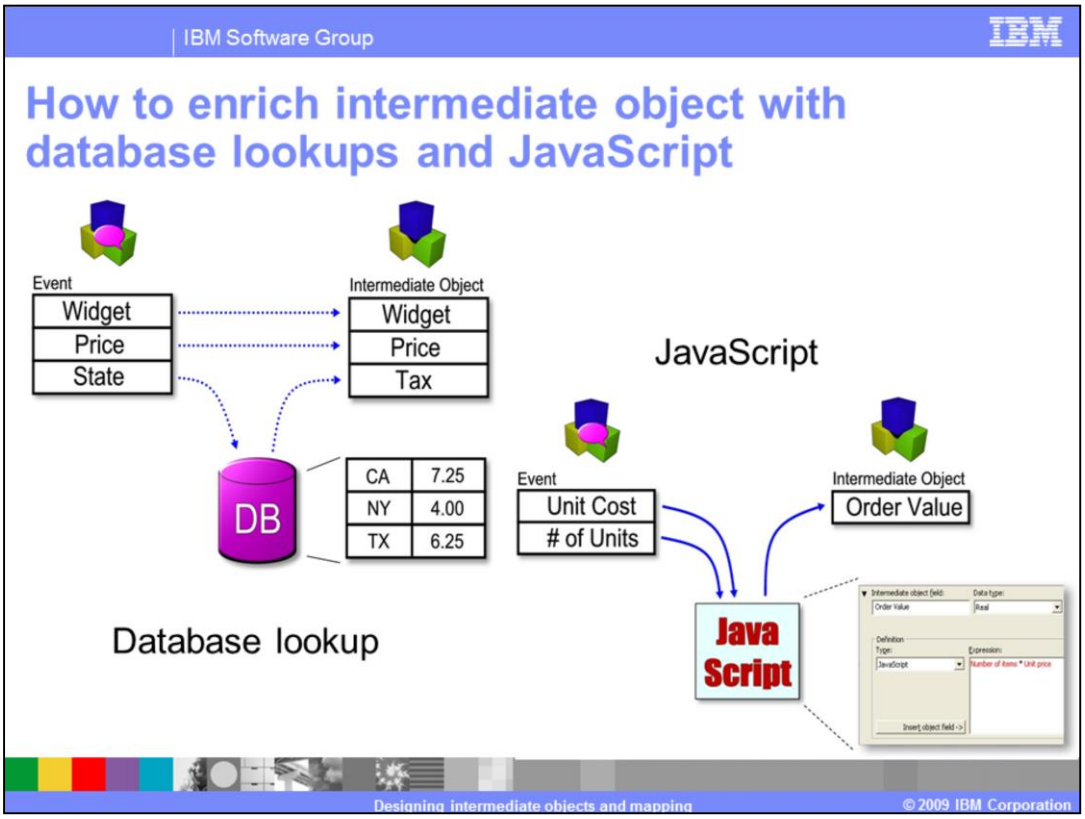
A field can be inserted by right clicking on the intermediate object and selecting the ‘Insert Intermediate Object Field’ menu option.

You should then provide a name for the new field, select the appropriate data type and then optionally provide a description of the field’s usage. Additional fields can be added by repeating this task.



Most of the parts of the solution are now in place. What remains is the mapping of the business data from the event object to the intermediate object fields. In the example shown, there is an event object called order which has three fields defined. Right-click the order event object and from the pop-up menu select ‘Insert Field Constructor from Object’ then select ‘Order’ and finally select ‘e-mail’ from the list.

