

# **Implementing Your ATG Commerce Solution Rel 10.1**

**Student Guide - Volume I**

D79786GC10  
Edition 1.0  
March 2013  
D81005

**ORACLE®**

**Authors**

Karin Layher

Kerwin Moy

**Technical Contributor  
and Reviewer**

Rick Wilson

**Editors**

Rashmi Rajagopal

Malavika Jinka

**Graphic Designer**

Maheshwari Krishnamurthy

**Publishers**

Sujatha Nagendra

Jayanthy Keshavamurthy

**Copyright © 2013, Oracle and/or its affiliates. All rights reserved.**

**Disclaimer**

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

**Restricted Rights Notice**

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

**U.S. GOVERNMENT RIGHTS**

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

**Trademark Notice**

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

## Contents

### 1 Introduction

- Student Introductions 1-2
- Course Goals 1-3
- Course Agenda: Days One to Three 1-4
- Course Agenda: Days Four and Five 1-5
- Objectives 1-6
- ATG Commerce 1-7
- Flexible Catalog Model 1-8
- Multisite 1-9
- Cross Selling and Upselling 1-10
- SKU Bundles 1-11
- Product Comparisons 1-12
- Gift Lists and Wish Lists 1-13
- Gift Certificates 1-14
- Flexible Pricing 1-15
- Flexible Shipping and Payment 1-16
- Abandoned Order Management 1-17
- Flexible Order Options 1-18
- Commerce-Related Personalization 1-19
- Designed for Integration 1-20
- Commerce Suite Architecture 1-21
- CRS 1-22
- Demonstration: Commerce Reference Store 1-23
- Summary 1-24

### 2 Commerce Reference Store

- Objectives 2-2
- Road Map 2-3
- Basic Commerce Module 2-4
- CRS Modules 2-5
- Store.EStore Module 2-6
- Store.Storefront Module 2-7
- Road Map 2-8
- CIM 2-9
- Product Selection 2-10

Commerce Addons	2-11
Interim Steps	2-12
Switching or Non-Switching Data Sources	2-13
CRS Addons	2-14
Storefront Addons	2-15
Storefront Sample Data	2-16
Additional Options	2-17
Final Steps	2-18
Quiz	2-19
Road Map	2-20
Using CRS for a New Implementation: Steps 1-3	2-21
Using CRS for a New Implementation: Steps 4-6	2-22
Module Dependencies: MyStore.Storefront	2-23
Module Dependencies: MyStore.Custom.Versioned	2-24
Configuring Store-Wide Properties	2-25
Road Map	2-26
Practice Premise	2-27
Taking It Further	2-28
Summary	2-29
For More Information	2-30
Practice 2 Overview: CRS	2-31

### **3 Personalization**

Objectives	3-2
Road Map	3-3
User-Centric Design	3-4
Profile Object: Session Starts	3-5
Profile Object: Implicit Profiling	3-6
Profile Object: User Logs In	3-7
Profile Object Relationships (Personalization Layer Only)	3-8
Design Decision: User Registration	3-9
Quiz	3-10
Road Map	3-11
Grouping Users	3-12
User Directory	3-13
B2C Example: Organizing by Region	3-14
Linking Users to Organizations	3-15
Segments	3-16
Design Decision: Grouping Users	3-17
Road Map	3-18
Commerce Profile Extensions	3-19

B2C-Related Commerce Profile Extensions	3-20
B2B-Related Commerce Profile Extensions	3-21
Extending the Profile Repository	3-22
Adding a Simple Property	3-23
Step 1: Add an Auxiliary Table	3-24
Step 2: Create a Repository Definition Layer	3-25
Adding a Business Object to the Profile: Example	3-27
Step 1: Add Necessary Tables	3-28
Step 2: Add the Property to the User Item	3-29
Viewing Combined Layers	3-30
Combined Layers in the Component Browser	3-31
Design Decision: Profile Extensions	3-32
Quiz	3-33
Summary	3-34
For More Information	3-35
Practice 3 Overview: Extending Personalization for Commerce	3-36

#### **4 Scenarios**

Objectives	4-2
Road Map	4-3
Scenarios	4-4
Scenario Server Architecture	4-5
Scenario Events	4-6
Commonly Used Commerce Scenario Events	4-7
Commonly Used General Scenario Events	4-9
Scenario Actions	4-10
Commonly Used Scenario Actions	4-11
Scenario Conditions	4-12
Scenario: Example	4-13
Extending Scenarios	4-14
Quiz	4-15
Road Map	4-16
Custom Scenario Actions	4-17
Individual Versus Collective Scenario Actions	4-18
Creating Custom Actions	4-19
Coding Custom Actions	4-20
Sources of Information for Actions	4-21
Action Parameters	4-22
Associated Properties File	4-23
Sample Configuration Class and Properties File	4-24
Road Map	4-25

Sample Action Class: getActionName() 4-26
Sample Action Class: initialize() and configure() 4-27
Sample Action Class: executeAction() 4-28
Road Map 4-30
Configuring the Scenario Manager: Defining the Action 4-31
Configuring the Scenario Manager: Configuring Action Parameters 4-32
Configuring the Scenario Manager: Configuring an Action Parameter Value as a Repository Item 4-33
Configuring the Scenario Manager: Configuring the Action Through a Component 4-34
Design Decisions: Scenarios 4-35
Quiz 4-36
Summary 4-37
For More Information 4-38
Practice 4 Overview: Extending Scenarios for Commerce 4-39

## **5 Catalogs**

Objectives 5-2
Road Map 5-3
Product Catalog Repository Item Types: Catalogs and Categories 5-4
Product Catalog Repository Item Types: Linking 5-5
Links Between Catalog-Related Item Types 5-6
Fixed and Dynamic Links 5-7
Catalog Item Types 5-8
Category Item Types 5-9
Selected Category Properties 5-10
Products 5-11
Selected Product Properties 5-12
SKUs 5-13
Selected SKU Properties 5-14
SKU Bundles 5-15
Configurable SKUs 5-16
Common Properties Across Catalog-Related Item Types: General Properties 5-17
Common Properties Across Catalog-Related Item Types: Media Items 5-18
Media Items 5-19
Media Item Properties 5-20
media-external 5-21
media-internal-binary 5-22
media-internal-text 5-23
Road Map 5-24
Ways to Display the Catalog 5-25

Displaying the Catalog (ATG)	5-26
Breadcrumb Trail (ATG)	5-27
Oracle Endeca Experience Manager Components	5-28
Cartridges	5-29
Templates and Cartridges	5-30
Design Decision: Catalog Structure	5-31
Road Map	5-32
Extending the Catalog	5-33
Add Properties to an Existing Item Type	5-34
Create a Subtype of an Existing Item Type	5-35
Create a New Item Type	5-36
Design Decision: Catalog Extensions	5-37
Extending the Catalog: Example	5-38
Item Subtypes	5-39
Extending Catalog Item Types	5-40
1. Create a New Table	5-41
2a. Extend the Type Property	5-42
2b. Define the Subtype Item Descriptor	5-43
3. Add Subtypes to CatalogTools	5-44
Quiz	5-45
Summary	5-46
For More Information	5-47
Practice 5 Overview: Extending the Catalog	5-48

## **6 Multisite Objects and Site Assignment**

Objectives	6-2
Road Map	6-3
Running Multiple Sites	6-4
Multisite: Examples	6-5
Multisite Repository and Item Types	6-6
Multisite Item Types: Site Groups	6-7
Default siteConfiguration Properties: Part One	6-8
Default siteConfiguration Properties: Part Two	6-10
siteConfiguration Properties: DCS Additions	6-11
siteConfiguration Properties: CRS Additions, Part One	6-12
siteConfiguration Properties: CRS Additions, Part Two	6-13
Site Categories	6-14
Road Map	6-15
Customizing Site Repository Items	6-16
Customizing the BCC	6-17
Hiding and Disabling Fields in the BCC	6-18

Road Map	6-19
Site Assignment	6-20
ProfilePropertyPipelineServlet	6-21
Disabling Multisite	6-22
Multisite Responsibilities	6-23
Design Decisions: Multisite	6-24
Summary	6-25
For More Information	6-26
Practice 6 Overview: Customizing and Extending Multisite	6-27

## **7 Multisite and ATG Applications**

Objectives	7-2
Road Map	7-3
New Site Templates	7-4
Site Templates	7-5
Creating a New Site Template	7-6
Creating a New Site Category Item in the BCC	7-7
Setting Default Values	7-8
Road Map	7-9
Ways to Use the Multisite Feature	7-10
Limiting Components by Site Group	7-11
Creating a Component Shareable Type	7-12
Component Shareable Type: Example	7-13
Creating an Abstract Shareable Type	7-14
Ways to Use the Multisite Feature	7-15
Setting Component Properties by Site	7-16
Steps to Configure Component Properties by Site	7-17
Site-Specific Property: Example	7-18
Ways to Use the Multisite Feature	7-19
Using Site Droplets in JSPs	7-20
Accessing the Current Site	7-21
Creating Links to Other Sites	7-22
Retrieving Sites Within a Site Group	7-23
SiteldForItem Droplet	7-24
Ways to Use the Multisite Feature	7-25
Add Listeners to Site Events	7-26
Ways to Use the Multisite Feature	7-27
Add SiteSessionManager Processors	7-28
Ways to Use the Multisite Feature	7-29
Extend Custom Repository Items	7-30
Quiz	7-31

Road Map	7-32
Multisite Extensions to Commerce	7-33
Multisite and Scenarios	7-34
Site-Aware Targeters	7-35
Sites and Catalogs	7-36
Site-Aware Promotions and Coupons	7-37
Summary	7-38
For More Information	7-39
Practice 7 Overview: Creating and Using New Site Categories	7-40

## **8 Orders: Overview**

Objectives	8-2
Road Map	8-3
Orders	8-4
Shopping Cart	8-5
Order Object Versus Order Repository Item	8-6
Order Life Cycle	8-7
The Session Starts	8-8
SKUs Added to the Cart	8-9
(Optional) The User Logs In	8-10
The User Opens the Shipping Page	8-11
(Optional) The User Adds a New Address	8-12
The User Allocates Shipping	8-13
The User Opens the Billing Page	8-14
User Allocates Billing	8-15
(Optional) The User Explicitly Saves the Order	8-16
The User Submits the Order	8-17
(Optional) The User Creates a Scheduled Order	8-18
The Order Is Sent to Fulfillment	8-19
(Optional) The Order Is Held for Approval	8-20
Design Decision: Order Process	8-21
Quiz	8-22
Road Map	8-23
Shipping Methods	8-24
Filtering Shipping Methods	8-25
Design Decisions: Shipping Methods	8-26
Payment Methods	8-27
Design Decisions: Payment Methods	8-28
Summary	8-29
For More Information	8-30

## 9 Extending Orders

- Objectives 9-2
- Road Map 9-3
- Order-Related Components 9-4
- Purchase Process Form Handlers 9-5
- CartModifierFormHandler Handle Methods 9-6
- CartFormHandler Extensions (CRS) 9-7
- Extending Purchase Process Form Handlers 9-8
- Extending CRS Form Handlers 9-9
- Shopping Cart 9-10
- Ways to Access the Current Order 9-11
- Manager Components 9-12
- CommerceItemManager 9-13
- Commerce Items 9-14
- CommerceItem Object 9-15
- OrderManager 9-16
- Helper Components 9-17
- OrderTools Component 9-18
- Road Map 9-19
- Pipeline Basics 9-20
- Commonly Extended Pipelines 9-21
- Pipeline Manager Components 9-22
- Commerce Pipeline Processors 9-23
- Example: ProcessOrder Chain 9-24
- ProcessOrder Chain Processors 9-25
- Customizing a Pipeline: Steps One and Two 9-26
- Customizing a Pipeline: Step Three 9-27
- Editing Pipelines in the ACC 9-28
- Quiz 9-29
- Road Map 9-30
- Modifying the Order Programmatically 9-31
- Steps to Safely Modify an Order 9-32
- Code Examples 9-33
- Road Map 9-34
- Order Object Versus Order Repository Item 9-35
- Order-Related Java Objects 9-36
- Order Repository 9-37
- Mapping of Classes to Repository Items 9-38
- Selected OrderTools Mapping Properties: Java Class to Repository Item Mapping 9-39

Selected OrderTools Mapping Properties: Object Type Mapping	9-40
Order Property “Flow Through”	9-41
Order Persistence	9-42
Extending Order Persistence	9-43
Summary	9-44
For More Information	9-45
Practice 9 Overview: Extending Orders	9-46

## **10 Pricing Architecture**

Objectives	10-2
Road Map	10-3
Pricing Engine	10-4
Pricing Process for Items	10-5
Pricing in Catalog Objects	10-6
Pricing in Price Lists	10-7
Road Map	10-8
Discounts	10-9
Discounts: Examples	10-10
Determining Who Gets a Potential Discount	10-11
Limiting by Distribution	10-12
Granting Individual Promotions to Users	10-13
Limiting by Conditions	10-14
Ways to Limit Promotions	10-15
Discounts Are Linked to Users	10-16
Differential Pricing Methods	10-17
Design Decision: Pricing	10-18
Summary	10-19

## **11 Displaying and Extending Pricing**

Objectives	11-2
Road Map	11-3
Price Lists	11-4
Price List Architecture	11-5
Assigning Price Lists to Users	11-6
Road Map	11-7
Item Pricing Droplets	11-8
Static Pricing Droplets	11-9
PriceDroplet	11-10
PriceRangeDroplet	11-11
Volume Pricing	11-12
ComplexPriceDroplet	11-13

Dynamic Pricing Droplets	11-14
PricelItemDroplet	11-15
PriceEachItemDroplet	11-16
PricingCommerceltem	11-17
PricelInfo Objects	11-18
Base Class: AmountInfo	11-19
ItemPricelInfo	11-20
DetailedItemPricelInfo	11-21
Road Map	11-22
Order Pricing	11-23
OrderPricelInfo	11-24
Road Map	11-25
Objects That Can Be Priced Dynamically	11-26
How Pricing Works	11-27
Ways to Extend Pricing	11-28
Extending the Pricing Engine: Default Classes	11-29
Extending the Pricing Engine: Critical Methods	11-30
Extending Existing Item Pre-Calculators	11-31
Extending Existing Order Calculators	11-32
Adding Pre- or Post-Calculators	11-33
Extending the PricelInfo Class	11-34
Quiz	11-35
Summary	11-36
For More Information	11-37
Practice 11 Overview: Displaying and Extending Pricing	11-38

## **12 Promotions Architecture**

Objectives	12-2
Road Map	12-3
Promotions: Overview	12-4
Promotions in the BCC	12-5
Stacking Rules	12-6
Road Map	12-7
Discount Process	12-8
Review: How Pricing Works	12-9
Gathering the Promotions List	12-10
Evaluating the Promotions	12-11
When a Price Is Requested	12-12
Sorting the Promotions	12-13
Determining the Qualifier Service to Use	12-14
Creating PricingContextObject	12-15

Calculating the Price Adjustments	12-16
Road Map	12-17
Promotion Item Descriptors	12-18
Promotion Repository Item: General Properties	12-19
Promotion Repository Item: Qualification Properties	12-20
Promotion Repository Item: Availability and Template Properties	12-21
Pricing Engine and Promotions	12-22
Pricing Engine Promotion: General Properties	12-23
Promotion-Related Pricing: Map Properties	12-24
Pricing Engine: Component Properties	12-25
Promotion-Related Pricing Engine Methods	12-26
Pricing Engine Stacking Rule Methods	12-27
PricingContext	12-28
Pricing Model Vetoers	12-29
Method Signatures	12-30
PricingModelComparator	12-31
Qualifier Service	12-32
QualifierService as an Example Qualifier Filter	12-33
PromotionQualifyingFilter	12-34
Discount Calculators	12-35
Existing Discount Calculators	12-36
Quiz	12-37
Summary	12-38
For More Information	12-39
Practice 12 Overview: Extending Order Discount Calculators	12-40

## **13 Extending Promotions**

Objectives	13-2
Road Map	13-3
Promotion Relationships	13-4
Promotion Model Definition Language	13-5
Basic PMDL Structure	13-6
Qualifier Tag	13-7
discount-structure Tag: Percent Off Example	13-8
discount-structure Tag: Volume Discount Example	13-9
Target Tag	13-10
Road Map	13-11
Adding Custom Promotion Types	13-12
Create the Pricing Calculator Class	13-13
Understanding a Pricing Method	13-14
pExtraParameters Map	13-15

DiscountStructure Object	13-16
DiscountDetail Object	13-17
Implementing the Pricing Method	13-18
Summary	13-19
For More Information	13-20
Practice 13 Overview: Custom Promotion Calculator	13-21

## **14 Promotion Templates**

Objectives	14-2
Road Map	14-3
Promotion Templates	14-4
Default Promotion Templates by Type	14-5
Example: BOGO Item Discount	14-6
Example: Tiered Promotion	14-7
GWP Promotion: Example	14-8
Advanced Promotions	14-9
Road Map	14-10
Promotion Template Relationships	14-11
Promotion Template Definitions	14-12
Linking UI Elements to Promotion Properties	14-13
Road Map	14-14
Promotion Template Example: Get Item Discount	14-15
Template Name and Priority	14-16
Screen Segments and Lines	14-17
Defining a Line	14-18
Defining Radio Buttons	14-20
Including and Excluding Product Set Criteria	14-21
Defining Combo Boxes	14-22
Additional Template UI Elements	14-23
Road Map	14-24
Simple Validation with Tag Attributes	14-25
Complex Validation	14-26
Multielement Translators	14-27
Multielement Translator: Example	14-28
multi-element-translators Tag	14-29
Multielement Translator: Example	14-30
PSC Multielement Translator	14-33
Default Translator Components	14-34
Writing a Translator Component Class	14-35
Road Map	14-36
Setting Promotion Properties	14-37

Setting Simple Promotion Properties	14-38
Setting the pmdlRule Property	14-39
Setting the pmdlRule Property Example: Part One	14-40
Setting the pmdlRule Property Example: Part Two	14-41
Setting the pmdlRule Property Example: Part Three	14-43
Including Closeness Qualifiers	14-44
Example: Including Closeness Qualifiers	14-45
Changing or Deleting Templates	14-46
Summary	14-47
For More Information	14-48
Practice 14 Overview: Creating a Custom Promotion Template	14-49

## **15 Shipping and Shipping Groups**

Objectives	15-2
Road Map	15-3
Shipping Examples	15-4
Example A: The Shopper Opens the Shipping Page from the Cart Page on a New Order	15-5
ShippingGroupDroplet (SGD) Is Triggered	15-6
SGD Clears ShippingGroupContainerService (SGCS)	15-7
SGD Calls Initializer Components to Determine the Valid Shipping Groups	15-8
HardgoodShippingGroupInitializer	15-9
SGD Populates Shipping Groups	15-10
Default Initializer Components	15-11
SGD Populates CommercelItemShippingInfos (CISIs)	15-12
CommercelItemShippingInfo	15-13
SGCS Data Is Used to Populate the Shipping Page	15-14
Determining the Available Shipping Methods in the Shipping JSP	15-15
AvailableShippingMethods Droplet	15-16
The Shopper Selects Shipping Options	15-17
Shipping Options Are Changed in the Order	15-18
Example B: The Shopper Returns to the Shipping Page in an Order	15-19
Shipping Information Is Not Cleared	15-20
ShippingGroupDroplet Input Parameters	15-21
ShippingGroupDroplet Output and Open Parameters	15-22
Quiz	15-23
Road Map	15-24
Adding New Shipping Groups	15-25
Splitting Shipping	15-26
ShippingInfoFormHandler	15-27
Example C: The Shopper Ships the Same Item to Multiple Addresses	15-28

Splitting Shipping and CISIs: During the Split	15-29
Splitting Shipping and CISIs: After the Split	15-30
Road Map	15-31
Shipping Groups	15-32
Existing Shipping Group Types	15-33
ShippingGroupImpl Properties	15-34
Additional Shipping Group Properties	15-35
ShippingGroupManager	15-36
Road Map	15-37
Steps to Add a Custom Shipping Group Type	15-38
Creating a ShippingGroupInitializer	15-40
Defining the ShippingGroupInitializer Methods	15-41
Summary	15-42
For More Information	15-43
Practice 15 Overview: Creating a Custom Shipping Group	15-44

## **16 Pipelines and Shipping Methods**

Objectives	16-2
Road Map	16-3
Review: Pipeline Basics	16-4
Invoking a Pipeline	16-5
Calling the runProcess() Method of Pipeline Links	16-6
Pipeline Arguments	16-7
Default Subclasses of ValidatePipelineArgs	16-8
Pipeline Result Object	16-9
Moving to the Next Link in a Chain	16-10
Ending the Pipeline	16-11
Pipeline Manager Components	16-12
Extending a Pipeline	16-13
PipelineProcessor Class	16-14
Example: Extending a Pipeline	16-15
Shipping Validator Example: Resource Bundle and Return Code	16-16
Shipping Validator Example: runProcess and Error Messages	16-17
Shipping Validator Example: Returning SUCCESS	16-18
Extending the validateShippingGroup Chain	16-19
Adding the Return Code and Shipping Group Type	16-20
Road Map	16-21
Shipping Methods	16-22
Shipping Calculator Class Hierarchy	16-23
ShippingCalculatorImpl Properties and Methods	16-24
Existing Shipping Calculators	16-25

Creating a New Shipping Method	16-27
Restricting Existing Shipping Methods	16-28
Quiz	16-29
Summary	16-30
For More Information	16-31
Practice 16 Overview: Customizing Shipping Methods	16-32

## **17 Payment Groups**

Objectives	17-2
Road Map	17-3
Payment Architecture	17-4
Default Payment Collection Process	17-6
CommerceldentifierPaymentInfo	17-7
CRS Payment Collection Process	17-8
BillingInfoFormHandler Methods	17-9
BillingInfoFormHandler Pre and Post Methods	17-10
BillingFormHandler Handle Methods (CRS Only)	17-11
Road Map	17-12
Payment Groups	17-13
Payment Group Class Hierarchy	17-14
CreditCard Properties	17-15
GiftCertificate and StoreCredit Properties	17-16
Payment Manager Classes	17-17
Road Map	17-18
Claimable Payment Groups	17-19
Claimable Repository Item Descriptors	17-20
Claimable Item Descriptor	17-21
GiftCertificateClaimable and StoreCreditClaimable Item Descriptors	17-22
The PromotionClaimable Item Descriptor and the BatchPromotionClaimable Sub-Type	17-23
Road Map	17-24
Adding New Payment Groups	17-25
Changing Allocated Amounts	17-26
PaymentGroupFormHandler	17-27
PaymentGroupFormHandler Example	17-28
Road Map	17-29
Payment Process	17-30
Payment Operation Sequence	17-31
Road Map	17-32
Steps to Add a Custom Payment Group Type	17-33
Summary	17-36

For More Information 17-37

Practice 17 Overview: Creating a Custom Payment Group Type 17-38

## **18 Other Purchase Process Form Handlers**

Objectives 18-2

Road Map 18-3

CancelOrderFormHandler 18-4

CancelOrderFormHandler: Properties and Handlers 18-5

CommitOrderFormHandler 18-6

Road Map 18-7

ExpressCheckoutFormHandler 18-8

Road Map 18-9

Saving Incomplete Orders 18-10

SaveOrderFormHandler 18-11

ShoppingCart Component 18-12

Road Map 18-13

CouponFormHandler 18-14

Summary 18-15

## **19 Inventory and Fulfillment**

Objectives 19-2

Road Map 19-3

Inventory 19-4

Inventory Manager 19-5

Default Inventory Implementation: Architecture 19-6

Default Inventory Implementation: Inventory Item Type 19-7

Default Inventory Implementation: Updating Inventory 19-8

Customizing Inventory Management 19-9

Option 1: Modify inventory.xml 19-10

Option 2: Model a Custom Inventory Repository 19-11

Option 3: Modify the InventoryManager Class 19-12

Example: Localizing Inventory Manager 19-13

Option 4: Write a New InventoryManager Class 19-14

Design Decision: Inventory 19-15

Road Map 19-16

Default Fulfillment Process: Submitting the Order 19-17

Default Fulfillment Process: Notifying Fulfillment 19-18

Default Fulfillment Process: Processing the Order 19-19

Default Fulfillment Process: Sending FulfillOrderFragment Messages 19-22

Default Soft Good Fulfillment Process 19-23

Design Decision: Soft Goods 19-24

Default Hardgood Fulfillment Process: Picking Stock	19-25
Default Hardgood Fulfillment Process: Shipping a Shipping Group	19-26
Integrating with a Shipping System: Options for Notifying the Shipping System	19-27
Integrating with a Shipping System: Options for Communicating with ATG	19-28
Design Decisions: Hard Goods	19-29
Default Payment Processing: Notifying OrderFulfiller of Shipment	19-30
Default Payment Processing: Calling the PaymentManager.debit() Method	19-31
Options for Customizing Fulfillment	19-32
Fulfillment Options	19-33
Design Decisions: Fulfillment	19-34
Summary	19-35
For More Information	19-36
Oracle Commerce Technical Training	19-37

## **Appendix A: Displaying the Catalog and Breadcrumbs**

Objectives	A-2
Road Map	A-3
Catalog Browsing Flow	A-4
Top-Level Categories: Non-CRS	A-5
Top-Level Categories: CRS	A-6
Drilldown Pages	A-7
Commerce Lookup Droplets	A-8
ItemLookupDroplet	A-9
Links to Categories and Products	A-10
Links to Categories and Products: Non-CRS	A-11
Links to Categories and Products: CRS	A-12
CatalogItemLink	A-13
ProductLookupItemLink	A-14
Example: CatalogItemLink	A-15
Road Map	A-16
Breadcrumb Trail	A-17
Key Breadcrumb Trail Objects	A-18
CatalogNavHistory Component	A-19
Breadcrumb Trail In Action: Step One	A-20
Breadcrumb Trail In Action: Clicking a Link	A-21
Example: Maintaining the Stack	A-22
Breadcrumb Trail In Action: Incrementing NavCount	A-23
Breadcrumb Trail In Action: Drilling Down to a Subcategory	A-24
Breadcrumb Trail In Action: Subcategory	A-25
Breadcrumb Trail In Action: Breadcrumb Display	A-26

Breadcrumb Trail In Action: Pop navAction A-27  
Using the Back Button: Part One A-28  
Using the Back Button: Part Two A-29  
Using the Back Button: Part Three A-30  
Jumping to Products or Categories A-31  
Example: Jumping to Products or Categories A-32  
Summary A-33  
For More Information A-34

## **Appendix B: Additional Multisite Information**

Objectives B-2  
Road Map B-3  
SiteContextPipelineServlet B-4  
SiteContextRuleFilters B-5  
Out-of-the-Box Rule Filters B-6  
RequestParameterRuleFilter B-7  
URLPatternMatchingRuleFilter B-8  
DefaultSiteContextRuleFilter B-9  
ProfilePropertyServlet B-10  
ProfilePropertySetters B-11  
Out-of-the-Box Property Setters B-12  
SiteProfilePropertySetter B-13  
CatalogProfilePropertySetter B-14  
PriceListPropertySetter B-15  
StoreProfilePropertySetter B-16  
SiteSessionEventTrigger B-17  
SiteSessionEventSender B-18  
SiteVisitManager B-19  
Disabling Multisite B-20  
Road Map B-21  
Creating a New Site Template B-22  
Creating a New Item Mapping B-23  
Item Mappings B-24  
Item Views B-25  
Property View Mappings B-26  
Property Views B-27  
siteConfiguration Item Mapping B-28  
Item Mapping to Item View Mapping B-29  
Item View Mapping to Item View B-30  
Property View Mapping B-31

Property View B-32  
For More Information B-33

## **Appendix C: Advanced Promotions Topics**

Objectives C-2  
Road Map C-3  
Promotion Import and Export API C-4  
Import and Export Templates C-5  
Importing and Content Administration C-6  
Configuring PublishingWorkflowAutomator Properties C-7  
PromotionImportExport Component C-8  
PromotionImportExportTools C-9  
Import Batch Size C-10  
Info Classes C-11  
PromotionImportExportInfo Properties: Part One C-12  
PromotionImportExportInfo Properties: Part Two C-13  
ClosenessQualifierImportExportInfo Properties C-14  
CouponImportExportInfo Properties C-15  
Accessing the PromotionImportExport Component C-16  
Starting and Ending Import and Export Sessions C-17  
Import Code C-18  
Exporting Promotions by ID C-19  
Exporting Promotions by RQL Query C-20  
Integrator Components C-21  
Integrator Components and PromotionImportExport Methods C-22  
Road Map C-23  
Extending the PMDL C-24  
Attribute Tag C-25  
Generic Expression Tags C-26  
Creating Custom Expression Tags C-27  
Example pmdlParser Mapping C-28  
Older ATG Version Expression Tags C-29  
Mapping PMDL Elements to Java Objects C-30  
Road Map C-31  
Gift with Purchase (GWP) Promotions: Overview C-32  
Default GWP Templates C-33  
Example: Spend Y Get GWP Template C-34  
Gift Pricing in the Shopping Cart C-35  
GWPManger Component C-36  
Storing Information About GWP Promotions C-37

Order Repository Extensions: gwpOrderMarker and gwpCommerceItemMarker	C-38
gwpOrderMarker Item Descriptor	C-39
giftWithPurchase Event Attribute	C-41
Pricing with the GWPManager	C-42
GWP Droplets and Form Handlers	C-43
GiftWithPurchaseSelectionsDroplet	C-44
GiftWithPurchaseSelectionChoicesDroplet	C-45
GiftWithPurchaseFormHandler	C-46
How the Droplets and Form Handler Work Together	C-48
Commerce Reference Store GWP Extensions	C-49
Road Map	C-50
Stacking Rules: Overview	C-51
Pricing Engine Maps Related to Stacking Rules and Included and Excluded Promotions	C-52
Pricing Engine Updates: setupStackingRuleCollection() Method	C-53
DID_PROMOTION_APPLY Parameter	C-54
PricingEngineService Methods Related to Stacking Rules	C-55
Pricing Engines and the applyPromotions() Method: Part One	C-56
Pricing Engines and the applyPromotions() Method: Part Two	C-57
Additional Stacking Rules Information	C-58
Road Map	C-59
Application Messaging: Stacking Rules and GWP	C-60
PricingTools Messaging Properties	C-61
MessageTools Properties	C-62
Messages	C-63
Road Map	C-64
When to Implement CalculatorInfoProvider	C-65
The CalculatorInfo Object	C-66
Constructing a CalculatorInfo Object	C-67
Extracting Values from Custom Attributes	C-68
Summary	C-69
For More Information	C-70

# 1

## Introduction

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Student Introductions

Please introduce yourself to the class by answering these questions:

- What is your name?
- Where do you work?
- What job role do you perform?
- What is your level of experience with Oracle ATG Web Commerce (ATG Commerce)?
- Which ATG courses have you taken, and when?
- How does your organization use or plan to use ATG Commerce?



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Course Goals

After completing this course, you should be able to:

- Describe the architecture of ATG Commerce
- Use Commerce Reference Store as a basis for new Commerce implementations
- Add properties to user profiles and catalog objects
- Describe and extend the classes used to work with orders
- Implement and customize a multisite architecture
- Implement a custom pricing calculator
- Add a custom promotion type and template
- Configure and extend shipping and payment methods
- Describe the various purchase process form handlers and their use



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Course Agenda: Days One to Three

Session	Lesson
Day One	1: Introduction 2: Commerce Reference Store 3: Personalization 4: Scenarios
Day Two	5: Catalogs 6: Multisite Objects and Site Assignment 7: Multisite and ATG Applications 8: Orders: Overview
Day Three	9: Extending Orders 10: Pricing Architecture 11: Displaying and Extending Pricing 12: Promotions Architecture



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Course Agenda: Days Four and Five

Session	Lesson
<b>Day Four</b>	13: Extending Promotions 14: Promotion Templates 15: Shipping and Shipping Groups
<b>Day Five</b>	16: Pipelines and Shipping Methods 17: Payment Groups 18: Other Purchase Process Form Handlers 19: Inventory and Fulfillment



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to describe the features of ATG Commerce.

## ATG Commerce

- Is a framework for building large-scale commerce sites
- Includes components to easily configure all basic commerce functionality, such as:
  - Product catalogs
  - Shopping carts
  - Pricing
  - Purchase process
- Includes many advanced features as well



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The advanced features are discussed in greater detail in subsequent slides.

## Flexible Catalog Model

- The flexible catalog hierarchy helps you to customize the catalog to your needs.
  - Products can be linked to multiple categories.



- Subcategories can be linked to multiple parent categories and catalogs.
- Multiple catalogs allow you to create different navigational structures for different groups of shoppers, times of year, or sites.

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

# Multisite

- Multisite:
  - Allows multiple store sites to run from one integrated ATG cluster
  - Simplifies server architecture and content management
- How much sites share is easily defined and customized.
  - Examples:
    - Shopping cart
    - Product catalog
    - Price list

Store : ATG Store | ATG Home

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The ATG Commerce Reference Store shows examples of multisite functionality with separate sites for clothing and furniture.

## Cross Selling and Upselling

- Both products and categories can be linked to related items.
  - Items are manually selected or dynamically rendered based on defined criteria, such as “shirts under \$50.”
- Products can also have a separate list of upsell products.



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## SKU Bundles

- SKU bundles allow stores to group SKUs for sale.
- Shoppers see the bundle as a single item.
- The inventory and fulfillment systems handle the bundle as individual SKUs.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Note

The example shown in the screenshot is not part of the standard CRS sample data.

## Product Comparisons

	
<b>Suede Blazer</b>	<b>Sponged Wool Jacket</b>
\$84.00 - \$99.00	\$235.00
<a href="#">View Details</a>	<a href="#">View Details</a>
<a href="#">Remove</a>	<a href="#">Remove</a>
<b>Colors:</b> Black / Blue / Olive	<b>Colors:</b> Maroon
<b>Sizes:</b> S / M / L / XL / XXL	<b>Sizes:</b> S / M / L
<b>Features:</b>  	<b>Features:</b> Wool, Recycled

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

# Gift Lists and Wish Lists

## Gift List



### Wedding

Kim Anderson   Wedding   8/20/2014  
Upcoming wedding gifts.

Site	Item	Price	Wants	Needs	Qty	
ATG HOME	 Jersey Wood Table Finish: Cherry xsku2095_1 In Stock	\$279.00	1	1	0	<button>Add to Cart</button>
ATG HOME	 Jack and Jill Glass Tumbler xsku2074 In Stock	\$12.00	2	2	0	<button>Add to Cart</button>

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

# Gift Certificates

	\$50 Gift Certificate Item <i>Cannot Be Gift Wrapped</i>	Price: \$50.00	<input type="text" value="0"/>
	\$100 Gift Certificate Item <i>Cannot Be Gift Wrapped</i>	Price: \$100.00	<input type="text" value="0"/>
<b>Complete Your Gift Certificate Order</b> Required Fields *			
Recipient Name: *	<input type="text"/>		
Recipient Email Address: *	<input type="text"/>		
(example: name@domain.com)			
Sender Name: *	<input type="text"/>		
Sender Email Address: *	<input type="text"/>		
(example: name@domain.com)			
Email Subject:	<input type="text"/>		
Your Message:	<input type="text"/>		

**ORACLE**

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ATG Commerce includes the ability to sell and deliver gift certificates as an "electronic" or "soft" good. The gift certificates are delivered by email as a unique code; recipients use this code on the checkout pages to redeem their certificates.

## Flexible Pricing

- Prices can be set at the SKU or product level.
- It is simple to implement different kinds of pricing, such as:
  - Differential pricing between groups
  - Multiple currencies
  - Static sale prices
  - Dynamically calculated discount offers



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Differential pricing between groups and pricing based on multiple currencies are configured through price lists.

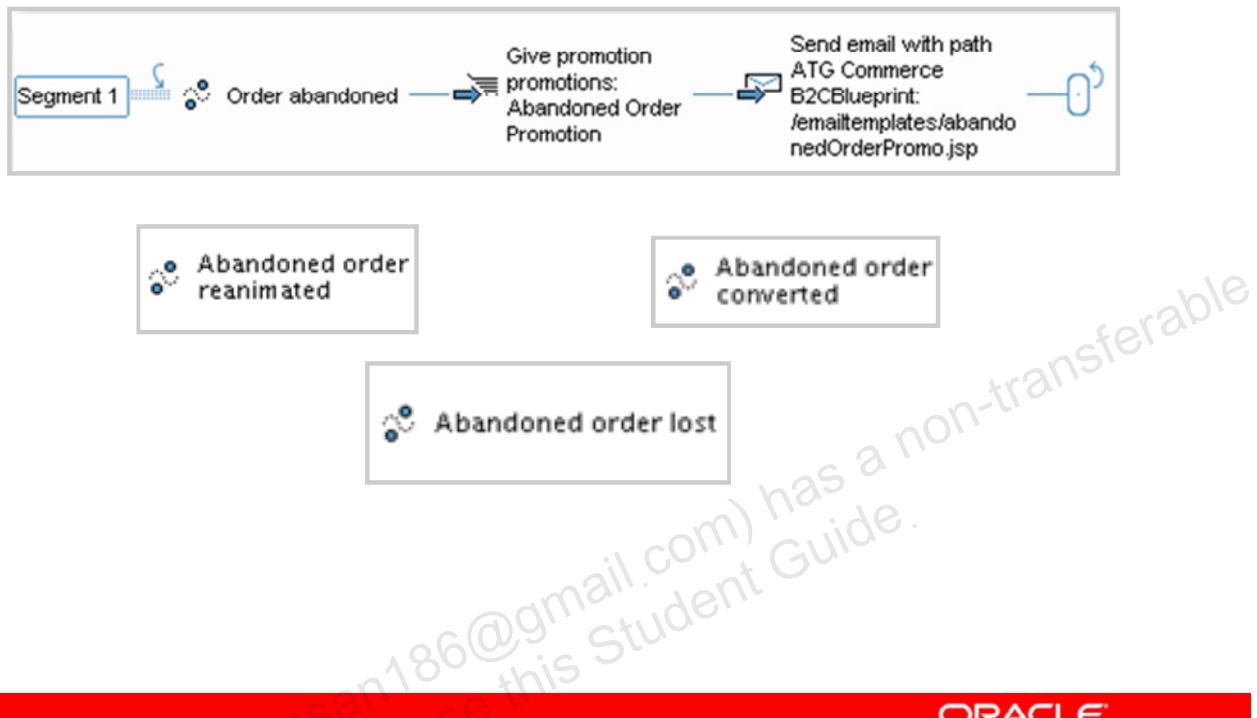
## Flexible Shipping and Payment

- Shipping methods included out of the box are:
  - Fixed price
  - By weight
  - By price
- Payment methods included out of the box are:
  - Credit card, gift certificate, store credit, purchase order
- It is easy to add custom shipping or payment methods.
  - Examples: Ship to store, pay by Paypal
- Stores can allow shoppers to split the shipping or billing of a single order.

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

# Abandoned Order Management



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

By default, an incomplete order is marked as abandoned 30 days after the last order activity. This setting can be adjusted. Incomplete orders, out of the box, are saved only for registered users.

## Flexible Order Options

If desired, stores can allow users to:

- Explicitly save incomplete orders
  - Registered shoppers can save any number of incomplete orders.
  - Shoppers can assemble and submit other orders, and then come back to a saved order.
- Schedule recurring or delayed orders
  - Example: “Thirty widgets on the first of every month”



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

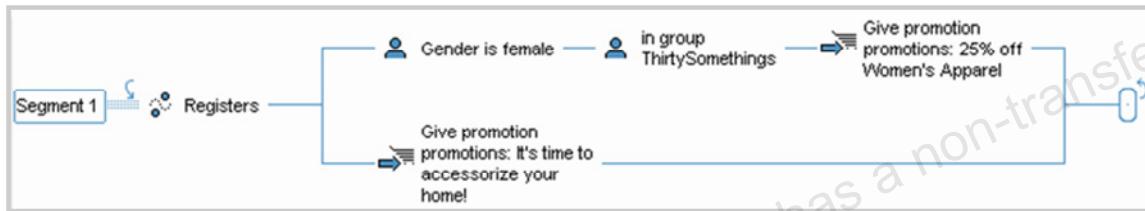
By default, registered users have their current, incomplete order saved automatically. This order becomes their current order when they return to the site. The saved order option allows users to explicitly save an incomplete order so that it can be put aside while they work on another order.

**Note:** Allowing shoppers to save or schedule orders is not currently implemented in CRS.

## Commerce-Related Personalization

Commerce adds many events and actions to the ATG Scenario Engine.

- Example events: Item added to order, Gift purchased
- Example actions: Add item to order, Revoke promotion



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full list of Commerce-related events and actions will be presented in a later lesson. In the scenario example in the slide, women in the ThirtySomethings group receive both promotions. All others get only the "It's time to accessorize your home" promotion.

## Designed for Integration

- ATG Optimization
  - Oracle Recommendations, Click to Call/Chat
- ATG Commerce Service Center (CSC)
- Business intelligence component that integrates with Oracle Business Intelligence (OBI)
- Third-party payment processors
  - Payflow Pro, Cybersource, Taxware
- Third-party inventory and shipping systems

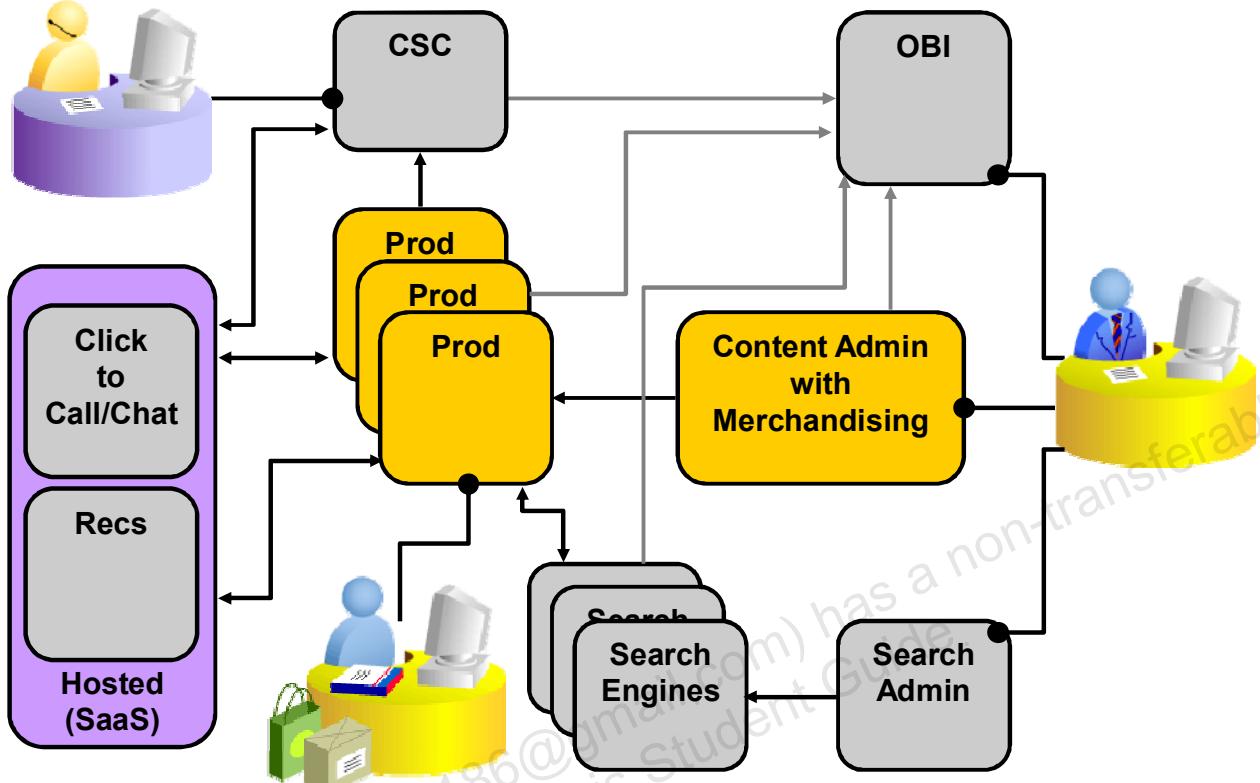


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The products listed in the slide are examples of common integrations. The ATG Commerce Reference Store demonstrates a sample integration with Oracle Recommendations. The ATG Commerce Service Center (CSC) is built on top of ATG Commerce, and you can easily add your Commerce extensions to CSC. When you purchase an ATG product, the information-gathering component of business intelligence is included (data warehouse loaders and the data warehouse). You can easily extend these to include custom information. You can also purchase Oracle Business Intelligence Foundation Suite or Oracle Business Intelligence Enterprise Edition for easy reporting by using the data collected.

ATG Commerce can be extended to integrate with virtually any system.

# Commerce Suite Architecture



**ORACLE**

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

This slide shows a very high-level overview of the architecture of a full Commerce site. Every site is required to have production servers running ATG Commerce and at least one Content Administration server running ATG Merchandising. Optionally, sites may also include:

- **Commerce Service Center (CSC):** Allows call center agents to assist ATG Commerce customers with completing or editing orders; can also be customized for use in-store by sales agents or customers.
- **Oracle Business Intelligence Foundation Suite for Oracle Applications (OBI) or Oracle Business Intelligence Enterprise Edition (OBIEE):** Provides business analysis and reporting; the production, CA, Search, and CSC servers all provide information for analysis. The OBI license restricts use of OBIEE to ATG data. You can also use OBIEE, which is not restricted to just ATG data, and can be used enterprise-wide.
- **Click-to-Call/Chat:** Allows customers to initiate a phone call or chat session with an agent; can be integrated with CSC to automatically transfer customer data to the CSC window.
- **Oracle Recommendations (Recs):** Uses real-time data to provide personalized product recommendations

Note that your organization may not have access to all of these applications. It depends on which products you have purchased.

# CRS

- CRS is a reference Business-to-Commerce (B2C) application that:
  - Implements best practices
  - Is designed to be starting point for new Commerce implementations
- The CRS product is included in the ATG Commerce purchase.
- CRS is installed separately in its own module.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Demonstration: Commerce Reference Store

The red bar spans most of the width of the slide, positioned above the copyright notice.

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Summary

In this lesson, you should have learned how to describe the features of ATG Commerce.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

As a review, ATG Commerce:

- Is a framework for building and deploying commerce sites
- Provides many advanced and flexible features
- Can be easily integrated with other ATG or third-party products

CRS is available as a base for new commerce implementations.

## Commerce Reference Store

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to:

- Describe the structure and use of the Commerce Reference Store (CRS) modules
- Configure CRS by using the Configuration and Installation Manager (CIM)
- Use CRS as the basis for a new ATG Commerce implementation

# Road Map

- Commerce-related modules
- Using CIM to configure CRS
- Using CRS for new implementations
- Premise for the course practices



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Basic Commerce Module

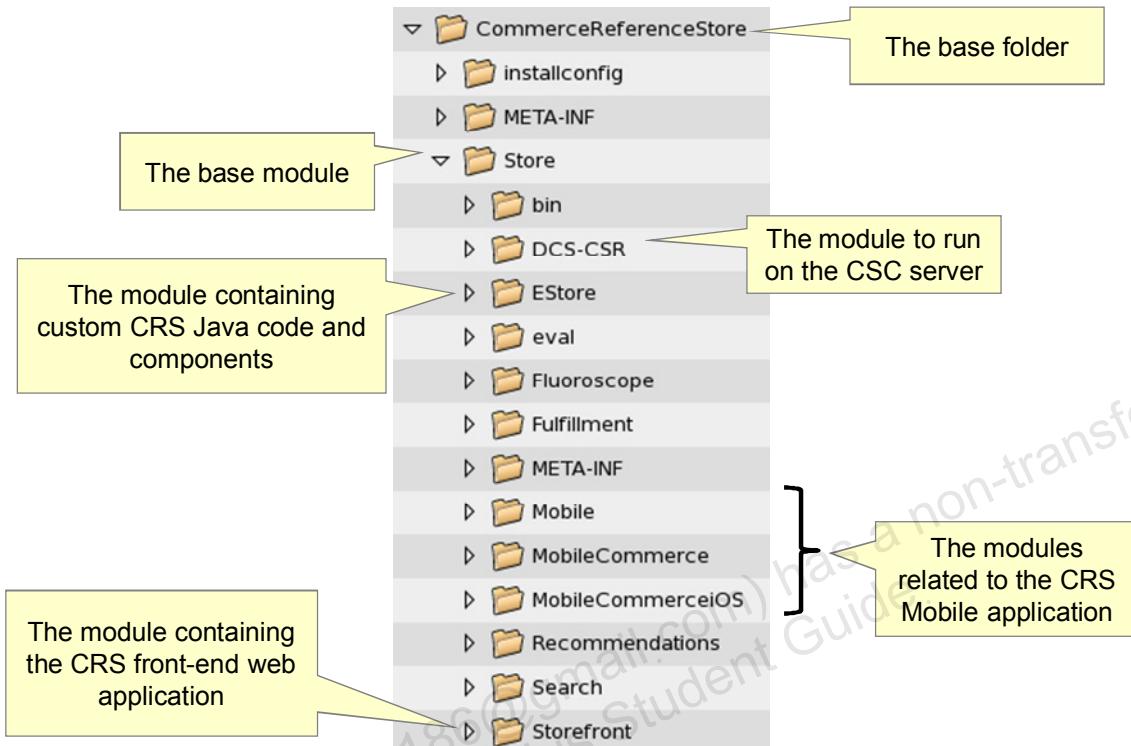
- The DCS module contains Commerce functionality.
  - The *atgDir/DCS/src/Java* folder contains selected source code for Commerce classes.
- Two additional modules are there for backwards compatibility only and should not be used for new sites:
  - B2CCommerce includes configuration and components specific to B2C sites.
  - B2BCommerce includes configuration and components specific to B2B sites.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Prior to ATG 10.1, the Commerce functionality was split between two separate modules, B2CCommerce and B2BCommerce. In ATG 10.1, the B2C and Business-to-Business (B2B) functionality were merged into the DCS module. Any new sites should be developed by using the DCS module. If you are an existing customer, the old modules still exist to help you as you convert to ATG 10.1. Eventually, you will need to migrate to the DCS module. Both the B2CCommerce and B2BCommerce modules depend on DCS, but override any changes from the code merge.