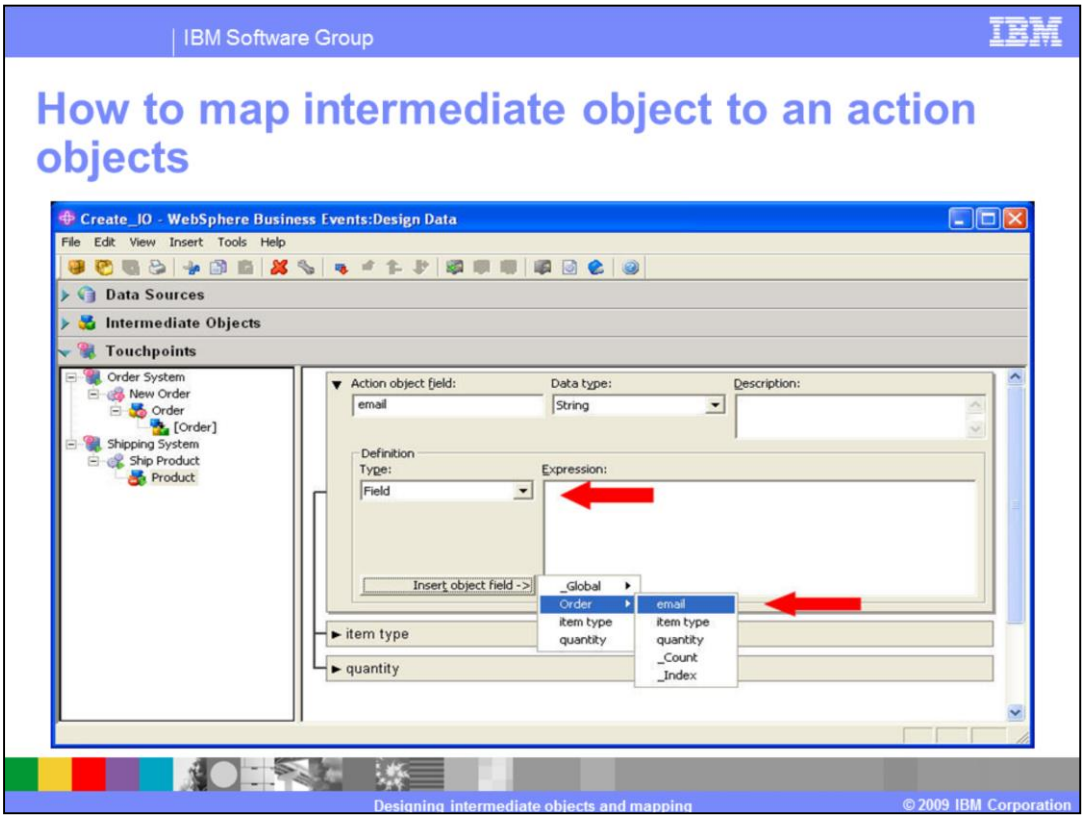
You now map the intermediate object field constructors to the event object fields. Expand the intermediate object field constructor. Select field in the type box and then select the corresponding event object field name as the source of the intermediate object field.



In the event that data is required in order to process an event or action, and this information is not available in the event object, that data can be retrieved from an external data source. Those external data sources fall into two categories: relational databases and remote sources. Relational databases are defined directly to WebSphere Business Events and remote sources are other data sources that are accessed by way of a purpose-written JavaScript API called a custom fetcher. These other remote sources can include web-sites, flat files and non-relational databases.

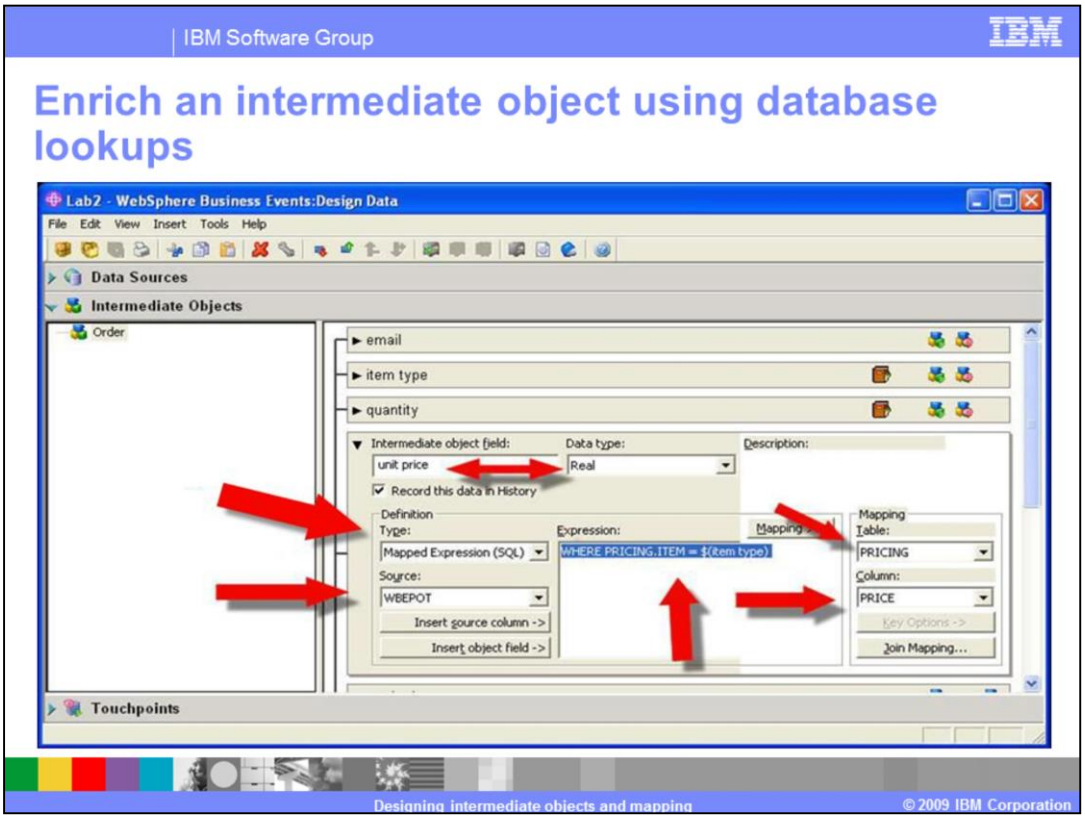
This slide shows some examples of situations where this can be useful. In the first example, a database lookup can be used to populate the tax field in the intermediate object, dependent on the US state field in the incoming event payload. In the second example, JavaScript can be used to multiply two fields in the incoming events payload and populate the order value field in the intermediate object.