# CRS Modules



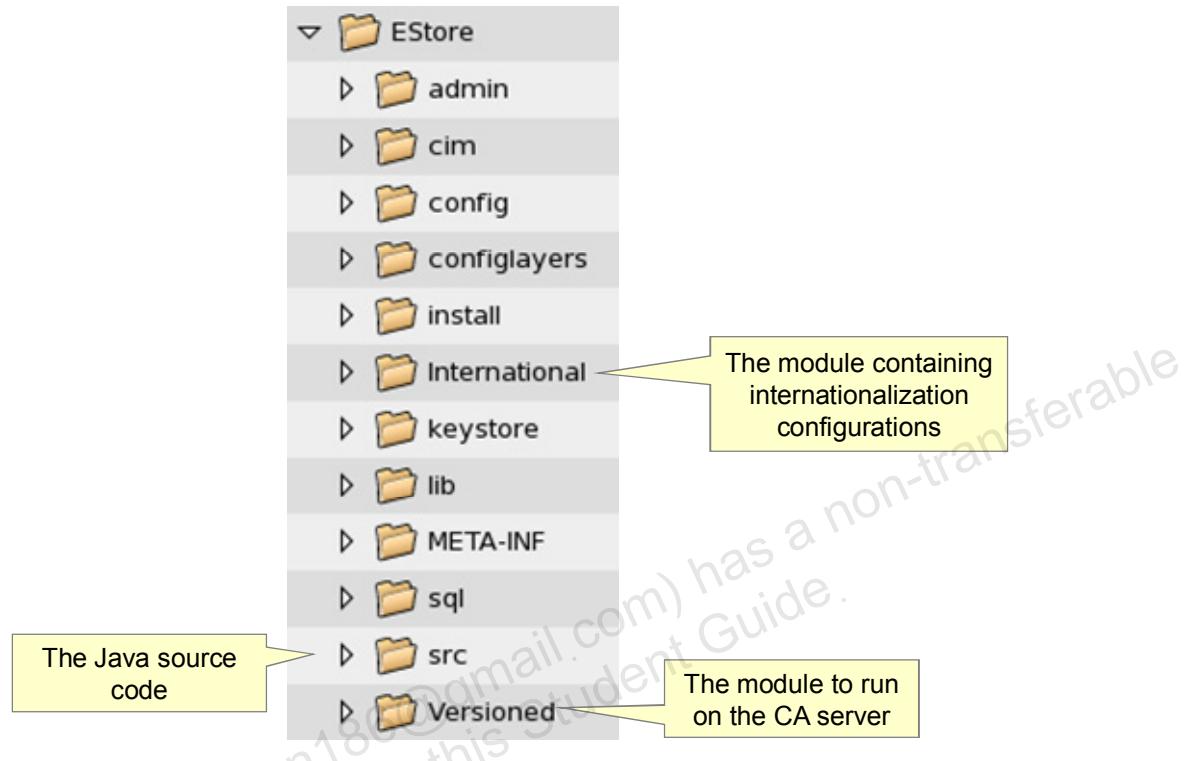
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

For more information on the mobile versions of CRS, refer to the associated white papers in the My Oracle Support (MOS) knowledge base, such as:

- Creating Mobile Web Applications with Oracle ATG Web Commerce: A Guide to the Mobile Commerce Reference Store
- Creating Native iPhone Applications with Oracle ATG Web Commerce: A Guide to the Mobile Commerce Reference Store

## Store.EStore Module



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The Store.EStore module contains the custom Java source files and components. Some key submodules and folders include the following:

- **International:** This module contains the internationalized functionality. It is necessary for CRS sites that support multiple languages or multiple countries. You can choose this option when configuring your environment in CIM.
- **src:** This folder contains the Java source code for the CRS application.
- **Store.EStore.Versioned:** This is the module that you run on your Content Administration (CA) server for Merchandising.

# Store.Storefront Module



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The Store.Storefront module contains the web application for the CRS front end.

## Road Map

- Commerce-related modules
- Using CIM to configure CRS
- Using CRS for new implementations
- Premise for the course practices



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# CIM

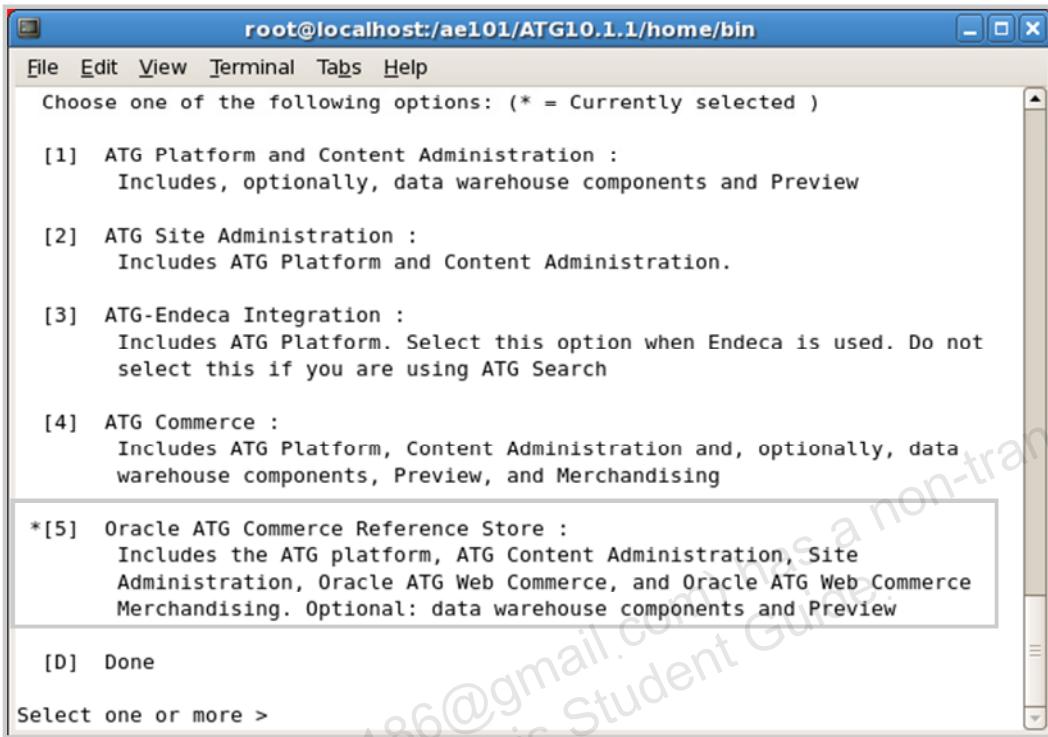
- CIM is a utility to simplify the process of configuring your ATG Commerce store.
- A series of text-based wizards guide you through configuration procedures, performing the correct steps in the correct order.
- For CRS, CIM takes you through the following steps:
  - Selecting the products
  - Configuring your data source
  - Creating database tables and optionally importing data
  - Creating server instances
  - Assembling and deploying your application

**ORACLE**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

CIM is a command-line utility. From the command line, change directories to `atgdir/home/bin` and run the `cim.bat` or `cim.sh` utility.

# Product Selection



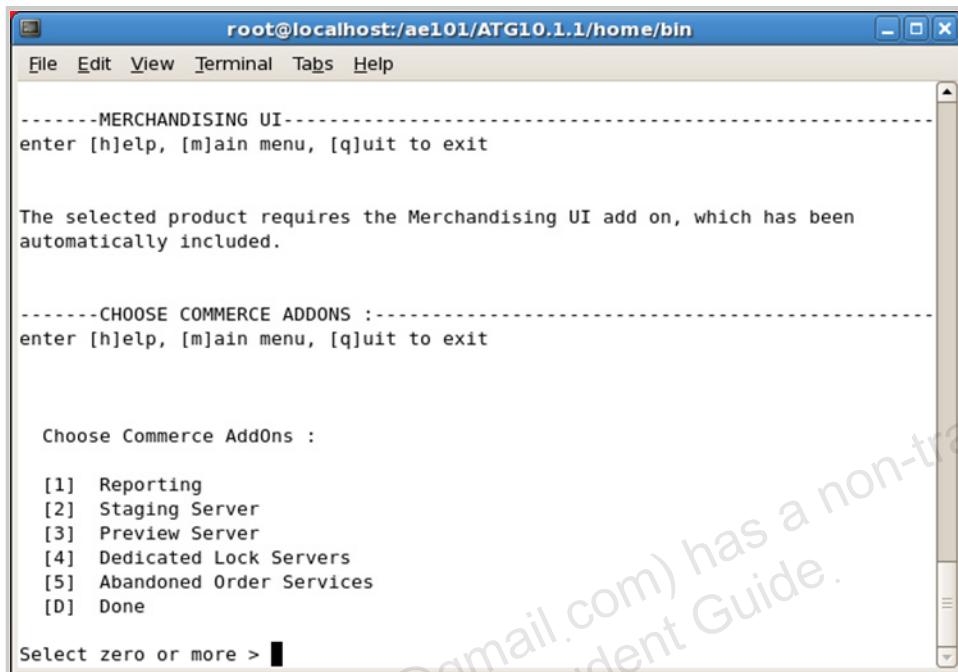
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

If you have CRS installed, it will appear as an option in the product selection menu. Each option lists the various products that will be configured. You type in the number or numbers (space separated) you want to select and press Enter. Retyping a number or numbers, and pressing Enter again, will toggle those choices off. For example, the screen shot in the slide shows five different product selection options. An asterisk appears next to option five (Oracle ATG Commerce Reference Store). If you type “5” and press Enter, it removes the asterisk and shows no options selected.

When finished, you type “d” for done.

# Commerce Addons



ORACLE

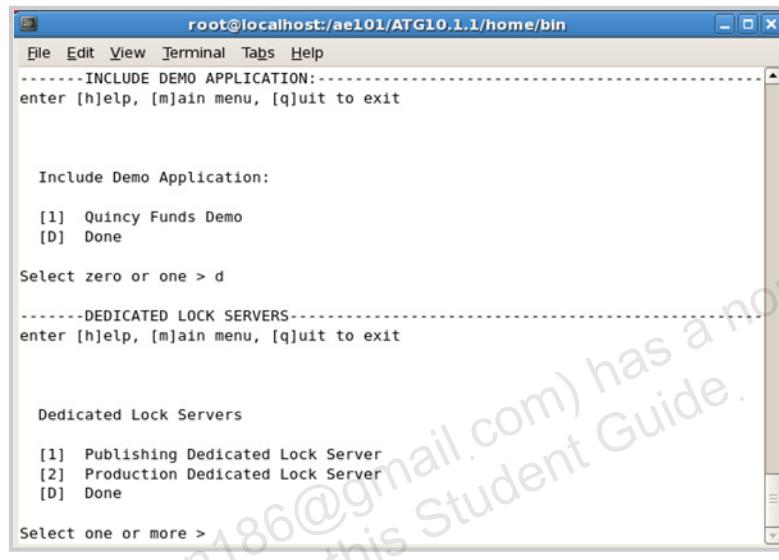
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The next step asks you to choose the commerce addons that you would like to use. You enter the numbers of the options you want individually or with a space between, such as “2 3 4 5” and then press Enter. When finished, you type “d” for done. You have the following options:

- **Reporting:** This option is important if you plan to use Oracle Business Intelligence (OBI). It will configure the module and databases for recording data in a data warehouse.
- **Staging Server:** If you plan to deploy changes to a staging server for testing before deploying to production, choose this option. It is highly recommended that your live environment use a staging server. For development purposes, you can choose not to configure one. Your practice environment does not use a staging server, for example.
- **Preview Server:** You can configure a preview server for your merchandisers to view their changes before deploying them to the staging server. This is an extremely useful tool, and has options to make changes “inline” while in the preview mode. When configuring your preview server, you can either run it on your CA server or configure an external preview server.
- **Dedicated Lock Servers:** For a live site, you will want dedicated lock servers. These help to maintain the integrity of data across multiple servers.
- **Abandoned Order Services:** ATG Commerce provides an optional module for handling abandoned orders.

## Interim Steps

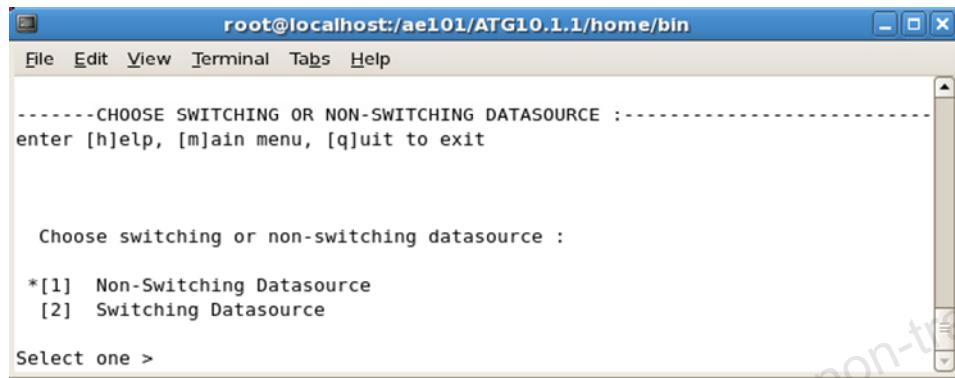
Based on your selected products and addons, you run through a series of configuration steps, such as the following:



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Switching or Non-Switching Data Sources



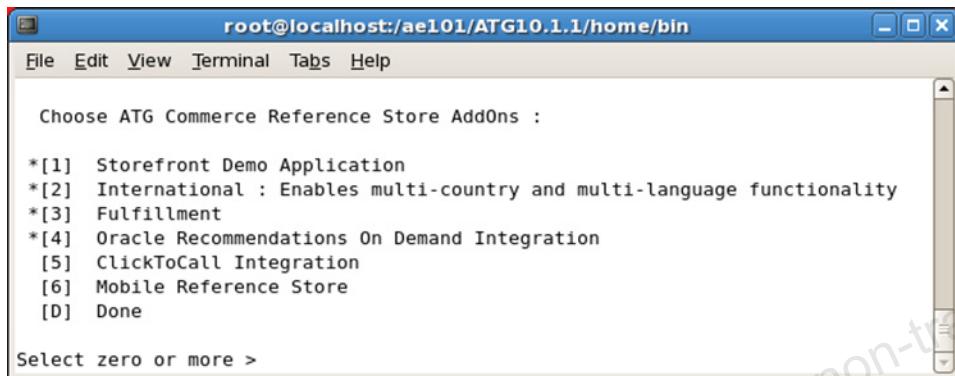
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Will you configure switching or non-switching data sources? For development, you may typically select non-switching data sources. This means that you have one production database. Any changes to the data during a deployment go to that database. A live site typically uses a switching data source. This option configures two data sources: a live data source and a secondary data source. During a deployment, data is deployed to the secondary data source. When the deployment is complete, the secondary, nonlive data source is switched to being live. The formerly live data source becomes the secondary data source and is updated with the new data to make it current with the newly live data source.

Enter the number that corresponds to the type of data source you want to use. The example in the slide shows the first, non-switching option as the default.

## CRS Addons



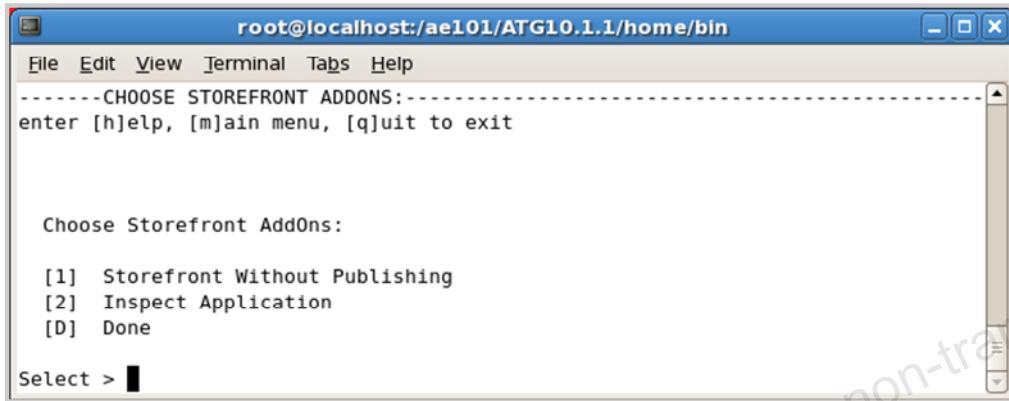
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

CRS has its own optional addons for you to select. In the example in the slide, the first four options are selected. You can easily add or remove options by entering the number at the command prompt. When finished, type “d” for done. The CRS addon options are:

- **Storefront Demo Application:** Populates the database with the ATG Store and ATG Home elements
- **International:** Enables multicountry and multilanguage functionality
- **Fulfillment:** Includes the Commerce Fulfillment module in your application
- **Oracle Recommendations On Demand Integration:** Enables and configures Oracle Recommendations. CRS integrates with sample data on the Oracle Recommendations server. If you have an Oracle Recommendations account, you can configure that for your application.
- **ClickToCall Integration:** Enables the ClickToCall functionality and lets you configure your information
- **Mobile Reference Store:** Loads the mobile version of CRS

## Storefront Addons

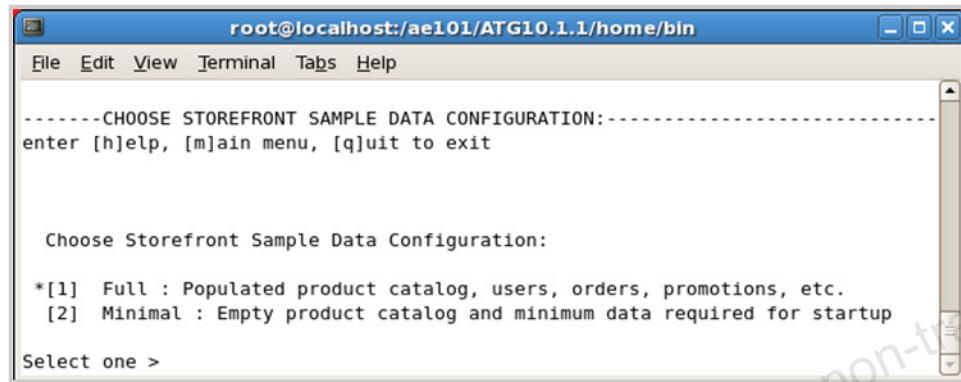


ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The next option lets you select storefront addons. You can select to load the storefront data without publishing. In other words, you will not use ATG Content Administration. You can also select “Inspect Application.” CRS provides a user interface for accessing Fluoroscope, a tool for viewing site HTML pages that reveals key JSP elements involved in rendering those pages, such as page includes, servlet beans, scenario events and actions, and form fields. The Inspect Application addon configures the Fluoroscope functionality.

# Storefront Sample Data



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

You can choose to load both the CRS infrastructure with the sample data or without. Your practice environment contains the full sample data.

## Additional Options

Choose Oracle Recommendations On Demand account:

- [1] Use Recommendations demonstration account
- [2] Enter retailer ID for a specific account

Select one > 1

Choose ClickToCall account:

- [1] Use Click To Call demonstration account
- [2] Enter Webcare account Id.

Select one > 1

Mobile CRS Site Configuration

- [1] UI Only
- [2] Dedicated Site

Select one > 1

Mobile Reference Store Web Services

- [1] REST Web Services for Native Applications
- [D] Done

Select zero or one >

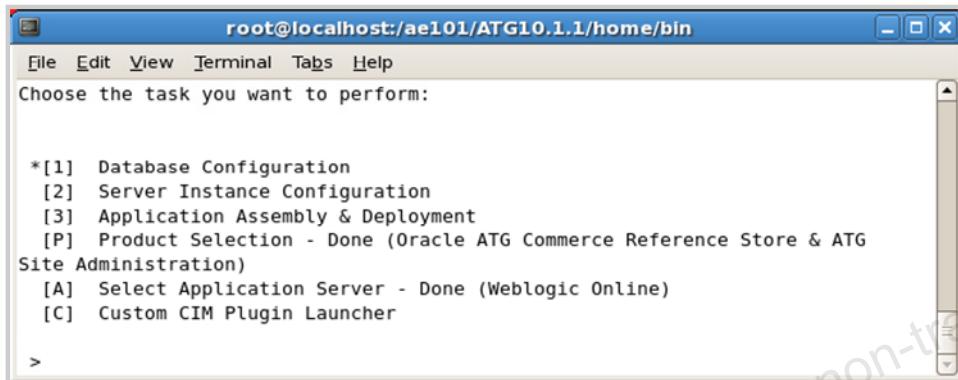
**ORACLE**

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Depending on your CRS addon selections, you will have the opportunity to configure those options. CRS has demonstration accounts for both Oracle Recommendations and ClickToCall. If you would like to see how these are configured in CRS in your development environment, you can choose to use the demonstration accounts. For production, you would choose to enter your organization's own IDs for these options. The mobile CRS application has two additional options, as shown in the slide.

Once you have made all of your selections, you return to the menu to continue to the next task.

## Final Steps



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Once you have completed your selections, CIM walks you through the various tasks. It marks completed tasks as done. In the example in the slide, the product and application server selections have been completed. The next step is the database configuration. When that is complete, you move to server instance configuration, and then application assembly and deployment. You provide the relevant information about databases, JDBC driver location, database names, usernames and passwords, and so on.

## Quiz

CIM is only a useful configuration tool when you are using CRS as the basis for your application.

- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Answer: b

CIM is a helpful tool to configure your ATG Commerce site, regardless of whether you will be using CRS for its infrastructure. For example, it creates database tables for the various types of servers, and configures components based on your product selections and configuration decisions. CIM also assembles your applications for you, which you will be using as part of the practices in this course.

# Road Map

- Commerce-related modules
- Using CIM to configure CRS
- **Using CRS for new implementations**
- Premise for the course practices



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Using CRS for a New Implementation: Steps 1-3

1. Copy the Store module to a new top-level module
  - Example: Copy  
C:\ATG10.1.1\CommerceReferenceStore\Store to  
C:\ATG10.1.1\MyStore
2. Change the name of the web application under Storefront, if desired.
3. Find and replace all references to `Store` in the manifest files with the new module name, similar to the following example:

```
Manifest-Version: 1.0
ATG-Class-Path: lib/
ATG-Config-Path: config/
ATG-Required: MyStore.EStore
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

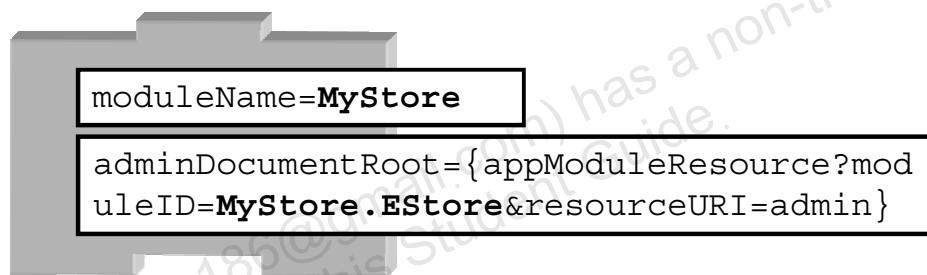
There are many modules that have a reference to `Store` in their manifest files. Depending on your product installation list, you may need to look at the manifest files in the following modules:

- Storefront
- EStore.Versioned
- For an internationalized site: EStore.International and EStore.International.Versioned
- Fulfillment
- For a site implementing Commerce Service Center: DCS-CSR
- For a site implementing Oracle Recommendations: Recommendations.International
- For Endeca integrations: Endeca, Endeca.International, Endeca.Index, and Endeca.International.Index
- For a mobile application: Mobile, Mobile.International, Mobile.DedicatedSite, Mobile.UIOnly, Mobile.Recommendations, Mobile.Recommendations.UIOnly, Mobile.Recommendations.REST, Mobile.REST.Versioned, MobileCommerce, and MobileCommerceiOS
- Cybersource

**Note:** If you are using ATG Search, you will also need to adjust some of the Search modules.

## Using CRS for a New Implementation: Steps 4-6

4. Create a new submodule for your customizations, such as MyStore.Custom, which depends on MyStore.EStore (and other desired submodules).
5. If you are using CA, create the Versioned module under your custom module that depends on both your custom module and EStore.Versioned.
6. In your custom module, override two properties in the /atg/modules/Store component:



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

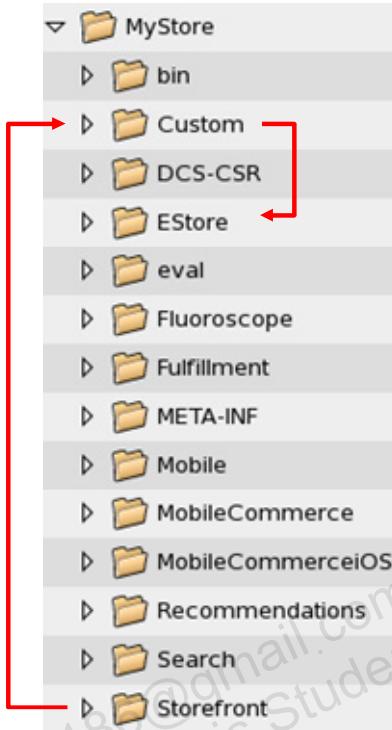
The easiest way to do this is to look at the module dependencies. Anywhere that you find a reference to the Store module or one of its submodules, replace "Store" with your module name, such as "MyStore." See the notes in the next slide for modules that might be affected.

The default CRS settings for the Store component are:

```
$class=atg.service.modules.Module  
moduleName=Store  
adminDocumentRoot={appModuleResource?moduleID=Store.EStore&resourceURI=admin}  
adminHomePageFragmentURL=/store-home.jhtml  
adminHomePagePrefixURL=/atg/store/admin/
```

You need to change the `moduleName` and `adminDocumentRoot` properties to match your store's module. If you change the names of JSPs, you may also need to make adjustments to the `adminHomePageFragmentURL` and `adminHomePagePrefixURL` properties as well.

## Module Dependencies: MyStore.Storefront

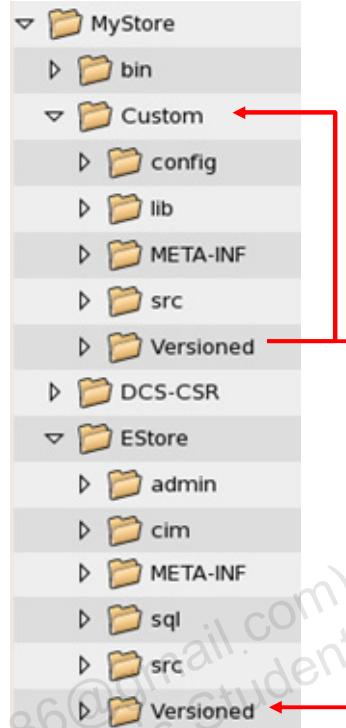


ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

As described in the previous slide, MyStore.Storefront is dependent on MyStore.Custom, which is dependent on MyStore.EStore.

## Module Dependencies: MyStore.Custom.Versioned



ORACLE®

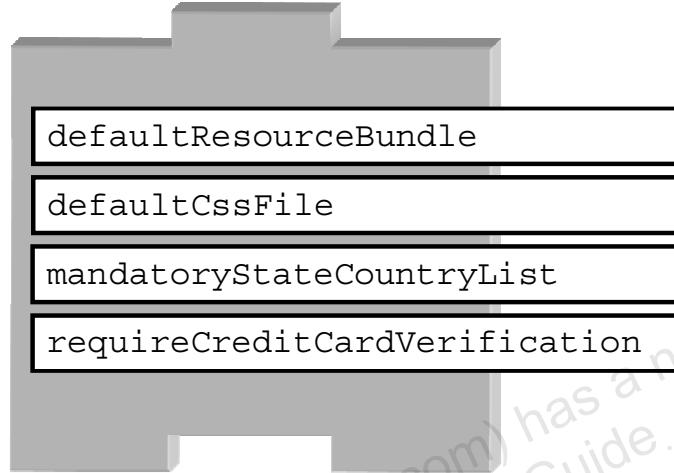
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

As described in an earlier slide, MyStore.Custom.Versioned is dependent on both MyStore.Custom and EStore.Versioned.

**Note:** Some of the folders under the EStore module are cropped out of the slide example to reduce the size of the image.

# Configuring Store-Wide Properties

## StoreConfiguration



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full path to the component is `/atg/store/StoreConfiguration`. This component is used by CRS to configure several store-wide properties, such as the default resource bundle and the default Cascading Style Sheet. It is a CRS-specific component. The properties listed in the slide are:

- `defaultResourceBundle`: The default resource bundle used by CRS. The out-of-the-box configured default is `/atg/projects/store/web/WebAppResources`.
- `defaultCssFile`: The default Cascading Style Sheet (CSS), which is configured out-of-the-box as `/css/site/store`
- `mandatoryStateCountryList`: The list of countries for which state is a required field
- `requiredCreditCardVerification`: A Boolean property that indicates whether orders paid by credit card must include the credit card verification code

For more information on additional properties, refer to the API Reference for the Commerce Reference Store.

## Road Map

- Commerce-related modules
- Using CIM to configure CRS
- Using CRS for new implementations
- Premise for the course practices

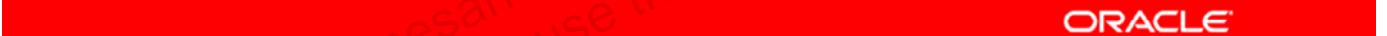


ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Practice Premise

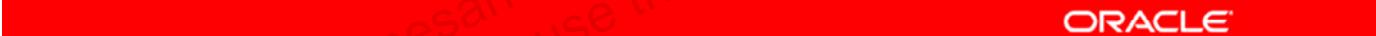
- In this course, you will extend CRS to implement a "points" feature.
  - Shoppers earn points for purchases made with currency.
  - Shoppers can use points to purchase reward items.
- This premise allows you to practice many of the skills needed to extend ATG Commerce.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Taking It Further

- In an actual implementation, you would also consider extending the Commerce-related applications.
- Commerce Service Center would be extended to award points when placing orders and deduct points for returns.
- The business intelligence components would be extended to track points earned and differentiate between reward orders and normal orders.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

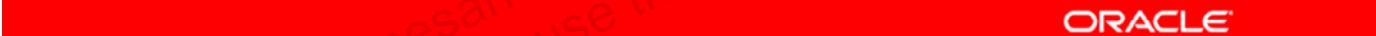
## Summary

In this lesson, you should have learned how to:

- Describe the structure and use of the CRS modules
- Configure CRS by using CIM
- Use CRS as a basis for new Commerce implementations

## For More Information

- Commerce Reference Store Overview
- Commerce Reference Store Installation and Configuration Guide
- API Reference
  - Most of the classes created for CRS extensions start with “Store.”



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Practice 2 Overview: CRS

This practice covers the following topics:

- Becoming familiar with the practice environment
- Starting ATG servers
- Importing the MyStore.Custom module into Eclipse
- Assembling and deploying the application EAR to WebLogic



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2015, Oracle and/or its affiliates.

Ganesan Sree (ganesan186@gmail.com) has a non-transferable  
license to use this Student Guide.

## 3 Personalization

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to:

- Describe user profiles in ATG
- Describe group users
- Describe Commerce-related profile extensions
- Add properties to the user profile

# Road Map

- The user profile
- Ways to group users
- Profile extensions



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## User-Centric Design

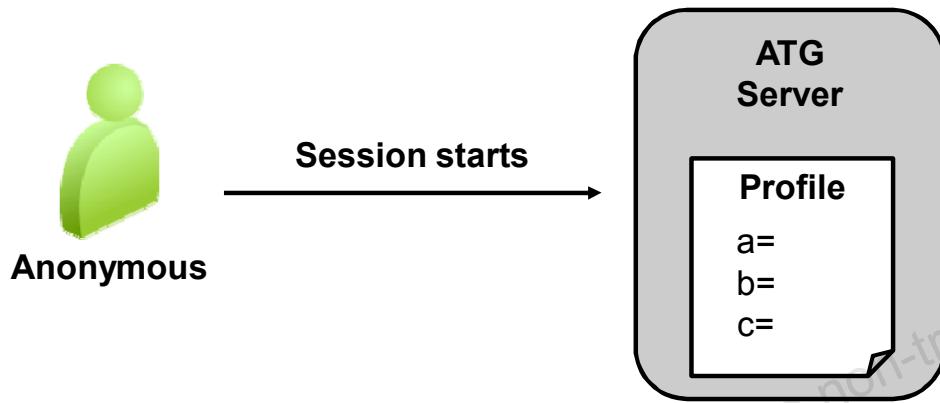
- Like the core ATG platform, Commerce is designed to center on the user.
  - Example: Promotion discounts are linked to users, not orders.
- Much more functionality is available for registered users versus anonymous users.
  - Example: Abandoned order management
- In a multisite application, profiles are always shared across sites.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Profile Object: Session Starts

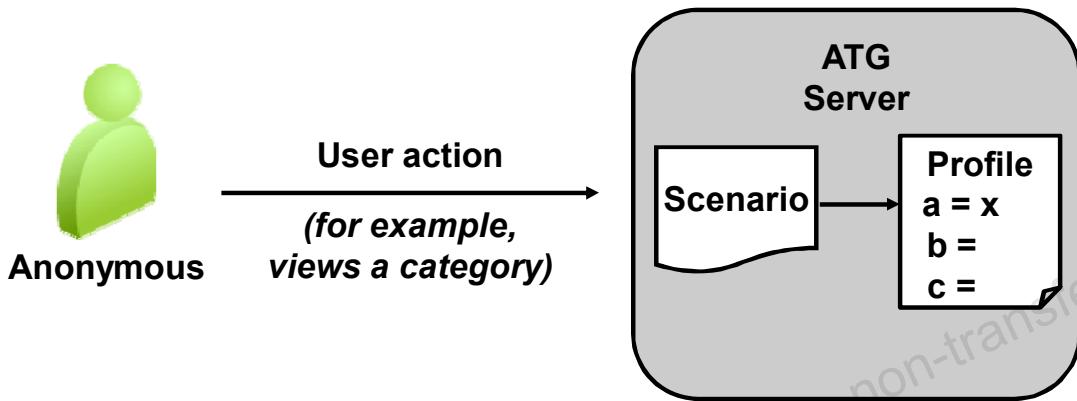


ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

When a user starts a session on an ATG server, a new profile object is created. The profile object starts off empty.

## Profile Object: Implicit Profiling

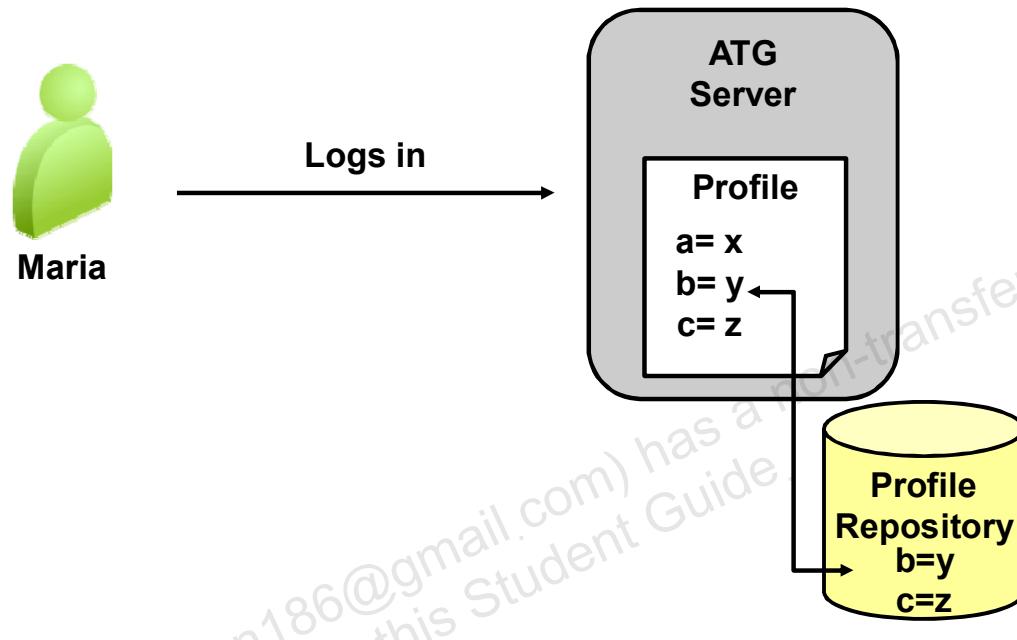


ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Scenarios can be used to react to a user's behavior by recording information about the user's actions in the profile. This implicit data gathering can happen at any point in the user's session, regardless of whether the user has logged in or not. You will learn more about scenarios in the lesson titled "Scenarios."

## Profile Object: User Logs In



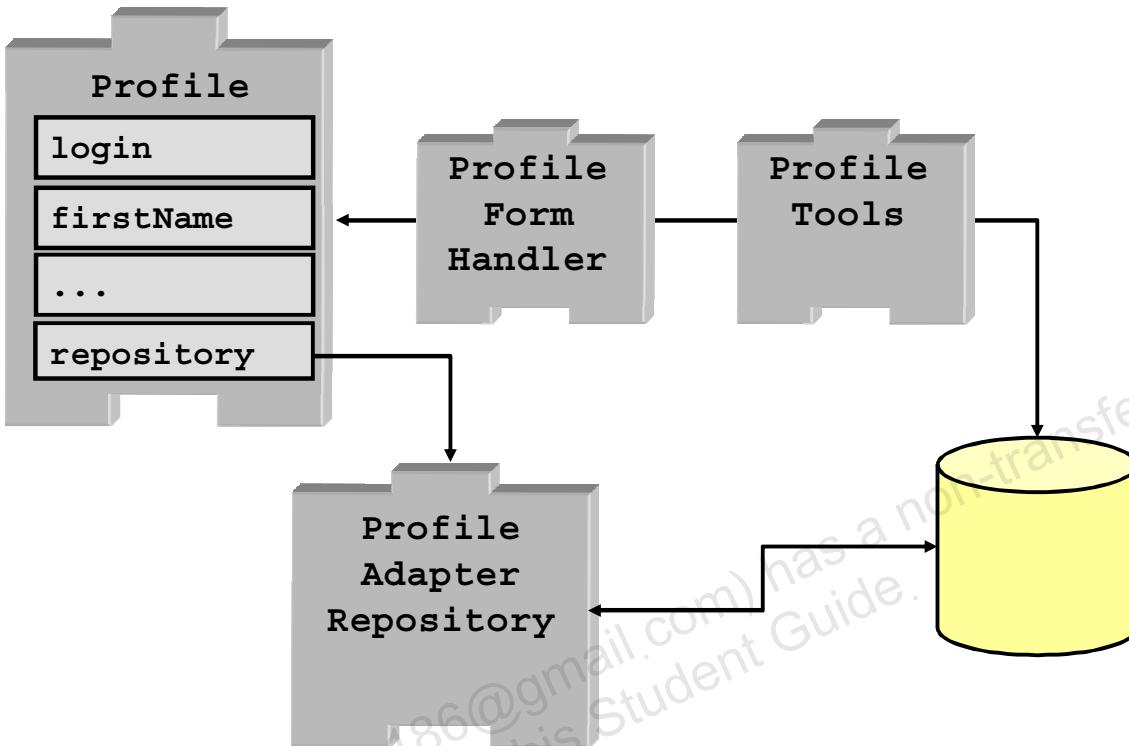
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

If the user logs in, the information stored about that user in the Profile Repository is loaded into the profile object on the server. From this point onward, changes to the profile object will be saved in the Profile Repository at designated times.

The profile object may contain information gathered before the user logged in. Developers can configure on a property-by-property basis whether this information is preserved, discarded, or overwritten by saved information (if any).

## Profile Object Relationships (Personalization Layer Only)



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Once a profile is moved from an anonymous user to a registered user, the profile data in memory is stored in the database and the properties from the database are pulled into the profile on the server (as discussed in the previous slide). The diagram in this slide illuminates some of the objects working in the background. The **Profile** component has a `repository` property that identifies the correct repository (by default, `ProfileAdapterRepository`). **ProfileFormHandler** is the form handler for operating on the current user's profile. It can be used to add new profiles, edit the current profile, log in (switch profiles based on login name and password), and log out. It, in turn, calls **ProfileTools**, which is a general set of tools that help manipulate the profile.

When you add other ATG layers, such as ATG Commerce (the DCS module), additional form handlers exist that can create changes to the profile. Depending on the change, it may prompt the movement of the data in the **Profile** object to the database (such as adding an SKU to an order).

## Design Decision: User Registration

- Will your site largely be used by registered or anonymous users?
- Will you use cookies or quasi-registration to track anonymous users?



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Quiz

A Profile object is created only after a user logs in or registers on the site.

- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Answer: b

The Profile object is created for a user when the ATG session is started. This occurs for both anonymous and registered users.

# Road Map

- The user profile
- Ways to group users
- Profile extensions



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Grouping Users

- There are two ways to group shoppers for personalization:
  - User directory
  - Segments
- Each approach has its particular uses.
- They are not mutually exclusive. A store may use one or both methods.

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## User Directory

- “User Directory” is another name for Profile Repository.
- The user directory groups users by organization, and has three item types:
  - Organization
  - User
  - Role
- Users can inherit information from an organization, such as the default address, catalog, and price list.
- Users can only belong to one organization directly.
  - Organizations can belong to other organizations to form a hierarchy.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The role item type is known as a global role. It also has a subtype called organizationalRole, also known as a relative role. This is a role that is relative to an organization.

## B2C Example: Organizing by Region

The screenshot shows the Oracle Business Control Center (BCC) interface. On the left, there is a navigation tree under the 'Organizations' tab, showing a hierarchy: root > North America > Canada > United States > California > New York. The 'New York' node is selected. On the right, the details for the 'New York' organization are displayed, including its ID (100010). The 'Members' tab is selected, showing two members: 'adrian@example.com' and 'maria@example.com'. Below this, the 'All Members' section shows the same two users, labeled as '2 Members'. At the bottom of the interface, there are 'Add Existing' and 'Add New' buttons.

**ORACLE**

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, the store has chosen to organize its users by region. The regions are organized in a hierarchy by continent, country, and province or state. Users are assigned to the province or state matching their home address. Users could be manually associated with the proper state in the Business Control Center (BCC), or a scenario could be written that assigns users based on their registration information.

The store can now link catalogs and price lists to any level of the hierarchy. For instance, a catalog and price list could be assigned to "United States," and all the users associated with any of the states would automatically inherit that catalog or price list. Alternatively, each state could have its own catalog or its own price list, or both. If the catalog or price list were not set for a particular state, it would inherit the United States' catalog or price list.

## Linking Users to Organizations

Users are typically assigned to organizations either

- Manually in the BCC or
- By scenarios
  - Example: The user provides the organization name or other required information when registering. The scenario reacts to the registration by assigning the user to the appropriate organization.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

# Segments

- User segments:
  - Are based on profile characteristics
    - Example: Users where gender=male and ageInYears>29
  - Become read-only Boolean profile attributes
  - Are automatically maintained by ATG
- Users are in as many segments as they qualify for.
- Users *cannot* inherit information from a segment.

User Segments	
<input type="checkbox"/> Hide segments user is not a member of	
Segment	Membership
Fashionista	false
ThirtySomethings	true
WomenOnly	true
Young	true
MenOnly	false

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Design Decision: Grouping Users

How will your organization group users?

- Will you use segments, the User Directory, or a combination of both?
- If you use the User Directory:
  - How will organizations and suborganizations be structured?
  - How will users be assigned to an organization?



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Road Map

- The user profile
- Ways to group users
- Profile extensions



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Commerce Profile Extensions

- ATG Commerce adds a number of profile extensions related to Commerce.
- Some extensions relate more to a B2C or B2B application, but are available to you to use in either type of store.
- Some Commerce extensions common in both types of stores include:
  - Linking promotion discounts to users
  - Storing credit card details
  - Creating wish lists, gift lists, and purchase lists
  - Setting options for partial shipment and express checkout
  - Linking users to a price list and catalog



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

For full details on the various profile extensions, refer to the chapter titled “Oracle ATG Web Commerce Profile Extensions” in the *Commerce Programming Guide*.

## B2C-Related Commerce Profile Extensions

Some B2C-related Commerce extensions support

- Linking credit cards to users
- Linking addresses to users
  - One default billing address, one default shipping address, multiple secondary addresses



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

For both credit cards and addresses, there are default properties and derived properties. These give customers the ability to either inherit the property from the parent organization or set the properties directly on the profile (potentially overriding the parent organization's settings, if applicable). For derived properties, the value comes first from the corresponding nonderived property and then from their parent organization's profile.

The `allSecondaryAddresses` field can be used to store either alternate billing addresses or shipping addresses, or both.

## B2B-Related Commerce Profile Extensions

- ATG Commerce extensions that support B2B functionality include:
  - Order restrictions and approvals
    - Example: Restricting users' payment methods
  - Preferred vendors
  - Contract objects and linking contracts to organizations
  - Billing terms
- You can also, for personalization purposes:
  - Set an organization's customer type (such as "preferred") and organization type (such as "division")
  - List an organization's contacts

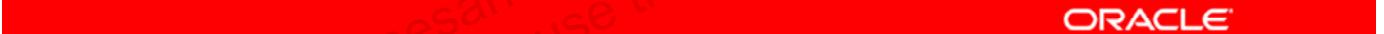


Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

By default, the initial values for the properties that restrict payment methods, restrict orders, require approvals, and set preferred vendors are inherited from the user's organization.

## Extending the Profile Repository

- Stores are expected to extend the Profile Repository to meet their needs.
- Extensions require the following steps:
  1. Creating new database tables or columns
  2. Writing an XML definition of new properties or objects
- Extending the Profile Repository is covered fully in the following courses:
  - *Foundations of ATG Application Development*
  - *Extending the ATG Platform*

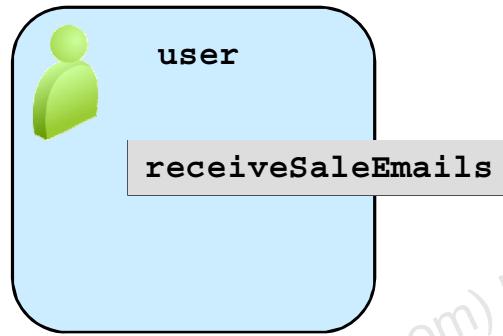
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Adding a Simple Property

Example: The store needs to add a Boolean profile property called receiveSaleEmails.

- This property indicates whether the user wishes to receive weekly sale emails.

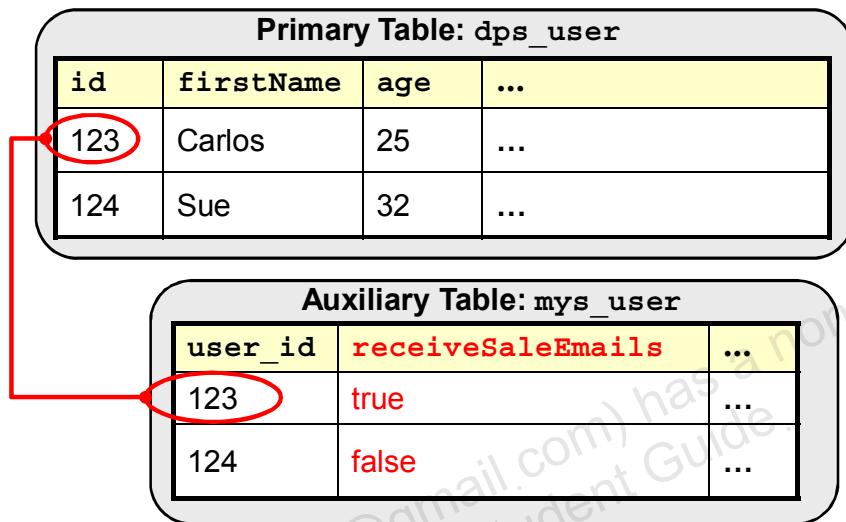


ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Step 1: Add an Auxiliary Table

Best Practice: Add new profile properties to an *auxiliary table*.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Every item type has one *primary table*. Every item has exactly one row in the primary table. Item types may have any number of *auxiliary tables*. Each item in the primary table may have at most one row in an auxiliary table (it may have zero).

Primary tables are joined with auxiliary tables by using the repository ID of the item.

When adding new profile properties, you should use an auxiliary table, rather than just adding them to the out-of-the-box table. If you alter the ATG-defined table, upgrades may eliminate your customizations.

## Step 2: Create a Repository Definition Layer

```
<atgdir>/MyStore/Custom/config/atg/userprofiling/  
userProfile.xml
```

```
<gsa-template>  
  <!-- extend the user item descriptor -->  
  
</gsa-template>
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The process for extending the user profile is similar for Commerce to what you have learned in the prerequisite courses. You create a new `userProfile.xml` file and place it in your module's `config` folder.

## Step 2: Create a Repository Definition Layer

<atgdir>/MyStore/Custom/config/atg/userprofiling/  
userProfile.xml

```
<gsa-template>
  <!-- extend the user item descriptor -->
  <item-descriptor name="user">
    <table name="mys_user" type="auxiliary"
      id-column-name="user_id">
      <property name="receiveSaleEmails"
        data-type="boolean"
        column-name="receiveSaleEmails"/>
    </table>
  </item-descriptor>
</gsa-template>
```

Add the receiveSaleEmails property to the user item type.

Auxiliary Table: mys\_user

user_id	receiveSaleEmails	...
123	true	...
124	false	...

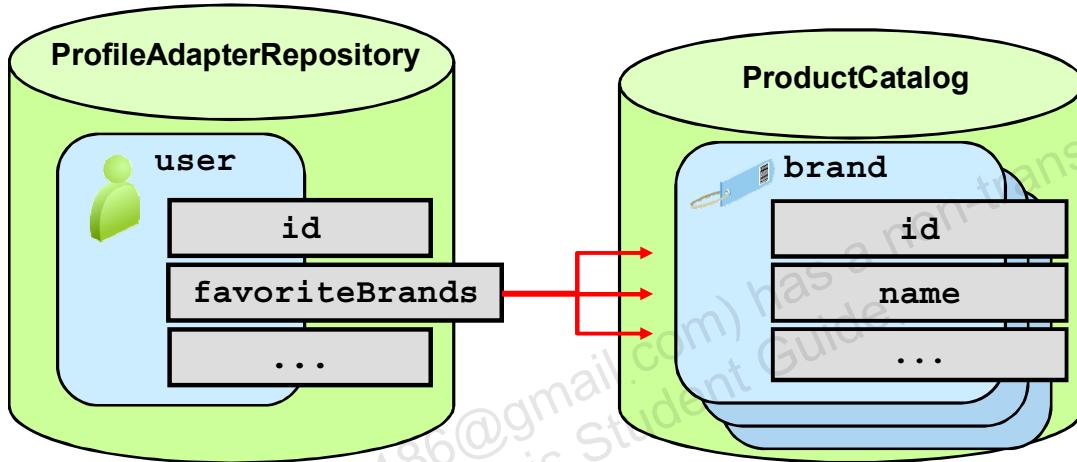
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Adding a Business Object to the Profile: Example

The store needs to add a profile property to point to another business object.

- favoriteBrands: A set of pointers to all brands that each user has identified as a favorite



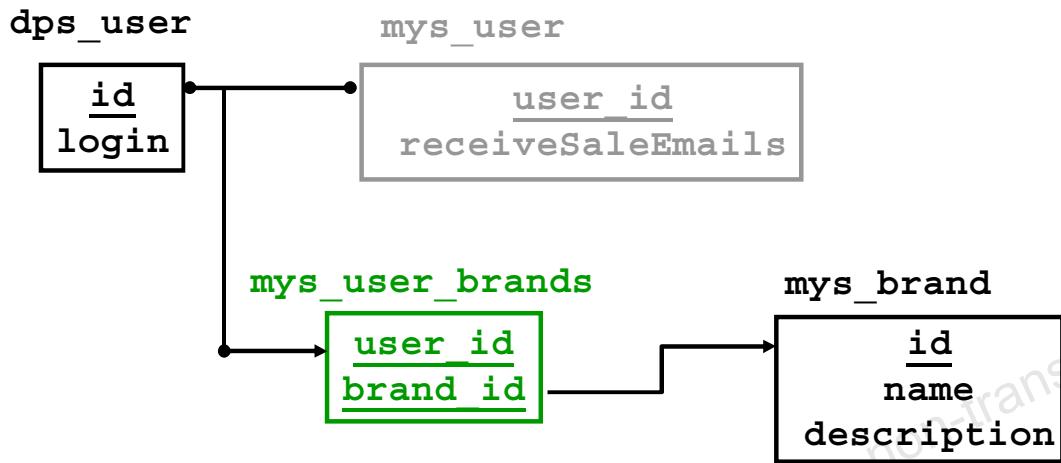
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, a link is being added between the profile and an object in another repository, but it is also possible to add new business objects to the same repository. Also, while this example is a multivalue property, links to other objects may be single-value.

The steps for adding a business object to the profile are similar to those you saw in the slides titled “Step 2: Create a Repository Definition Layer.” The following slides show the two steps.

## Step 1: Add Necessary Tables



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, two tables are involved with the `favoriteBrands` property. One table, `mys_user_brands`, maps the user ID to the brand ID. The second table, `mys_brand`, holds the brand objects.

## Step 2: Add the Property to the User Item

userProfile.xml

```
...