How to map intermediate object to an action objects



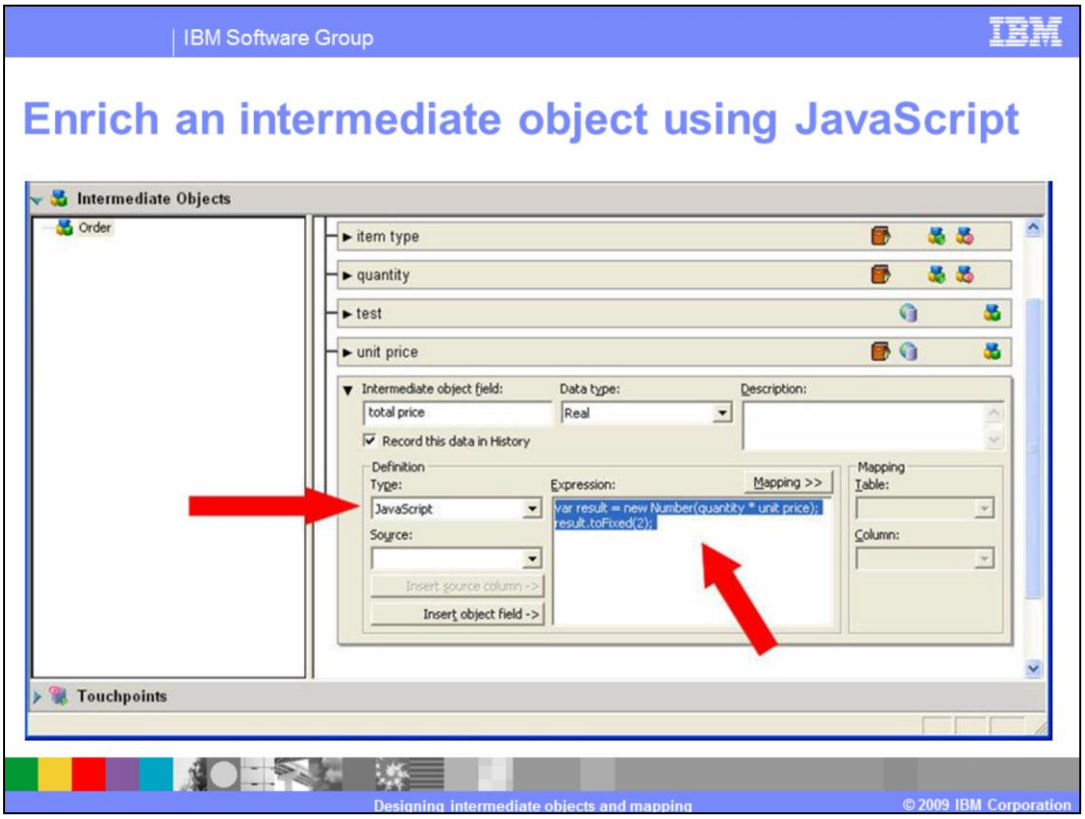
When an action is externalized, data associated with the action object fields has to be populated. This data comes from one or more intermediate objects.

In the example shown, you select the product action object and then click the twisty to expand the e-mail action object field definition. Select ‘field’ in the type pull down list and then click ‘insert object field’ and select the intermediate object field that you want to use to populate this field in the action object.