```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

This example is a set of item pointers and is referring to a component-item-type of brand. The brand item type is not defined within the same repository. Because of that, you need to define the `repository` attribute with the full path to the repository component.

## Viewing Combined Layers

ATG's administrative user interface (UI) provides an option to view the combined XML file.

1. Access the administrative (admin) interface:  
<http://localhost:7003/dyn/admin>
2. Use the Component Browser to browse to the Nucleus component  
</atg/userprofiling/ProfileAdapterRepository>
3. Click the Examine Repository Template Definition link.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

A helpful debugging tip is to view the combined XML file. This shows you the final file (built by using all the config layers) that ATG uses to construct the item descriptors for this repository.

This technique is not specific to the Profile Repository; it works for any SQL Repository. Note that the URL provided above for the admin interface may vary depending on which J2EE application server your application is using.

**Note:** The domain and (default) port that you use to access the admin interface may vary at your site, and depends on the server you will be accessing. The example in the slide shows how to access the admin interface for your production server in your training environment.

# Combined Layers in the Component Browser

## Service /atg/userprofiling/ProfileAdapterRepository/

Class [atg.adapter.gsa.GSARespository](#)

### Property definitionFiles

displayName	definitionFiles
expert	false
hidden	false
propertyType	class [Latg.xml.XMLFile;

#### Short Description

#### Value

atg.xml.XMLFile [1]

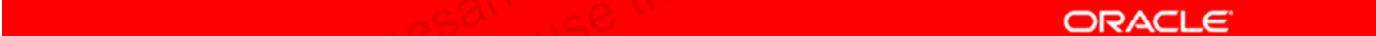
index	CONFIGPATH /atg/userprofiling/userProfile.xml filename Source files	<ul style="list-style-type: none"> <li>• /ae101/ATG10.1/DPS/config/profile.jar/atg/userprofiling/userProfile.xml</li> <li>• /ae101/ATG10.1/DSS/config/config.jar/atg/userprofiling/userProfile.xml</li> <li>• /ae101/ATG10.1/DCS/config/config.jar/atg/userprofiling/userProfile.xml</li> <li>• /ae101/ATG10.1/DCS/Versioned/config/config.jar/atg/userprofiling/userProfile.xml</li> <li>• /ae101/ATG10.1/DCS/AbandonedOrder Services/config/config.jar/atg/userprofiling/userProfile.xml</li> <li>• /ae101/ATG10.1/CommerceReference Store/Store/EStore/config/config.jar/atg/userprofiling/userProfile.xml</li> <li>• /ae101/ATG10.1/CommerceReference Store/Store/EStore/International/config/config.jar/atg/userprofiling/userProfile.xml</li> <li>• /ae101/ATG10.1/home/localconfig/atg/userprofiling/userProfile.xml</li> <li>• /ae101/ATG10.1/home/servers/bizpub/localconfig/atg/userprofiling/userProfile.xml</li> </ul>
System Id (DTD name)	http://www.atg.com/dtds/gsa/gsa_1.0.dtd	

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The example in the slide is a screenshot from the Component Browser. It shows all of the layers that are used to build the combined Profile Repository definition. It shows the default profile from the DPS module, with the layers added by various other modules, including CRS and customizations for the training environment. When you scroll down in the Component Browser, you can see the full, combined definition.

## Design Decision: Profile Extensions

- What properties or objects will your store need to add to
  - Differentiate users?
  - Match users and merchandise?
  - Track user behavior?
  - Limit who is eligible for certain discounts?
- How will those properties be populated?
  - By the customer?
  - By a scenario?
  - From a third-party system?



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

# Quiz

To extend the profile with a simple property, alter the `dps_user` table to add the new column and add the property to `userProfile.xml`.

- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Answer: b

To extend the profile, you should add a new table to the database with an ID column and a column for the new property. You then add the property to `userProfile.xml`.

## **Summary**

In this lesson, you should have learned how to:

- Describe and extend the user profile
- Group users by using the User Directory and segments

## For More Information

### *ATG Personalization Programming Guide*

- Setting Up a Profile Repository
- Extending the Standard Repository Definition



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Practice 3 Overview: Extending Personalization for Commerce

This practice covers extending the user profile to add:

- An integer property for loyalty points
- A “My Home Store” property to link to a physical store location



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# 4

## Scenarios

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to:

- Describe the basics of scenarios and how ATG Commerce extends scenarios
- Create a custom scenario action

# Road Map

- Scenarios
- Custom scenario actions
- Sample action class
- Scenario Manager configuration



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Scenarios

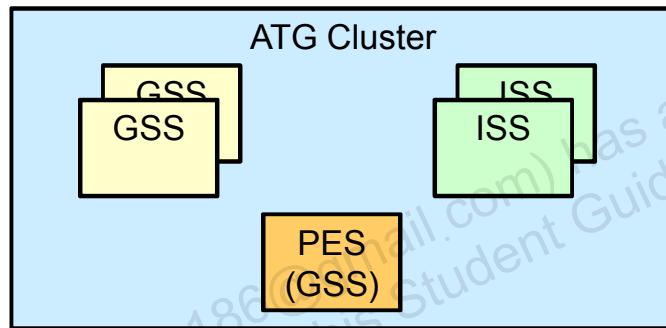
- Scenarios provide a mechanism by which stores can interact with users over time.
- ATG Commerce adds commerce-specific events and actions to available scenario building blocks.
- The Scenario Manager is an event-driven component.
  - When target events occur, specified actions are taken in response.

**ORACLE**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Scenario Server Architecture

- A cluster of ATG servers must always contain the following:
  - Exactly one process editor server (PES)
  - Zero or more global scenario servers (GSS)
  - Zero or more individual scenario servers (ISS)
- The Scenario Manager configures which scenario servers perform which functions.



**ORACLE®**

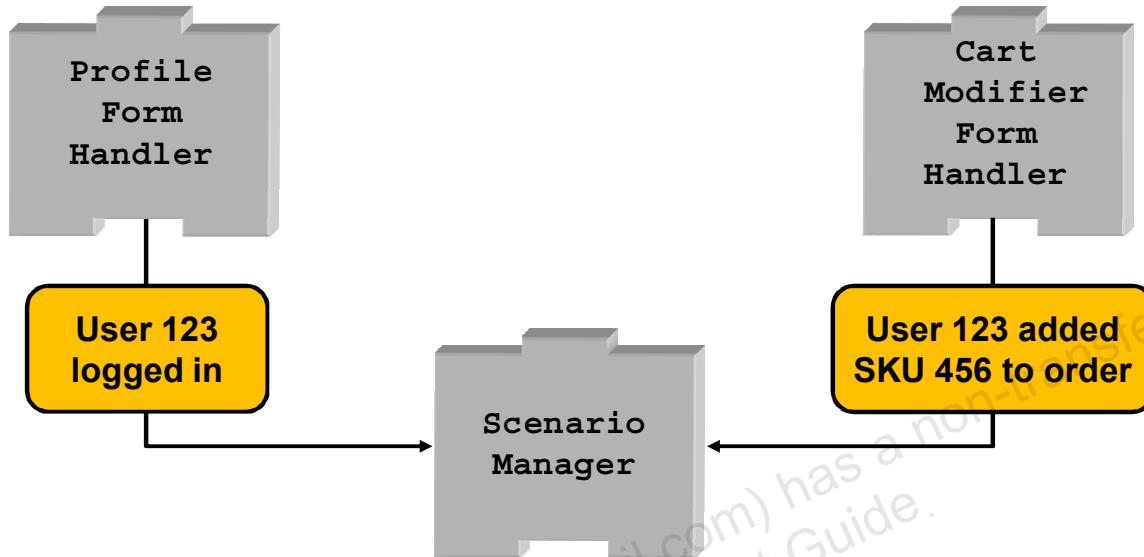
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The individual scenario server handles all the individual, or visitor-specific, events (such as “Visits Page”) for the profiles whose sessions are on that server. For example, suppose you have a simple scenario that records all the page visit events to a data set. When someone visits a page, the corresponding individual server receives the page visit event and performs the scenario action that records the data.

Global scenario servers, in addition to handling the individual events for any of their own sessions, handle global events (such as “Dynamo Starts”) and timer events (such as “Wait 3 hours”). In general, they handle any operations that are not visitor specific. For example, if you have a scenario that sends email to all your registered users, the scenario action to send the email is executed on a global server.

The process editor server is a specific instance of a global server. In addition to handling global events and the individual events for any of its own sessions, it is also responsible for starting and stopping scenarios. When using Content Administration (CA), you create and edit scenarios only in the ACC that is connected to the management server. When you deploy the project containing your scenarios, those changes are deployed to all ATG servers configured to accept the deployment. The Process Editor Server then creates and stores state machines for all enabled scenarios, and stores them in tables in the profile repository. If you were not using CA, you would create and edit scenarios only in the ACC that is connected to the Process Editor Server. The recommendation is that use CA to manage scenarios. You can then test them on a staging server before making them “live.”

## Scenario Events



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

As stated in the slide titled “Scenarios,” the Scenario Manager is an event-driven component. It waits to be told by other store components that certain events have occurred.

Note that this diagram is a simplification of the actual components involved in sending messages to the Scenario Manager. The messages in question do not come directly from ProfileFormHandler or CartModifierFormHandler.

## Commonly Used Commerce Scenario Events

- Item added to order
- Item removed from order
- Promotion offered
- Promotion revoked
- Promotion used
- Gift purchased
- Inventory threshold reached



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The slide lists some of the most commonly used commerce-related scenario events. For the full list, see the *Commerce Guide to Setting Up a Store*.

## Commonly Used Commerce Scenario Events

- Item quantity changed in the order
- Item SKU changed
- Order saved
- Order submitted
- Price changed (triggered when the order subtotal changes)



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Commonly Used General Scenario Events

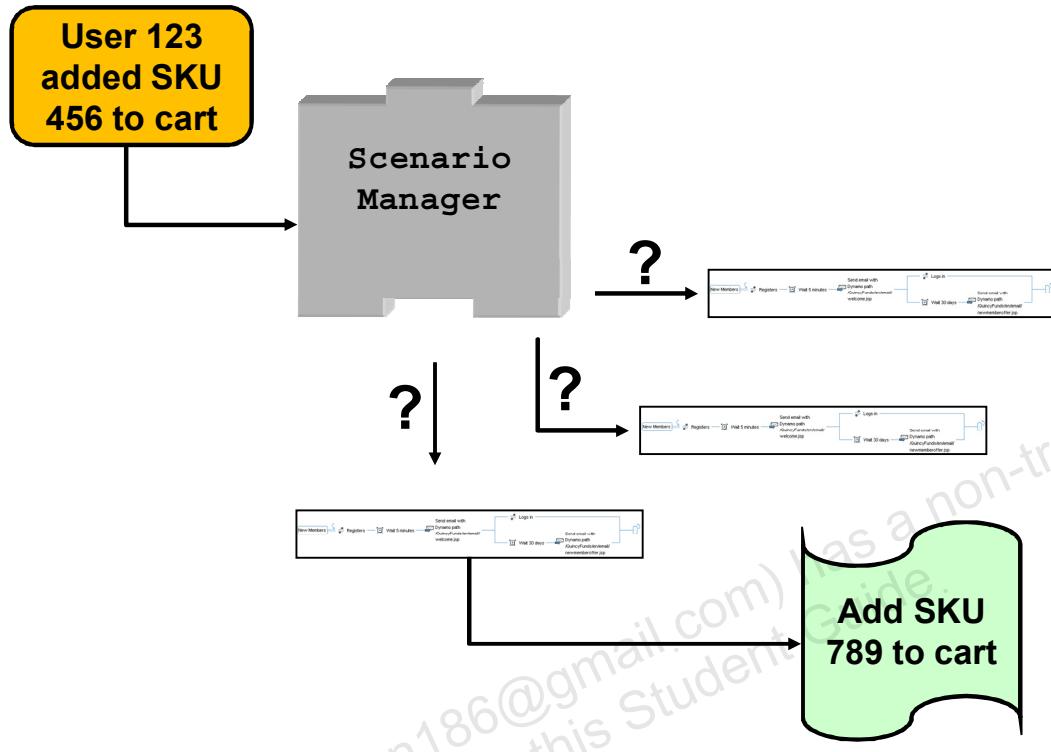
- User registered
- User logged in
- User logged out
- Session started
- Session ended
- Form submitted
- Repository item viewed (such as a category or a product)



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The slide lists some of the most commonly used general scenario events. For the full list, see the *Personalization Programming Guide*.

# Scenario Actions



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

When an event comes in, the Scenario Manager checks to see whether any scenarios are waiting for that event. If there are, those scenarios are triggered to move on to the next defined step, which will often include taking an action, such as sending email or changing values of profile properties. In the example in the slide, a user adds SKU 456 to the cart. The Scenario Manager sees three scenarios and finds one that contains the item added to order event. That scenario then adds a second SKU to the cart, SKU 789.

## Commonly Used Scenario Actions

- Commerce-specific scenario actions:
  - Add item to order
  - Fill related items to slot
  - Give promotion
  - Revoke promotion
- General scenario actions:
  - Send email
  - Change profile
  - Fill slot



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

This slide lists the four commerce-related scenario actions. For more details, see *Commerce Guide to Setting Up a Store*. General scenario actions are listed in *Personalization Programming Guide*.

## Scenario Conditions

- Scenarios can also use conditions to select when to act.
- Commerce-specific conditions include:
  - Item where...
  - Order where...
- General conditions include:
  - People whose...

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Scenario: Example



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, the scenario is triggered when an order is submitted. If the user's email address is either empty or marked “undefined,” nothing happens, and the scenario ends. If the user's email address property has a valid value, a confirmation email is sent.

## Extending Scenarios

- Developers can create custom events, actions, and conditions for use in building scenarios.
- Scenarios are covered more fully in the course titled *ATG Personalization and Scenario Development*.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Quiz

Which of the statements best describe scenarios? (Select all that apply.)

- a. Scenarios allow you to interact with users over time.
- b. The Scenario Manager is an event-driven component that takes action when target events occur.
- c. You may only use the out-of-the-box scenario events, actions, and conditions.
- d. ATG Commerce supplies default commerce-related events, actions, and conditions that you can use to build scenarios.

**ORACLE**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

**Answer:** a, b, d

# Road Map

- Scenarios
- Custom scenario actions
- Sample action class
- Scenario Manager configuration



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Custom Scenario Actions

- Actions perform some kind of task.
- They are commonly used to integrate ATG with other systems.
- A custom action can do anything Java code can do.
- Actions are NOT components.
  - Action instances are created as needed by the Scenario Manager component.



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Individual Versus Collective Scenario Actions

- Scenario actions can be designed to run either
  - Once for each user (individual)
    - Example: If 500 users are going through scenario at same time, send 500 emails.
  - Once for all users (collective)
    - Example: If 500 users are going through scenario at same time, send one email.
- This course focuses on individual actions.



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Creating Custom Actions

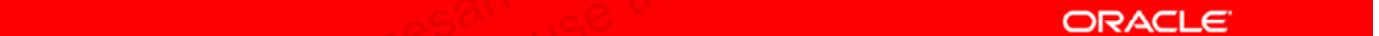
1. Create a class that extends the `atg.process.action.ActionImpl` class.
2. Configure the `ScenarioManager` to recognize the new action in `/atg/scenario/scenarioManager.xml`.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Coding Custom Actions

- In general, action code will:
  1. Extract necessary information from context
  2. Perform some kind of task
    - Examples: Connecting with a third-party system, editing a repository item, altering the order object
- This course focuses on the code needed to access information.
  - The second portion of code is outside the scope of this course.

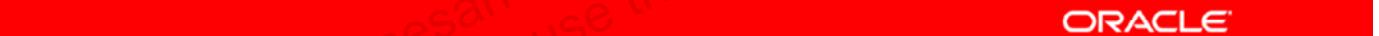


ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Sources of Information for Actions

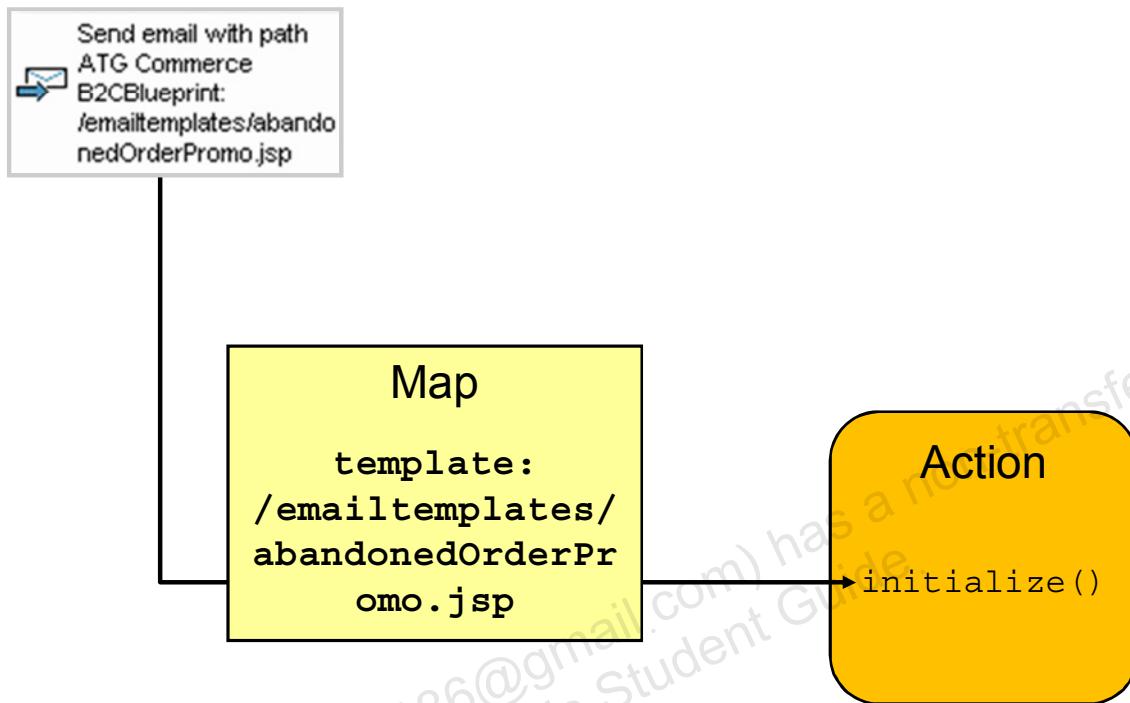
- Execution Context
  - User
  - Event
  - Dynamo HTTP request and response
- Action parameters
  - Set by the scenario writer
  - Used when values might be different each time the action is used
- Associated properties file
  - Set by the administrator
  - Used when values will be same for all uses of the action



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Action Parameters



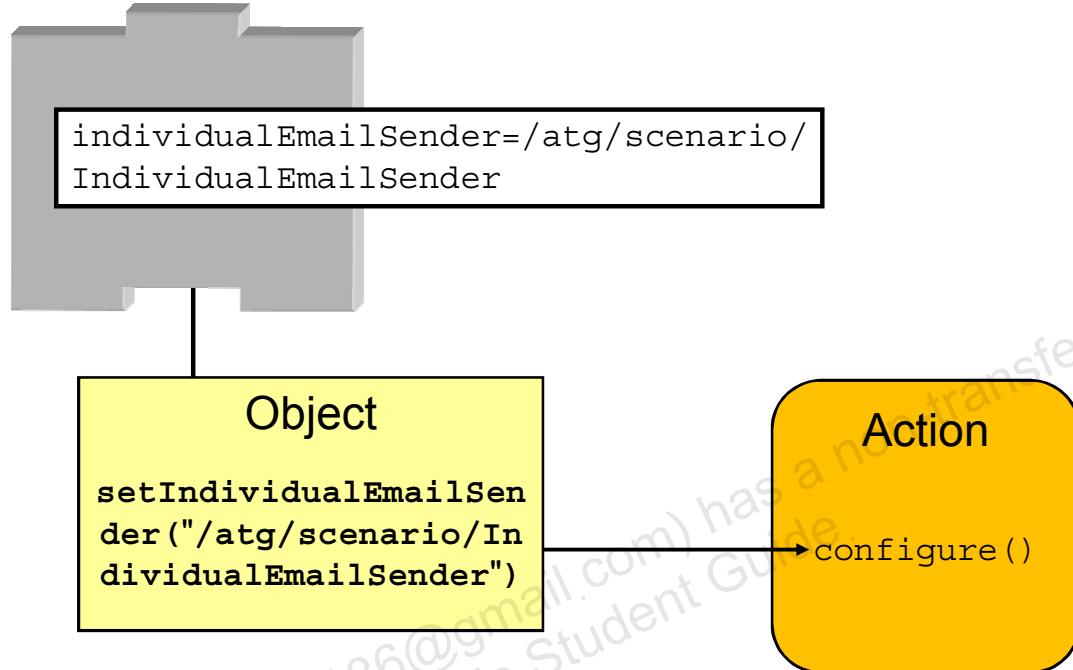
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Action parameters are set by the person writing the scenario. Parameter values will be different for each use of the action. They are passed by the Scenario Manager as a map to the `initialize()` method of the action class. The example in the slide shows a single parameter being passed, but the map may include many parameters. The parameter names are defined in the `scenarioManager.xml` file, which will be discussed in detail later in the lesson. Parameter values can be objects of any class.

## Associated Properties File

SendEmailConfiguration



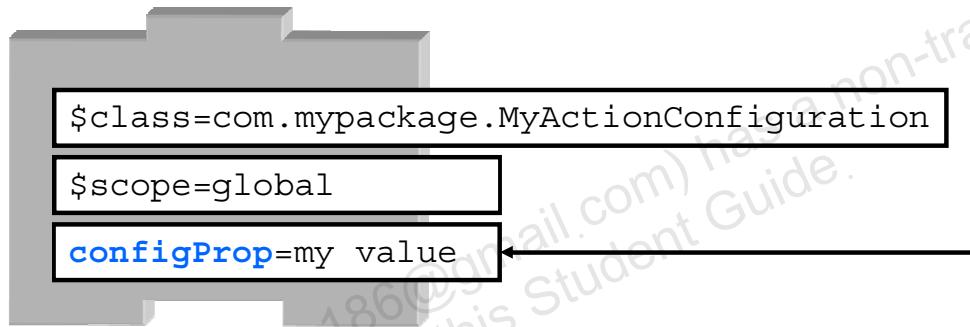
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Actions are not components, and therefore do not have properties files of their own. However, they can be associated with a properties file to store configuration options that will be the same across all uses of the action. These options are passed by the Scenario Manager as a generic Object (component instance) to the `configure()` method of the action. Although the example in the slide only shows one property (`SendEmailConfiguration.individualEmailSender`), many properties can be set and passed by using an associated properties file.

## Sample Configuration Class and Properties File

```
public class MyActionConfiguration  
    extends atg.nucleus.GenericService  
{  
    String configProp;  
    //get method not included in code sample  
  
    public void setConfigProp(String configProp) {  
        this.configProp = configProp;  
    } ...  
}
```



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The configuration class is a very simple bean, used only to get, set, and store properties. Properties can be of any class.

**Tip:** Normally, because an action is not a component, it would not have access to the standard ATG logging capabilities. However, if its associated configuration component extends `GenericService`, the component will include the standard logging methods and properties. The action can then use the configuration component object to log messages.

## Road Map

- Scenarios
- Custom scenario actions
- Sample action class
- Scenario Manager configuration



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Sample Action Class: `getActionName()`

```
public class MyAction
    extends atg.process.action.ActionImpl {

    MyActionConfiguration config;

    getActionName() {
        return "My Most Excellent Action";
    }
    ...
}
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In the practices for this lesson, you create an action class. You will need to follow the example shown in this slide and the next few slides to create your custom action class.

## Sample Action Class: `initialize()` and `configure()`

```
initialize (Map pParameters)
throws ProcessException {

    storeOptionalParameter(pParameters,
                           "optParam",
                           String.class);
    storeRequiredParameter(pParameters,
                           "reqParam",
                           Integer.class);
}

...
configure (Object pConfiguration){

    this.config =
    (MyActionConfiguration) pConfiguration;
}
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

As mentioned in an earlier slide, the action parameters are passed to the `initialize()` method in a map. Use the `storeOptionalParameter()` or `storeRequiredParameter()` methods, as appropriate, to store the parameter values in `ProcessExecutionContext` for later use (discussed in the next slide). The methods both require the parameter map, the name of the parameter (defined in the `scenarioManager.xml` file), and a Class object. The associated component is passed to the `configure()` method as an object. It must be cast to the correct class.

## Sample Action Class: executeAction()

```

executeAction (ProcessExecutionContext pContext) throws
  ProcessException {
    RepositoryItem user;
    if (pContext.isIndividual())
      user = pContext.getSubject();

    Integer reqParam = (Integer)
      getParameterValue("reqParam", pContext);
    ...
    if (config.isDebugEnabled()) {
      config.logDebug("Debugging message");
    }
    ...
} //close executeAction()
} //close MyAction

```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The executeAction() method uses the stored parameter values in ProcessExecutionContext. The ProcessExecutionContext interface represents the context in which a process segment is being executed. The information in the context can be used to evaluate expressions, and to execute actions. The interface contains the following methods:

- getMessage(): Returns the JMS message bean that prompted the process segment execution to start or resume
- getMessageType(): Returns the JMS type of the message that prompted the process segment execution to start or resume
- getProcessInstance(): Returns the process instance going through the process segment
- getRequest(): Returns the current HTTP request, which is null if the process segment is not being executed in the context of an HTTP request
- getResponse(): Returns the current HTTP response, which is null if the process segment is not being executed in the context of an HTTP request.
- getSite(): Returns the site the triggering event uses, if any

- `getSubject()`: Returns the subject going through the process segment, which is null if the segment is not being executed in the context of an individual subject
- `isIndividual()`: Returns true if this context corresponds to an individual subject going through the process segment and returns false if it corresponds to a "collective" process instance that operates on behalf of all subjects

## Road Map

- Scenarios
- Custom scenario actions
- Sample action class
- Scenario Manager configuration



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Configuring the Scenario Manager: Defining the Action

```
<process-manager-configuration>
  <action-registry>

    <action>
      <action-name>myAction</action-name>
      <display-name>My Action</display-name>

        <action-class>
          myapp.MyAction
        </action-class>
        Specifies the Java
        class of the new
        action

        <action-execution-policy>
          individual
        </action-execution-policy>
        Signals that this
        action is an
        individual action

        <action-error-response>
          continue
        </action-error-response>
        Tells the Scenario
        Manager what to do if
        there is an error

      ...
    
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Once you have created your new action class, you need to register your custom action with the Scenario Manager. You do this in the /atg/scenario/scenarioManager.xml file. The example in the slide shows the first part of the configuration:

- <action-name> is the logical name of the action as passed to an action handler.
- <display-name> is the name that appears in the ACC when viewing the list of available actions for a scenario. This tag is optional.
- <action-class> specifies the Java class of the new action.
- <action-execution-policy> has two possible options: collective and individual, as discussed earlier.
- <action-error-response> has three options: continue, retry, and delete. "Continue" moves the scenario ahead as if the action completed successfully, essentially ignoring the error. "Delete" stops the execution of the entire scenario for this user or instance. The scenario is still enabled, and a new instance is created if the first event is triggered again. "Retry" moves the scenario back to its previous state; in other words, it goes back to waiting for the event that immediately preceded the action. When that event occurs again, the Scenario Manager attempts to execute the action again.

# Configuring the Scenario Manager: Configuring Action Parameters

```
...
<action-parameter>
  <action-parameter-name>optParam</action-parameter-
name>

  <action-parameter-class>
    java.lang.String
  </action-parameter-class>

  <required>false</required>
    <description>
      An example of an optional parameter
    </description>
  </action-parameter>
...

```

Determines if the user is required to specify the parameter when using this scenario action



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can define one or more parameters for your action, based on your custom action class. The example in the slide defines an optional String parameter. Parameter types can be any Java class, including arrays. They can also be repository items. To specify that a parameter value should be a repository item, the parameter class should be String or String[], and the <action-parameter-repository-name> and <action-parameter-repository-item-type> tags should be used to specify the repository and item type. See the next slide for an example.

## Configuring the Scenario Manager: Configuring an Action Parameter Value as a Repository Item

```
...
<action-parameter>
  <action-parameter-name>
    stores
  </action-parameter-name>
  <action-parameter-class>
    java.lang.String[]
  </action-parameter-class>
  <action-parameter-repository-name>
    /atg/store/stores/StoreRepository
  </action-parameter-repository-name>
  <action-parameter-repository-item-type>
    store
  </action-parameter-repository-item-type>
</action-parameter>
...
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The example in the slide defines a parameter value that should be a repository item. In that case, the parameter class should be String or String [] (as shown in the example) and the `<action-parameter-repository-name>` and `<action-parameter-repository-item-type>` tags should be used to specify the repository (StoreRepository) and item type (store).

# Configuring the Scenario Manager: Configuring the Action Through a Component

...

```
<action-configuration>
  /myapp/MyActionConfiguration
</action-configuration>
```

```
</action>
</action-registry>
</process-manager-configuration>
```

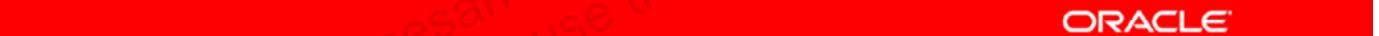
Allows you to configure an action through a Nucleus component's properties file

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Design Decisions: Scenarios

- How might your organization use scenarios?
- Will any custom profile properties be needed to support these scenarios?
- Will any custom events, actions, or conditions be required?



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Quiz

Which one of the statements best describes a custom action?

- a. You must configure your action programmatically.
- b. When configuring the Scenario Manager, you must define <display-name>, but do not have to supply <action-name>.
- c. You can specify a repository as an action parameter value.
- d. A custom action can define only one parameter.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Answer: c

Answer a: You can choose to configure your action through a component.

Answer b: You must supply <action-name>. <display-name> is optional. If you do not define <display-name>, <action-name> appears in the Scenario Editor in the ACC.

Answer d: You can define multiple parameters within your action class.

## Summary

In this lesson, you should have learned how to:

- Describe scenario actions and events, including those added by ATG Commerce
- Create a custom scenario action by writing the action class, configuring `scenarioManager.xml`, and possibly writing a configuration class.

## For More Information

### *ATG Personalization Programming Guide*

- Adding Custom Events, Actions, and Conditions to Scenarios

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

**ORACLE**

## Practice 4 Overview: Extending Scenarios for Commerce

This practice covers the following topics:

- Creating a custom scenario action to award loyalty points
- Registering the new action with the Scenario Manager
- Writing a scenario that gives users 200 points when they register with the site



**ORACLE**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

At this stage, loyalty points will be handled through a simple numeric property. Later in the course, you will replace this with a link to an object.

Unauthorized reproduction or distribution prohibited. Copyright© 2015, Oracle and/or its affiliates.

Ganesan Sree (ganesan186@gmail.com) has a non-transferable  
license to use this Student Guide.

# 5

## Catalogs

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to:

- Describe the structure of ATG Commerce catalogs and the various catalog-related objects, including
  - Catalogs
  - Categories
  - Products
  - SKUs
- Extend the catalog



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Road Map

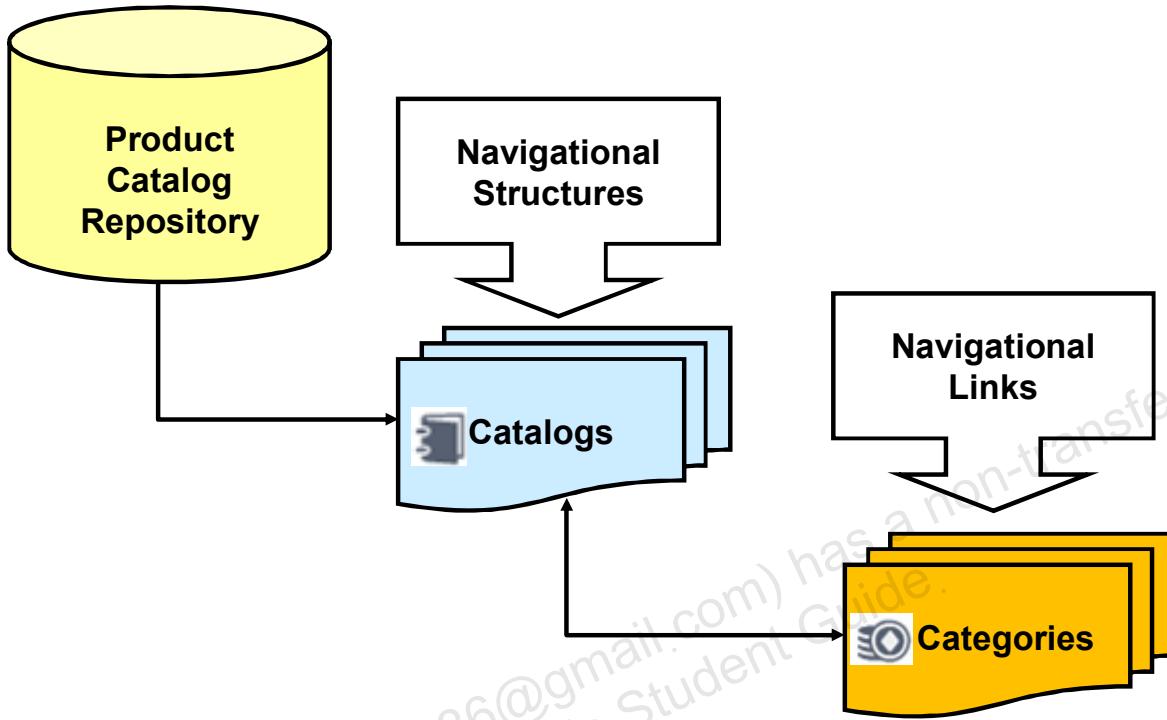
- Catalogs
- Displaying the catalog
- Extending the product catalog



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Product Catalog Repository Item Types: Catalogs and Categories



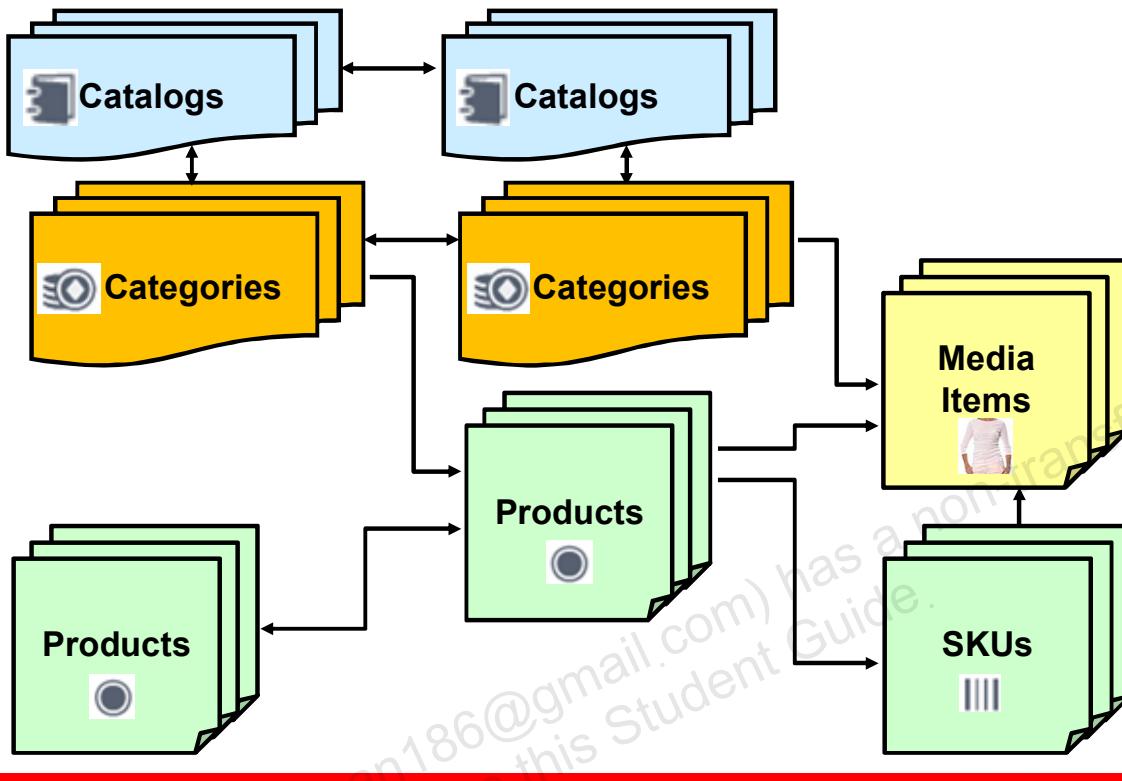
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The Product Catalog Repository stores all of the catalog-related objects. The full path name of the repository is /atg/commerce/catalog/ProductCatalog.

Catalogs can be thought of as a container for a particular navigational structure. Categories can be thought of as navigational links (categories can also be used to differentiate products for personalization and discounts).

## Product Catalog Repository Item Types: Linking



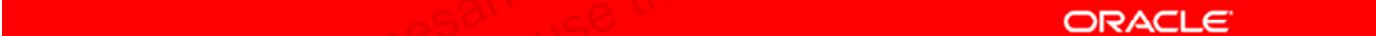
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Catalogs are a container for a navigational structure made up of categories. Catalogs can contain links to subcatalogs and to categories. Categories have properties that link them to other categories and to products. They can also link to other catalogs. Products have properties that link them to other products and to SKUs (stock-keeping units). Categories do not, strictly speaking, “contain” products, nor do products “contain” SKUs. Rather, the objects are peers that are linked together through their various properties. Categories, products, and SKUs have properties that link them to media items.

## Links Between Catalog-Related Item Types

- There are many ways to link item types provided out of the box:
  - Related objects
  - Child objects
  - Parent objects
- These links are used for navigation.
  - Links are not a data structure.

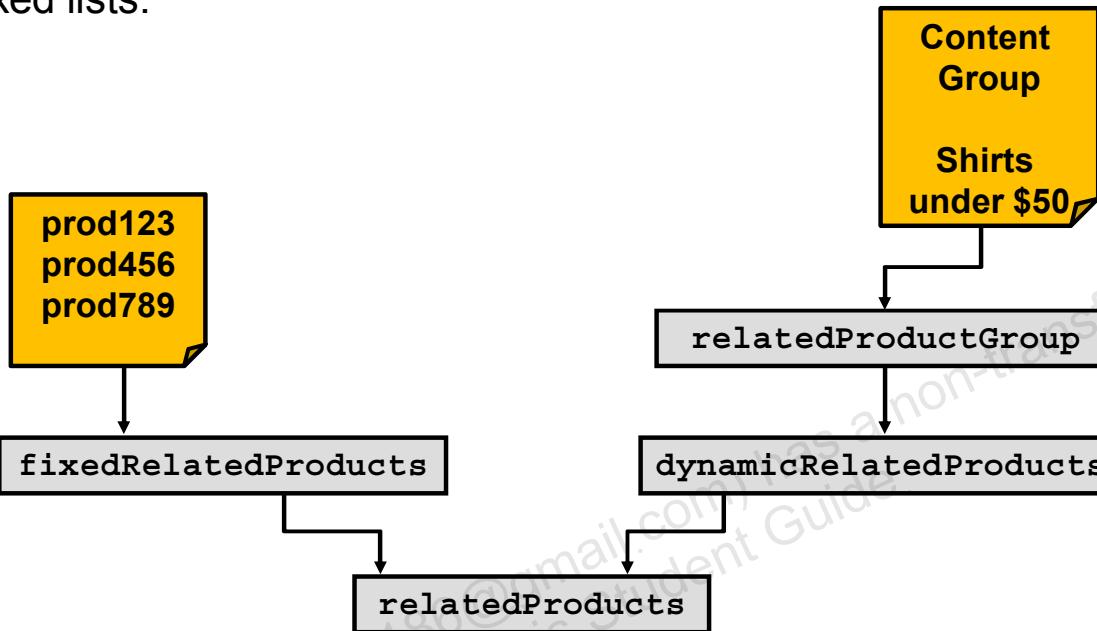


ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Fixed and Dynamic Links

Most lists of linked objects are compiled from dynamic and fixed lists.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Content groups are similar to user segments, in that they are a way to group content items together using a rule. A content group is to content items what a user segment is to people. A content group contains items of one item type, which is selected when you create the content group. Content groups are covered in more detail in the course titled *ATG Personalization and Scenario Development*.

The example in the slide shows the properties used to compile the list of related products. The related products that display on a product page are a combination of the fixed related products and products in the related products content group. The fixed related products are prod123, prod456, and prod789. The content group is defined as shirts that cost less than \$50. All of these would be combined into the `relatedProducts` property that would be used to display the related products in the JSP.

The properties used for other lists follow the same naming format (properties such as `fixedChildCategories`, `childCategoryGroup`, `dynamicChildCategories`, and `childCategories`).

## Catalog Item Types

- Catalog item types:
  - Are a “container” for a navigational structure made up of categories
  - Are invisible to users
  - Can contain subcatalogs
- Subcatalogs can be linked to many parent catalogs.
  - This allows organizations to create a library of reusable navigational “building blocks.”



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

As a reminder, catalog items do not actually “contain” other item types. They have properties that link to categories and subcatalogs.

## Category Item Types

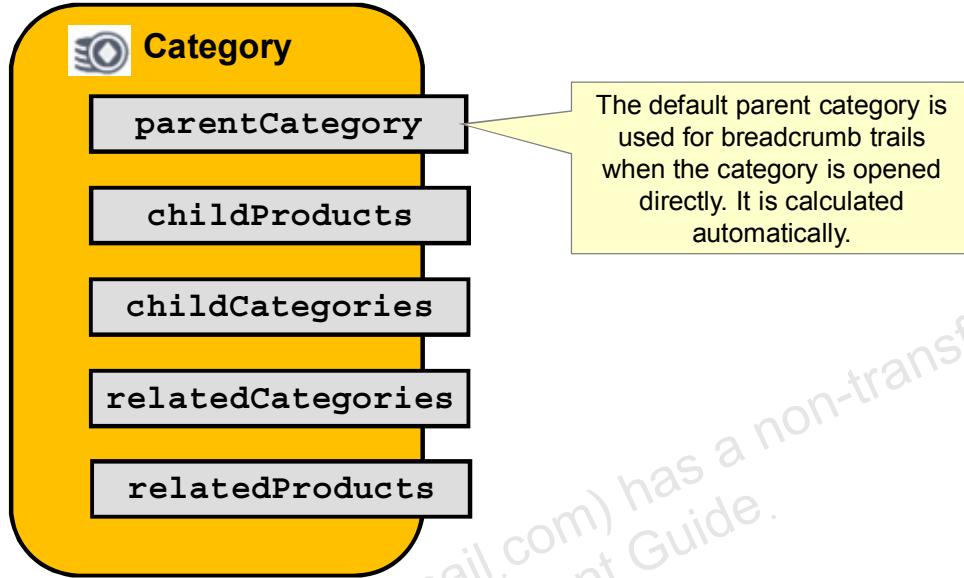
- Category item types:
  - Define navigational links for browsing
  - Are also used to differentiate products for personalization and discounts
    - Example: An unlinked “weekend sale” category used to identify products for discount
  - Can contain links to subcategories
  - Can be linked to more than one catalog



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Selected Category Properties



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

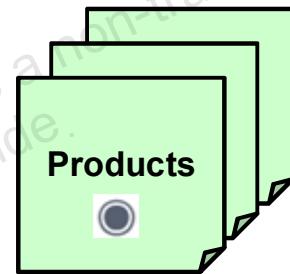
The hierarchy structure is maintained primarily from the top down. A category can “belong” to multiple categories. Similarly, products can belong to multiple parent categories and SKUs to multiple products. Why is there single `parentCategory` property? The `parentCategory` property does not indicate a single parent category, but rather the parent category to use for the breadcrumb trail when there is no browsing history leading to the opening of the category (for example, when the category is opened from a search or a browser bookmark).

Other properties of categories will be covered later in this lesson.

# Products

## Products

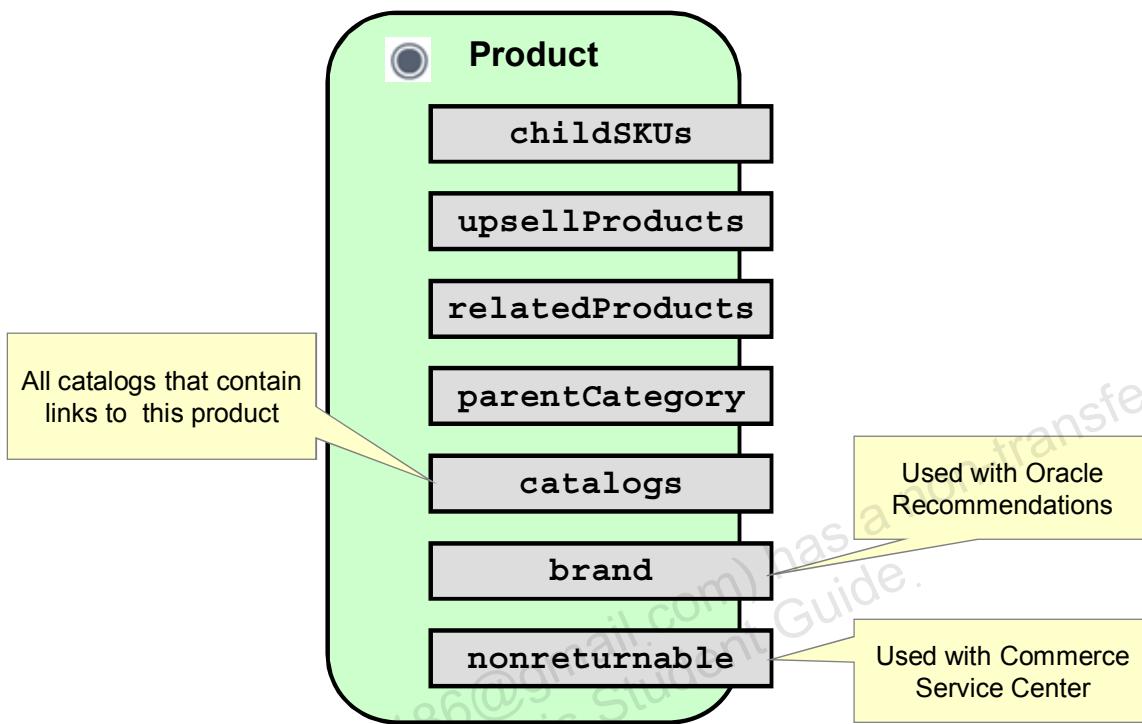
- Contain links for text and images used to describe a group of SKUs
  - Example: “button-down oxford shirt”
- Can be linked to any number of categories in any number of catalogs
  - All links lead back to same product object. The data is not duplicated.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Selected Product Properties



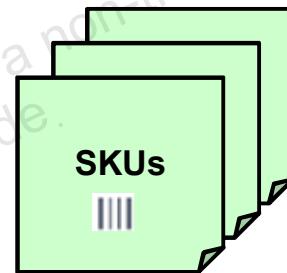
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Similar to categories, the **parentCategory** property of a product is used to construct a breadcrumb trail when the product page is opened directly (for example, through a search or a bookmark).

# SKUs

- SKUs are purchasable items.
  - Examples: “Button-down oxford shirt, size large, color blue”
- Pricing is generally done at SKU level.
- Each SKU is associated with a particular fulfills.
  - Examples: Hard good versus soft good
- A SKU can:
  - Include descriptive elements and images, such as a color swatch image
  - Can be linked to any number of products

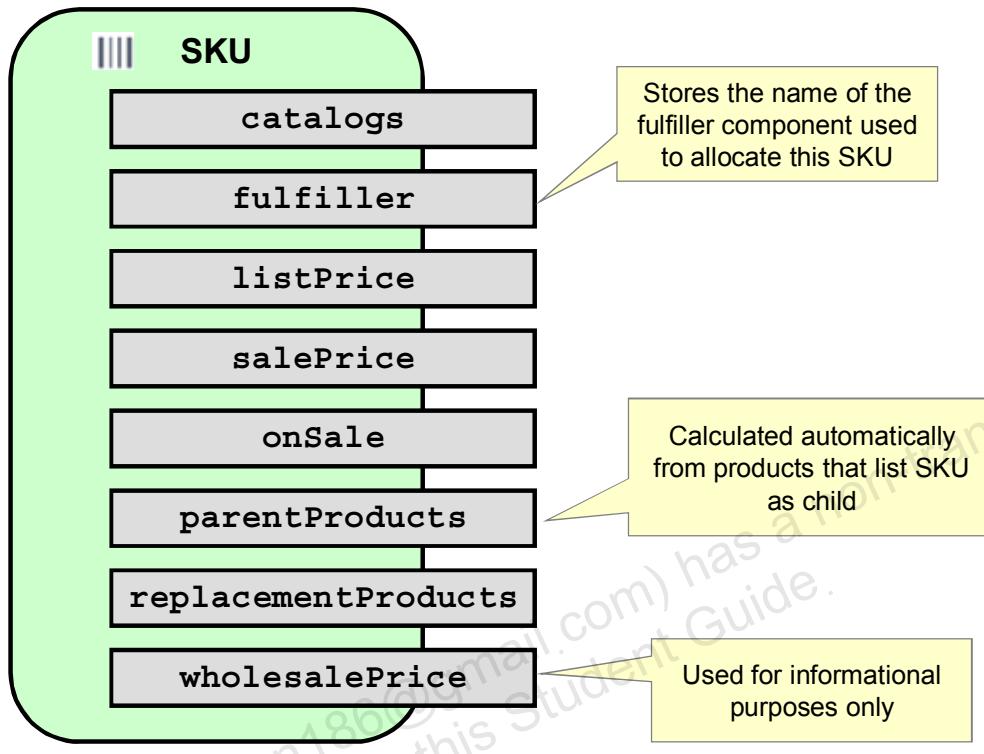


ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A hard good is a physical SKU that would be delivered in the mail, such as a shirt. A soft good is also known as an electronic good. This is an item that would typically be downloaded, such as a music file or an electronic gift certificate. You need to associate each SKU with the fulfiller that handles its type.

## Selected SKU Properties



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Examples of fulfillers that come with ATG Commerce out of the box are the “HardgoodFulfiller” that handles physical goods that are shipped through the mail and the “SoftgoodFulfiller” that handles electronic goods, such as an electronic gift certificate.

## SKU Bundles

- SKU bundles allow you to package several SKUs as one purchasable item.
  - There is one price for the entire bundle.
  - There is one item in the shopper's cart.
  - The bundle is split into member SKUs for fulfillment.
- SKUs can be linked to as many bundles as needed.
- SKU bundles are available out of the box if you are using ATG's default inventory management.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Inventory management will be covered in detail in a later lesson.