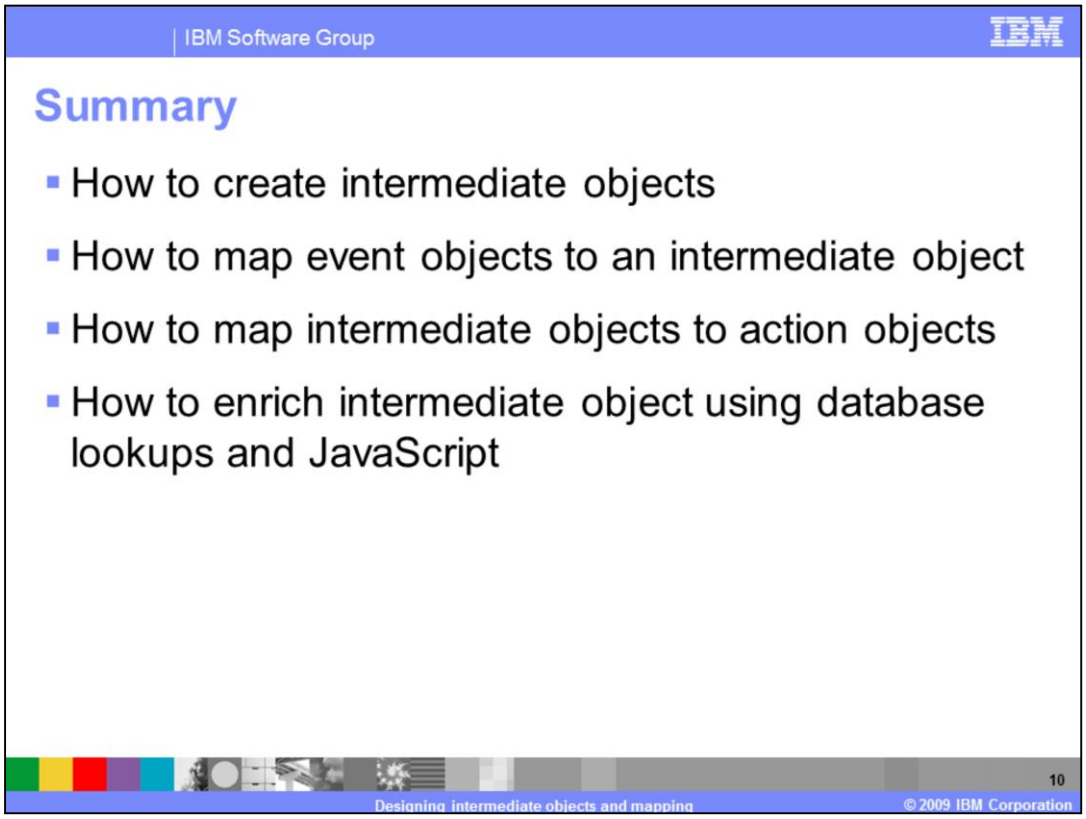


Intermediate objects can be enriched using database lookups. In order to achieve this, a data source must first be defined. To see how to define a data source refer to the demonstration 'Using the RDBMS connector'.

In the example shown, the unit price field in the order intermediate object is populated using a database lookup based on another field in the intermediate object, the item type. There is a database called WBEPOT with a table called PRICING that contains the required data. Expand the definition of the unit price field in the Intermediate Object and set the type to 'Mapped expression (SQL)'.
This defines that the value of the unit price field is to be determined based on a database lookup. Define the source to be WBEPOT. Then in the mapping section of the definition specify that the 'price' column within the PRICING table should be used for the mapping. After completing these steps, the SQL WHERE expression will appear. The SQL WHERE clause can be completed as shown to use a database lookup to return the price for the item specified in the intermediate object item type field.



You can also use JavaScript to enrich an intermediate object. In this example, the total price field in the intermediate object is computed using JavaScript based on two other fields within the intermediate object. To use JavaScript within an intermediate object field definition, set the type to JavaScript and then use the expression field to define how the value of the field should be calculated. In the example shown, the expression calculates the total price by multiplying the quantity by the unit price and rounding the result to two decimal places.



In this presentation you have learnt how to create intermediate objects. You have also learnt how to map event objects to intermediate objects, and intermediate objects to action objects. Finally, you saw how intermediate objects can be enriched using database lookups and JavaScript.

Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

WebSphere

If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of other IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

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

Other company, product, or 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 or changes in the products or programs 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.

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 (for example, 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.