## Configurable SKUs

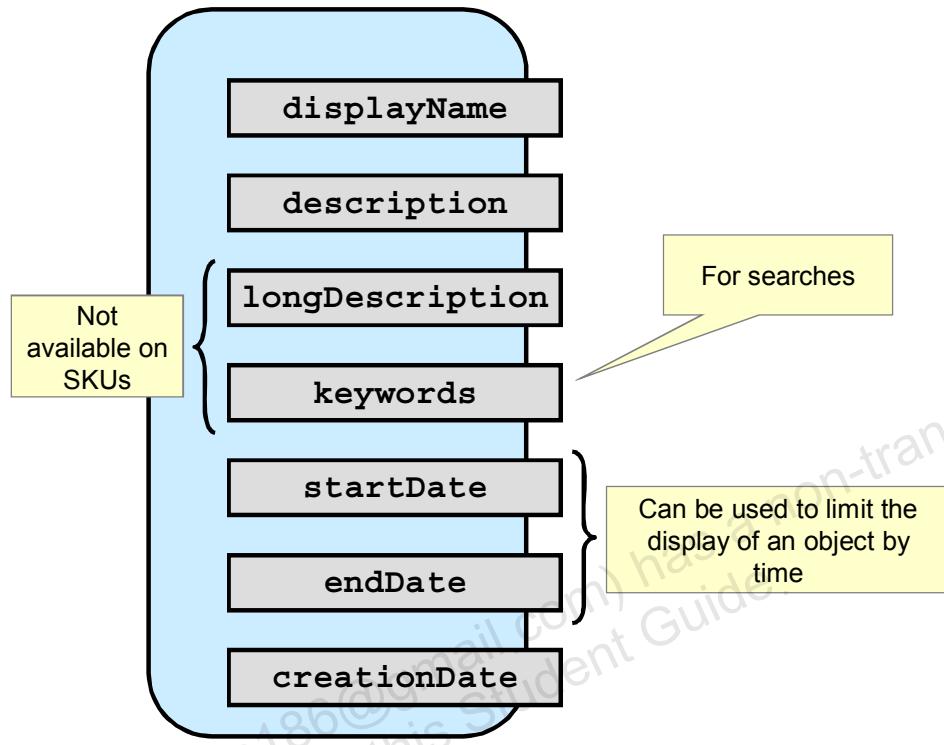
- Configurable SKUs are used when items have options that affect price, such as:
  - The amount of memory and hard-drive size for a computer
  - Personalization text printed on the product
  - The number of minutes in a mobile plan
- After the user selects values for all options, a single item is added to the cart.
  - This item is also considered a single item for fulfillment.
- Configurable SKUs require some custom development to implement.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

While the order management components include all the necessary code for handling configurable SKUs, there are no form handlers available out of the box for adding these SKUs to the order.

# Common Properties Across Catalog-Related Item Types: General Properties

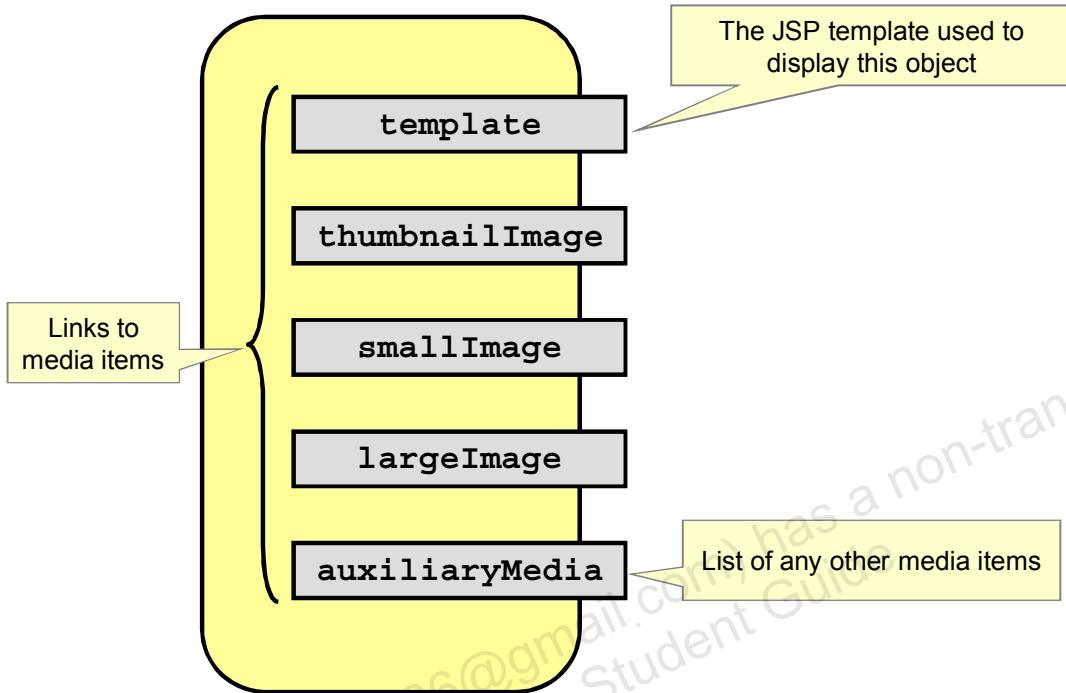


ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Categories, products, and SKUs have several identical properties in common. This slide and the next slide discuss these common properties.

## Common Properties Across Catalog-Related Item Types: Media Items



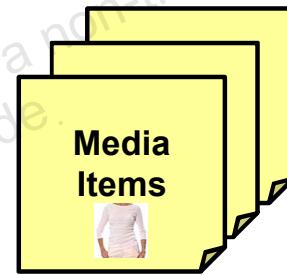
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The categories, products, and SKUs have links to media items. The properties listed in the slide show the common properties of categories, products, and SKUs that link to media items. CRS adds a few additional image properties that it uses, such as the medium image. You will learn more about media items in subsequent slides.

## Media Items

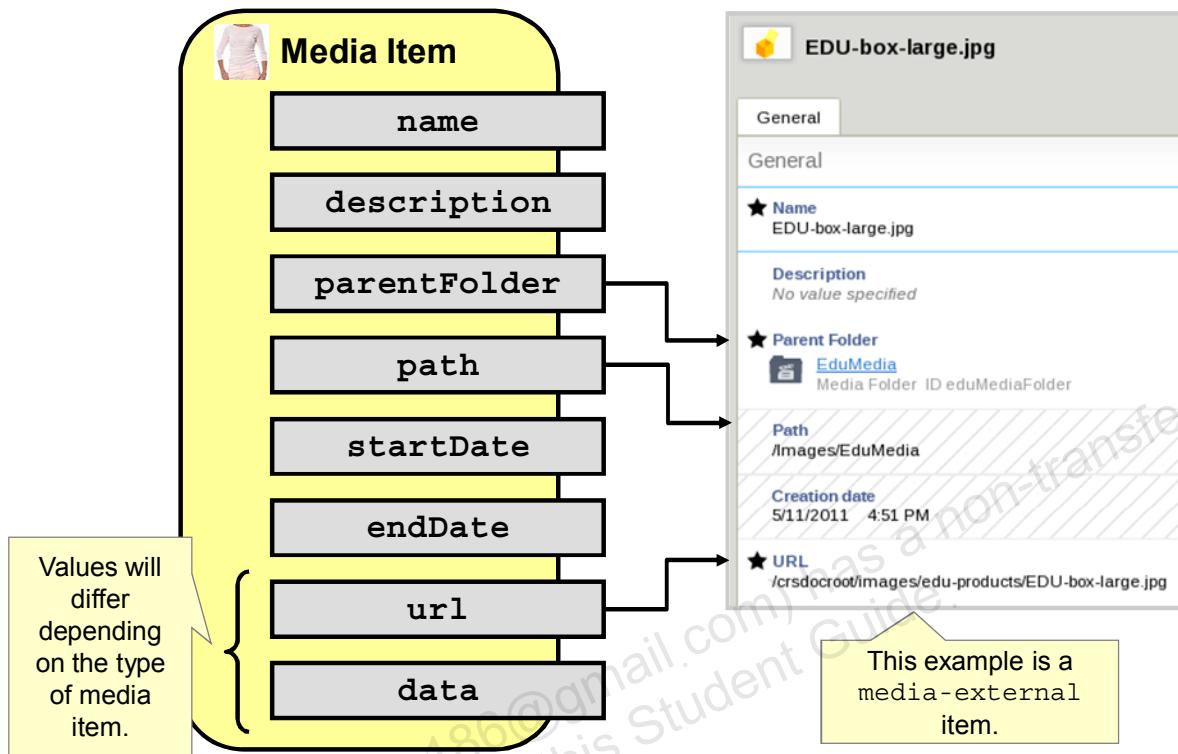
- media is a base item-type in the Catalog Repository.
- Media items:
  - Represent files used to illustrate or display other catalog objects, such as images and JSPs.
  - Are stored separately from other catalog objects to make reuse easier
    - The file is stored once no matter how many times it is used.
- There are three media subtypes:
  - media-external
  - media-internal-binary
  - media-internal-text



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Media Item Properties



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The Product Catalog Repository is a SQL content repository. A content repository is composed of folder and content repository items. The product catalog defines folder and media item types that correspond to these two parts of the repository.

You can use folders to organize media elements. Folders are used only in the administrator user interface (the BCC), not in the commerce site itself (the JSPs).

For more information about SQL content repositories, see “SQL Content Repositories” in the *ATG Repository Guide*.

The example in the slide shows a media-external item type. The `parentFolder` and `path` properties show the organization of the item within the BCC. This item's `parentFolder` is EduMedia and its `path` is `/Images/EduMedia`. This means that the media item is in the EduMedia administrative folder, which is in the Images folder. The `path` property is filled in automatically, based on the folder hierarchy.

For a media-external item type, the `url` property points to the correct location of the actual image file on the target server. This property would be entered by the BCC user. The example in the slide has the following URL: `/crsdocroot/images/edu-products/EDU-box-large.jpg`.

## **media-external**

- The file is stored outside of the database.
- The data property is defined with property-type="atg.repository.FilePropertyDescriptor" and data-type="java.io.File".
  - Its value is an object instantiated by ATG as needed.
- The url property is entered by users.
  - Must point to the correct location of the file on target servers
- CRS examples of media-external items include images and templates.

**ORACLE**

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## **media-internal-binary**

- The contents of a media file are stored in the Catalog Repository database, accessed through the data column.
- The data property is defined as data-type="binary" in ATG.
  - Maps to a binary, varbinary, image, long raw, or blob database column, depending on the database vendor
- The url property is calculated by ATG to provide access to the internal file contents.

**ORACLE**

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## **media-internal-text**

- The media file is stored in the database in a data column.
- The data column is defined as big string in ATG.
  - Maps to long varchar or clob depending on the database vendor
- The url property is calculated by ATG to provide access to internal file.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The data for a media-internal-text item can be entered in a text field in the BCC, when the business user creates a media item of this type.

# Road Map

- Catalogs
- Displaying the catalog
- Extending the product catalog



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Ways to Display the Catalog

You can choose different ways to display your ATG Commerce catalog by using:

- ATG Commerce's navigational structure through JSPs
- Oracle Endeca templates and cartridges
- A combination of both



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Using a combination of both ATG Commerce's navigational structure and Endeca's templates and cartridges requires some customization.

## Displaying the Catalog (ATG)

- Part of creating the catalog structure is also thinking about the structure of your JSPs.
- CRS shows an example of navigating to categories, and then drilling down to subcategory and product pages.
  - Uses ATG-provided lookup droplets and sample drill-down JSPs.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

For more information about how to construct your JSPs to display categories and products, see Appendix A: Displaying the Catalog and Breadcrumbs.

## Breadcrumb Trail (ATG)

- The breadcrumb trail:
  - Helps users traverse category hierarchy by keeping a “stack” of catalog elements
  - Is sometimes called “Historical Navigation”
  - Is needed because there may be more than one path to a product or subcategory
- ATG provides components to track the historical navigation as a customer browses the store.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE®

For more information about the navigation components and how these work with your JSPs, see Appendix A: Displaying the Catalog and Breadcrumbs.

## Oracle Endeca Experience Manager Components

- Experience Manager uses a different approach to construct pages, by using templates and cartridges.
- Templates:
  - Are prebuilt page layouts
  - Determine where content and data are placed



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

# Cartridges

- Cartridges are prebuilt, modular components that pull in content and data from the Endeca engine and external systems, such as ATG Commerce.
- Cartridges expose Endeca features such as guided navigation, search, and spotlighting.

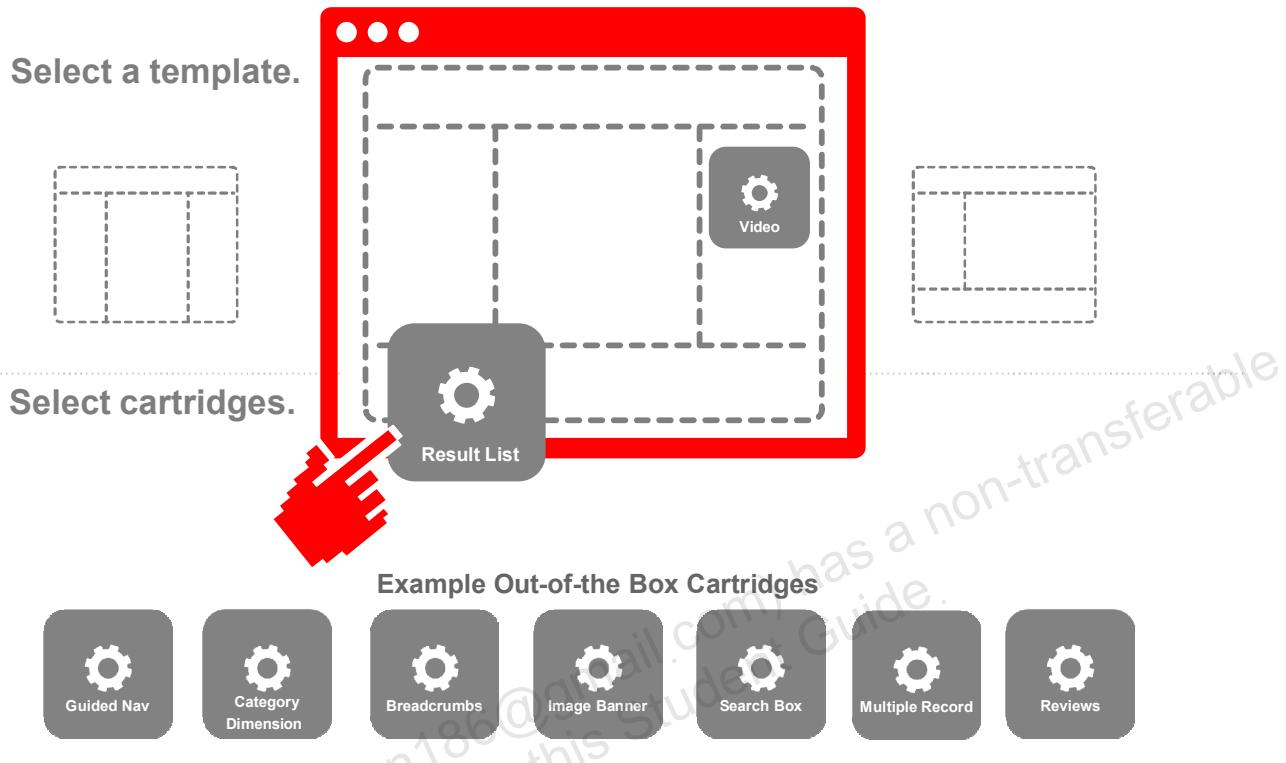


ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

CRS 10.1.2 contains several cartridges that demonstrate how to integrate ATG Commerce with Endeca.

# Templates and Cartridges



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

From within Experience Manager (the business user UI), you can configure pages by using templates and cartridges. To learn more about how to set up Endeca from the technical side, see the course titled “Developing Endeca Commerce Experience Manager Apps.”

## Design Decision: Catalog Structure

- How will your organization map the items it sells to ATG's catalog objects?
- Will binary files be stored as media-internal-binary or media-external?
- How will you set up your store pages?
  - Will you use ATG's catalog navigation?
  - Will you use Endeca to create cartridges?

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

If you are using ATG Commerce Search, will you navigate your catalog by using categories, facets, or both?

## Road Map

- Catalogs
- Displaying the catalog
- Extending the product catalog



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Extending the Catalog

- Stores typically need to extend the Product Catalog Repository.
- There are multiple ways to extend the catalog:
  - Add properties to an existing item type
  - Remove or modify properties of an existing item type
  - Create a subtype of an existing item type
  - Create a new item type

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The Product Catalog Repository is an ATG repository like others you learned about in the course titled *Foundations of ATG Application Development*. You extend it like you would any other ATG repository by adding an XML file in your module layer that adds, modifies, or subtracts properties from other module layers. Examples of catalog extensions include things such as a product's weight, a SKU subtype for shoes that holds the width, or a manufacturer object.

## Add Properties to an Existing Item Type

- You add properties to an existing item type when custom properties apply to every instance of that item type.
- CRS examples of custom properties added to the product definition include:



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The examples in the slide show a new “Enable Email A Friend feature” property and a “New” property. Additional examples of CRS properties link to another item type. You will see an example of that in a subsequent slide.

## Create a Subtype of an Existing Item Type

- This is used when custom properties apply to only some instances of that item type.
- The subtype automatically inherits all properties of the supertype.



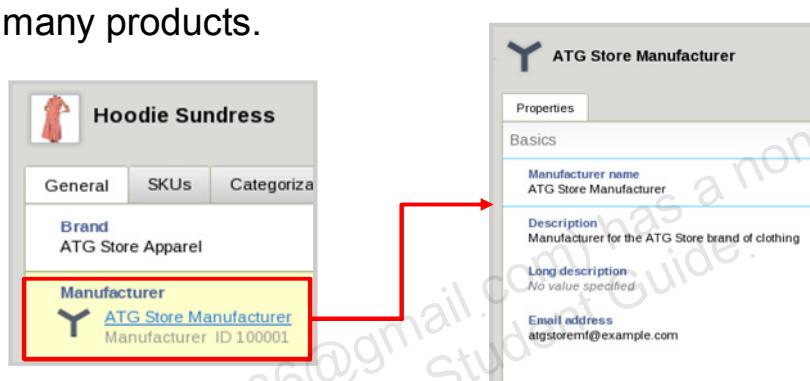
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

CRS adds several SKU subtypes, including the furniture SKU subtype shown in the slide. When you create a new furniture SKU, you have two custom properties that are added for that type of SKU: the default wood finish and a color swatch.

## Create a New Item Type

- A new item type is used to store a group of properties under a single, reusable object.
- Best practice: It is better to make an item type and link it to other item types than to duplicate values.
  - Example: Link many products to one manufacturer item type rather than storing the same manufacturer's information in many products.



**ORACLE**

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

An example of a new item type is a manufacturer item that holds the manufacturer's name, URL, contact email, and a description. It is much better to make a new item type and link other items to it than to duplicate values. In the example in the slide, CRS adds a manufacturer property to the product definition. The property links to a manufacturer item type. If the manufacturer's information changes, you only have to change it once. The example combines two different types of catalog extensions. The manufacturer is a new item type, but a property has also been added to an existing item type, the product, to link to the manufacturer item type.

CRS also adds a store repository that holds information about store locations.

## Design Decision: Catalog Extensions

What will your organization need to add to the catalog to:

- Differentiate products, categories, and SKUs for personalization?
- Store new item types?
- Integrate with non-ATG systems?
- Implement custom functionality, such as drop shipping?



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Drop shipping is when items are shipped directly from the manufacturer, rather than from the store's warehouse.

## Extending the Catalog: Example

ATG Store is expanding its shoe product line and needs to add a new SKU subtype for shoes to add a width property.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Adding properties to existing repository items is covered in the lesson titled “Personalization.” The example in this lesson focuses on creating subtypes of existing catalog objects.

## Item Subtypes

- New item types may be subtypes of existing items.
  - Similar to object-oriented subclassing
- Subtypes inherit all properties of supertype.
- Subtypes may add additional properties.

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Extending Catalog Item Types

1. Create one or more new tables to store added properties.
2. Extend /atg/commerce/catalog/custom/  
customCatalog.xml to:
  - a. Include your subtype
  - b. Define the subtype item descriptor and its properties
3. Add the subtype to CatalogTools.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

When you extend the customCatalog.xml file as part of step two, you do two things:

- Extend the type property to include your subtype. The Product Catalog Repository is configured for subtyping by default.
- Define the subtype item descriptor and its properties.

## 1. Create a New Table

Table: dcs\_sku

sku_id	display_name	sku_type
sku1334	Favorite trouser – Size:2 – Color: Grey	0 (base sku)
sku1351	Flowing A-line skirt – Size: 12 – Color: Red	0 (base sku)
sku1197	Varsity trainer – Size: 6 – Color: Blue	300 (shoe sku)

Table: mys\_shoe\_sku

sku_id	width
sku1197	AA

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

In the slide example, you see a default table called dcs\_sku that contains all SKUs. It has a sku\_type property for handling subtypes. The example adds a new table called mys\_shoe\_sku that links on the sku\_id property and adds the width column. Subsequent slides show how you can configure the XML to map the repository to the database.

## 2a. Extend the Type Property

/atg/commerce/catalog/custom/customCatalog.xml

```
<item-descriptor name="sku">
  <table name="dcs_sku">
    <property name="type">
      <option value="shoe" code="300"/>
    </property>
  </table>
</item-descriptor>
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The SKU, product, and category item types have already been configured to allow subtypes. It does not matter what number is chosen for the value of the option's code; however, this number should not overlap with any of the other options. To avoid conflicting with present or future code values, choose a high number.

You add this XML file to your module layer. This will add your configuration to the underlying configurations for this table.

## 2b. Define the Subtype Item Descriptor

The item descriptor for a subtype item has a:

- super-type attribute
- sub-type-value attribute

```
<item-descriptor name="shoe_sku"
    display-name="Shoe SKU"
    super-type="sku" sub-type-value="shoe">
    <table name="mys_shoe_sku" type="auxiliary"
        id-column-name="sku_id">
        <property name="width"
            column-name="width"
            data-type="string"/>
    </table>
</item-descriptor>
```

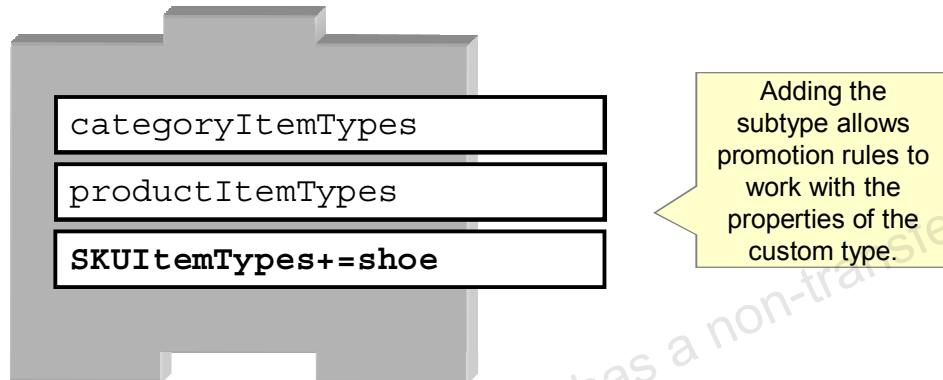


Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

In the example in the previous slide, the new shoe subtype was added to the sku item descriptor's type property with an option value of shoe. In the example in this slide, you are setting the super-type attribute to the sku item descriptor and the sub-type-value to match the relevant option value defined in the sku's type property. The example then defines the width property in the mys\_shoe\_sku table.

### 3. Add Subtypes to CatalogTools

/atg/commerce/catalog/CatalogTools



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Custom subtypes must be added to the CatalogTools component. This allows promotion rules to work with the properties of the custom types. You add the subtype to the appropriate property, depending on whether you are extending the category, product, or sku item descriptor to add a subtype. In the example in the slide, the shoe subtype is added to the SKUItemTypes property in your custom module layer.

# Quiz

Which of the following is the navigational endpoint of the catalog?

- a. Catalog
- b. Category
- c. Product
- d. SKU



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Answer: c

When navigating through your store, the customer does not see the catalog at all. A category is a way of organizing products directly or in a subcategory. The product page shows the SKU options, so is the navigational endpoint.

## Summary

In this lesson, you should have learned how to:

- Describe the structure and objects of the product catalog
- Extend the product catalog to add custom properties, item types, and subtypes



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## For More Information

### *ATG Commerce Programming Guide*

- Using and Extending the Product Catalog



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Practice 5 Overview: Extending the Catalog

This practice covers extending the product repository item to add a new Boolean taxfree property.



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Multisite Objects and Site Assignment



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to:

- Describe
  - The multisite feature
  - The objects in the site repository
- Extend the site repository
- Describe how sites are assigned to a request

## Road Map

- Multisite objects and architecture
- Site repository customization
- Site assignment



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Running Multiple Sites

- Many ATG customers use a single ATG installation to serve multiple sites.
- Sites share some components and content, but not all.
- Examples include regional sites, brand sites, and “micro” sites for specific events, such as sales or special product launches.
- Multisite is an optional feature.

 ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

**Note**

Only sessionwide components are shared.

## Multisite: Examples



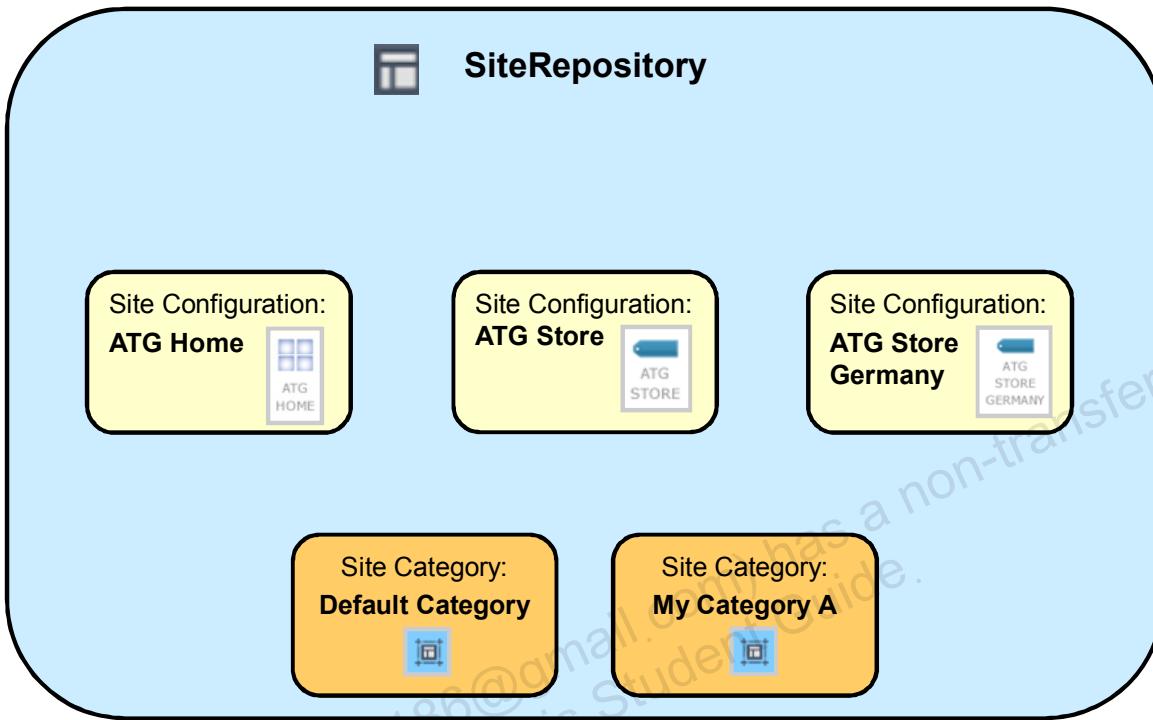
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE®

Three common multisite approaches are described in the slide:

- Affiliated sites are “sibling” sites that have largely independent catalog content that differs by theme or brand, with little or no overlap between the items sold at the two sites.
- A microsite is a relatively small site that is a child of a larger parent site. It typically shares many resources with parent site. Examples include a specific product line, seasonal items, or items associated with a specific event.
- Country stores are “sibling” sites, but catalog content may differ by country or locale. Typically, they do not share a shopping cart because of different currencies and shipping rules.

# Multisite Repository and Item Types



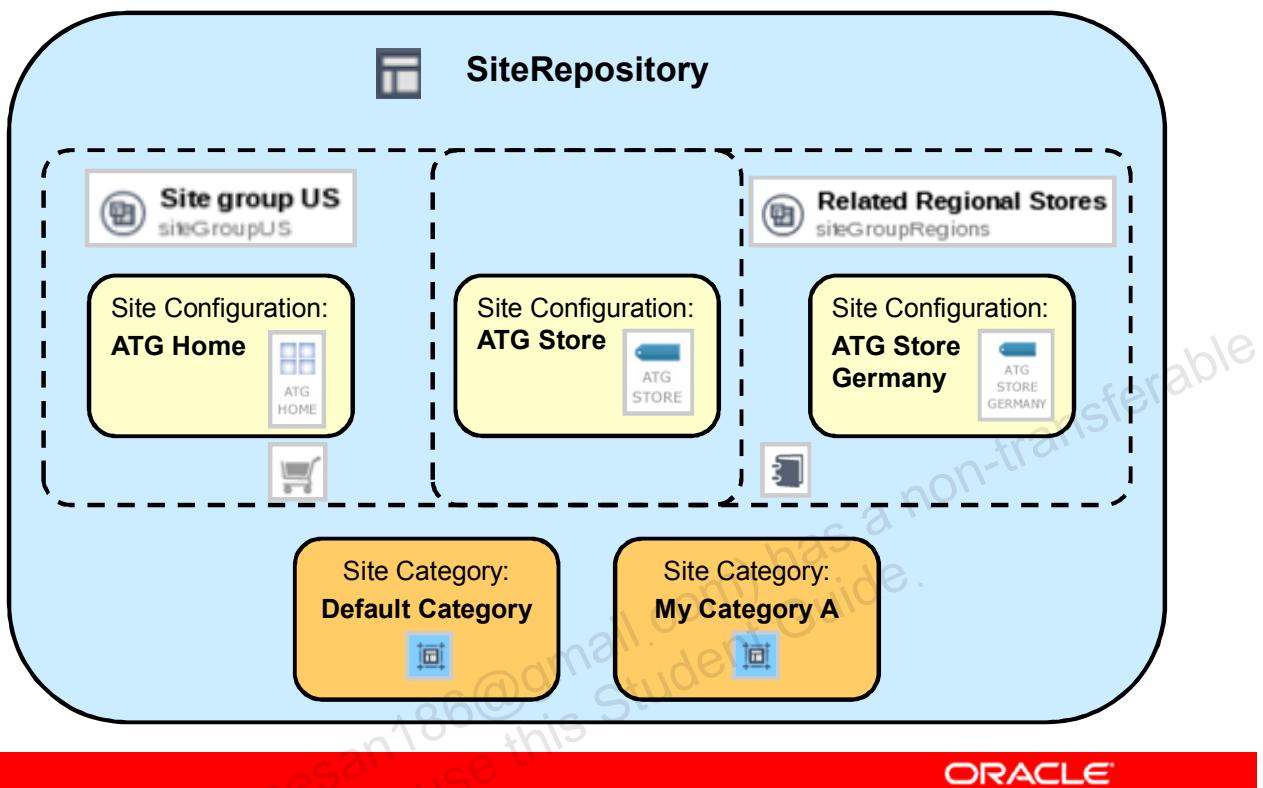
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The site repository stores the site configurations (each site object), site groups, and site categories. Site configurations represent each site with its configurations. Site categories are templates for creating sites in the Business Control Center (BCC). Site groups are discussed in more detail in the next slide.

The full path of the site repository is /atg/multisite/SiteRepository. Repository item types are defined in /atg/multisite/siteRepository.xml.

## Multisite Item Types: Site Groups



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

A site group is a way of grouping site configurations and determining what options the sites in the group share. Site groups have two main purposes:

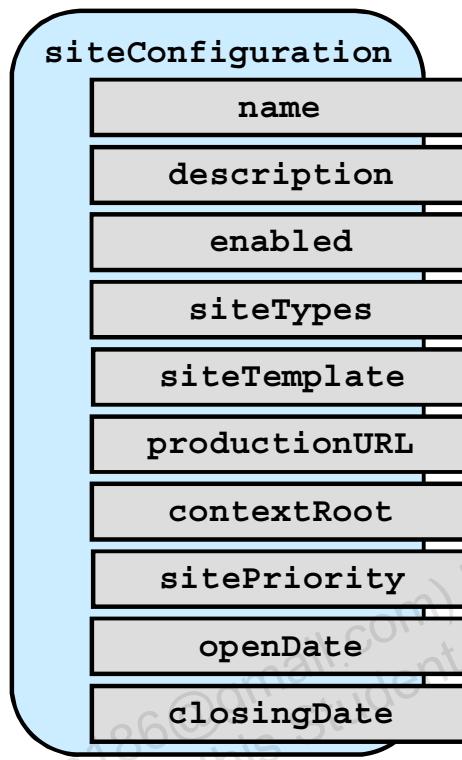
1. Provide a logical site grouping, such as linking between alternate sites or limiting promotion usability
2. Limit sharing of session-scoped components, such as the shopping cart

Sites can be in many groups, but cannot be in two groups that share same components. For example, one site cannot be in two groups that share the cart.

In the example in the slide, you see two site groups. The first site group, Site group US, contains the ATG Home and ATG Store sites and shares the shopping cart (a session-scoped component). The second site group, Related Regional Stores, contains the ATG Store and ATG Store Germany sites. These two sites share the catalog, but do not share a shopping cart.

**Note:** The shopping cart is not a component that you can select when configuring site groups. CRS, however, configures the same default catalog for the ATG Store and ATG Store Germany sites.

## Default siteConfiguration Properties: Part One



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

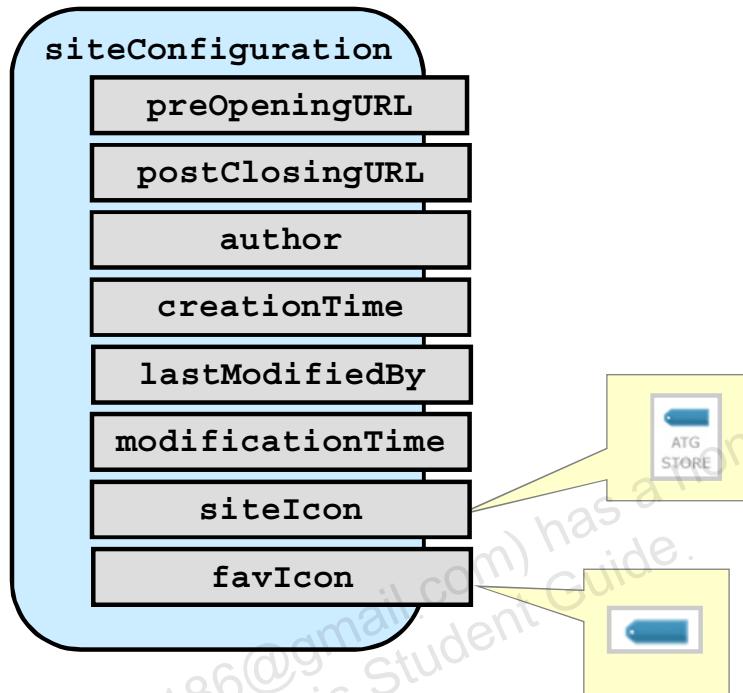
The next few slides highlight some of the properties of the `siteConfiguration` object in the site repository. Properties include:

- `name`: The name of the site configuration, such as “ATG Store”
- `description`: An optional description of the site
- `enabled`: A Boolean that determines whether the site appears on your staging or production server. When you no longer want a site to appear, it is a best practice to disable the site (`enabled=false`) rather than delete it. That site may have been used in user segments, the catalog, promotions, and more throughout ATG, and this keeps the references in place.
- `siteTypes`: Used in the BCC to determine which sites to display in which contexts, for instance, when selecting the sites for a promotion. If `siteTypes` is null, the site is considered to be a member of all types. This differentiates between Commerce and Self-Service sites. Commerce sites appear in ATG Merchandising in the BCC.
- `siteTemplate`: The site category or template used to define the site in the BCC. This is an item mapping. For more information on item mapping, see the appendix titled “Multisite and ATG Applications.”

## **siteConfiguration Properties**

- `productionURL`: The main URL substring used to identify the site. It should not contain the URL protocol, such as `http://` or `https://`. There is also an `additionalProductionURLs` property that holds additional URL substrings for the site.
- `contextRoot`: Used when `productionURL` contains the context root for a virtual web application. It identifies the context root of the underlying web application.
- `sitePriority`: Used by page droplets to order the sites. A lower number signals a higher priority. It can also be used by an application to order sites or select from multiple sites. It is also used as the last-resort criterion for assigning a user to a site at the beginning of a session.
- `openDate`: The date at which the site becomes available
- `closingDate`: The date after which the site is no longer available

## Default siteConfiguration Properties: Part Two



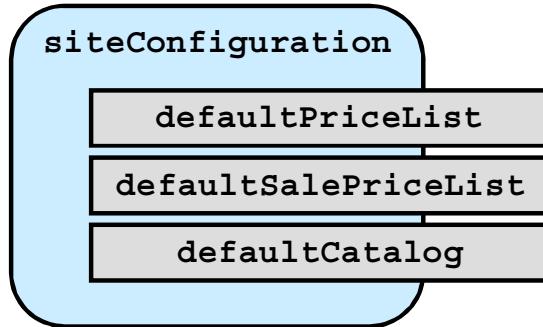
ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The next few slides highlight some of the properties of the `siteConfiguration` object in the site repository. Properties include:

- `preOpeningURL` and `postClosingURL`: The URLs to which you direct shoppers before the opening date or after the closing date for the site. A similar property, `siteDownURL`, is the URL to which shoppers are redirected when the site is down. When this happens, `siteDownRedirect=true` is appended to the query string of the URL.
- `author`: The person who created the site configuration
- `creationTime` and `modificationTime`: The time stamp for when the site was created or last modified
- `lastModifiedBy`: The person who last modified the site configuration
- `siteIcon`: The path to the graphic file used as a site icon in the JSPs
- `favIcon`: The path to the graphic file used as the site's favIcon. This icon is displayed in the browser tab and in bookmarks and favorites list.

## siteConfiguration Properties: DCS Additions



ORACLE

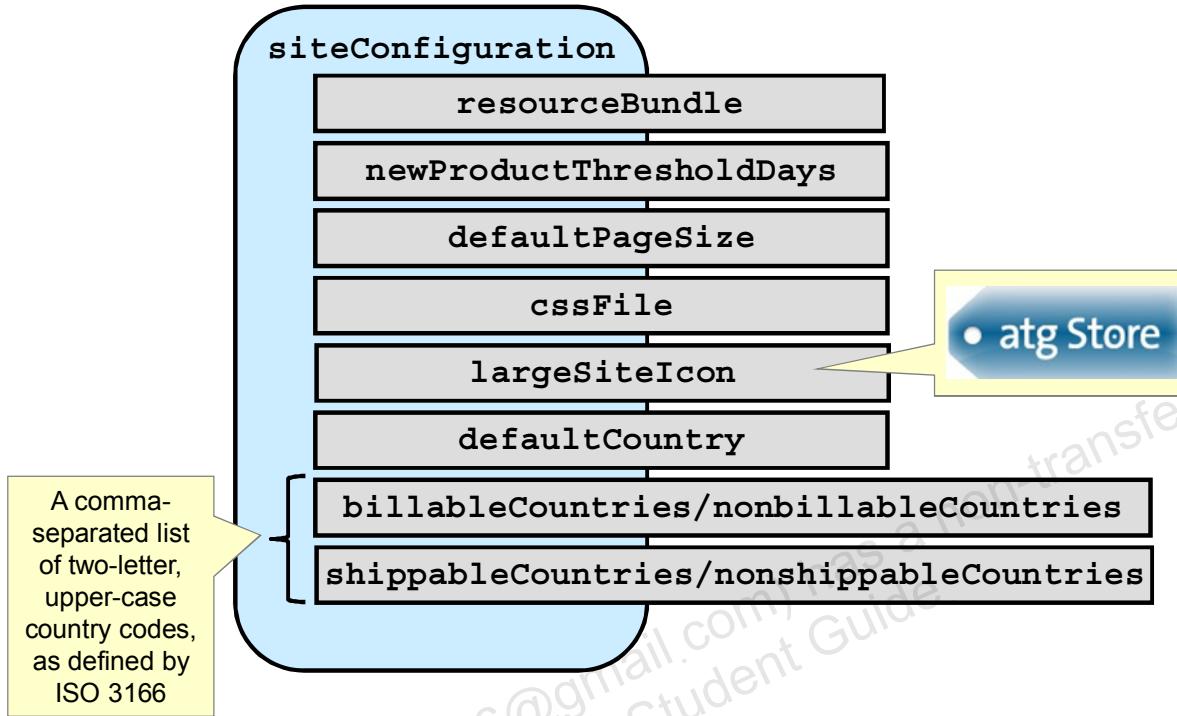
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The DCS module adds several properties to the `siteConfiguration` repository item:

- `defaultPriceList`: The link to a price list repository item
- `defaultSalePriceList`: The link to a price list repository item
- `defaultCatalog`: The link to a catalog repository item

**Note:** You will learn more about price lists in the lessons titled “Pricing Architecture” and “Displaying and Extending Pricing.”

## siteConfiguration Properties: CRS Additions, Part One



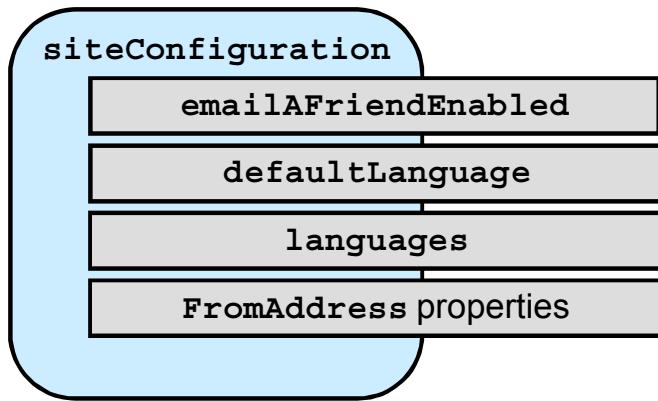
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

CRS also extends the `siteConfiguration` definition. Properties added by CRS include:

- `resourceBundle`: The default resource bundle for the text that appears on the site
- `newProductThresholdDays`: The number of days to consider a product “new”
- `defaultPageSize`: The default number of products to display on category or search pages
- `cssFile`: The path to the style sheet for this site
- `largeSiteIcon`: The large graphic to use on JSPs to represent the site
- `defaultCountry`: The default country for localization
- `billableCountries` and `nonBillableCountries`: The countries that are allowed or not allowed in billing addresses
- `shippableCountries` and `nonshippableCountries`: The countries that are allowed or not allowed in shipping addresses

## siteConfiguration Properties: CRS Additions, Part Two



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Site Categories

- Site categories are templates used to create site configurations.
  - Site categories are represented by the `siteTemplate` item type.
- They determine how site configuration fields are presented in the BCC.
- You can create as many site categories as you need for different types of sites.
- ATG Commerce provides one out-of-the-box site category called “Default Category.”



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The out-of-the-box “Default Category” site category includes all properties, all editable, with no default values. You can create a custom site category that gives you finer control over the appearance and behavior of site category properties. Options include:

- Hiding one or more properties
- Making a property read-only
- Setting default values for properties

## Road Map

- Multisite objects and architecture
- Site repository customization
- Site assignment



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Customizing Site Repository Items

- Custom properties should be added to the main type:
  - siteConfiguration
  - siteGroup
  - siteTemplate
- New properties are rendered in the BCC automatically.
- Best practices:
  - Site repository items are not intended to be subtyped.
  - Developers should not remove properties.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

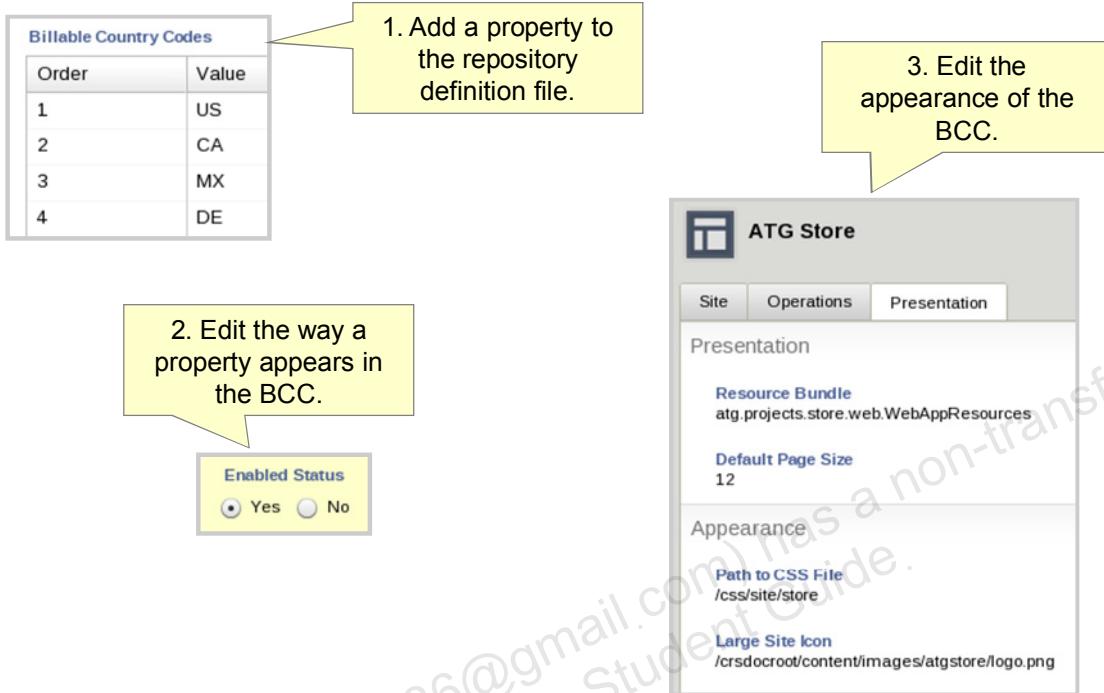
The slide lists the actual names of the item descriptors in the site repository. There are two important things to note when customizing site repository items:

- Site repository items are not intended to be subtyped. The components that handle sites are not written to handle subtypes with varying properties. The “site type” property does not indicate a different repository item type for subtyping. Instead, all sites are instances of siteConfiguration.
- You should not remove properties from the site repository. If you do not use the property, you can create a new site category that hides that property.

Properties are added by using XML layering. You customize in your module layer.

**Note:** The siteTemplate item descriptor refers to site categories.

# Customizing the BCC



The diagram illustrates three levels of customization for the Billable Country Codes (BCC) in ATG:

1. Add a property to the repository definition file. This is shown by a callout pointing to a table titled "Billable Country Codes" with columns "Order" and "Value". The table contains four rows: Order 1 (Value US), Order 2 (Value CA), Order 3 (Value MX), and Order 4 (Value DE).
2. Edit the way a property appears in the BCC. This is shown by a callout pointing to a "Enabled Status" field containing two radio buttons: "Yes" (selected) and "No".
3. Edit the appearance of the BCC. This is shown by a callout pointing to a screenshot of the "ATG Store" configuration interface under the "Presentation" tab. It shows settings for a "Resource Bundle" (atg.projects.store.web.WebAppResources), "Default Page Size" (12), and "Appearance" (Path to CSS File: /css/site/store, Large Site Icon: /crsdocroot/content/images/atgstore/logo.png).

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

There are three levels of customizations that you can do in the BCC:

1. Add a property to the repository definition. This is the simplest customization. When you add a property to the repository definition for a repository that appears in the BCC (such as the product catalog or the site repository), the property automatically appears. You saw an example of this when you edited the product catalog in the practices for the lesson titled "Catalogs." You can also hide a property or designate a property as read-only in the BCC by editing the property in the repository definition (see the next slide). The example in the slide shows the Billable Country Codes property that is added by CRS to the site configuration item descriptor.
2. Edit the way the property appears in the BCC. The BCC uses view mappings to determine how different types of properties should appear in the BCC. If you add a new property that you would like the BCC to handle differently, you would use view mappings. This topic is out of the scope of this course, but you can learn a bit about item mapping in relation to site categories in the appendix titled "Multisite and ATG Applications." The example in the slide shows the `enabled` property. By default, a Boolean property appears with Yes and No radio buttons. If you wanted it to appear differently, you would map the item to a different view.
3. Edit the appearance of the BCC. The BCC is an Adobe Flex application. You can edit the BCC user interface (UI). For example, CRS adds a new tab to the Site Administration area to hold CRS-specific customizations to the site configuration.

## Hiding and Disabling Fields in the BCC

You can hide or disable fields in the BCC through the repository.

To hide a property

```
<property name="example" hidden="true".../>
```

To disable a property  
(making it read-only)

```
<property name="example" ...>  
  <attribute name="uiwritable" value="false"/>  
</property>
```

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

You have some flexibility to control repository properties in the BCC. Rather than removing a field from the repository definition, you can hide the property by adding the `hidden` attribute and setting it to `true`. To disable a field (make it read-only), you add the `uiwritable` attribute and set it to `false`. The property still appears in the BCC, but the user cannot edit it.

**Note:** If you make these changes to the site configuration, the changes apply to all appearances of that property in all site categories.

## Road Map

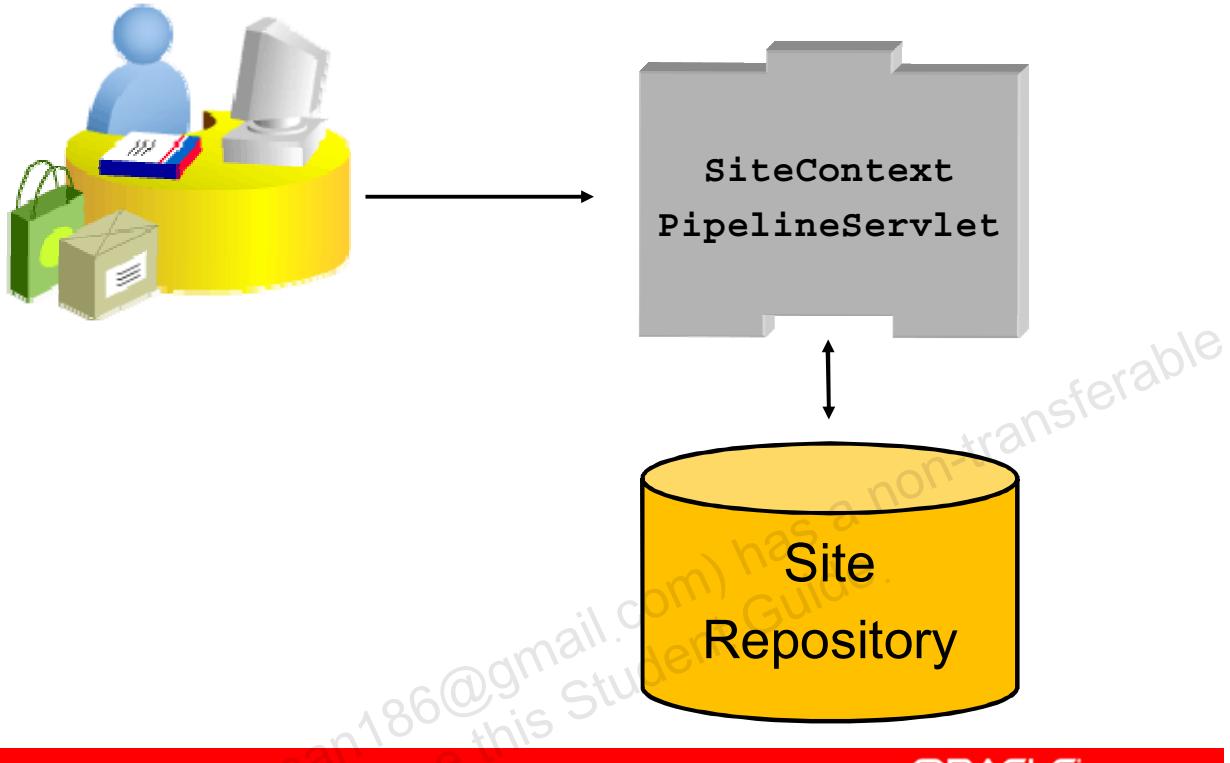
- Multisite objects and architecture
- Site repository customization
- Site assignment



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Site Assignment



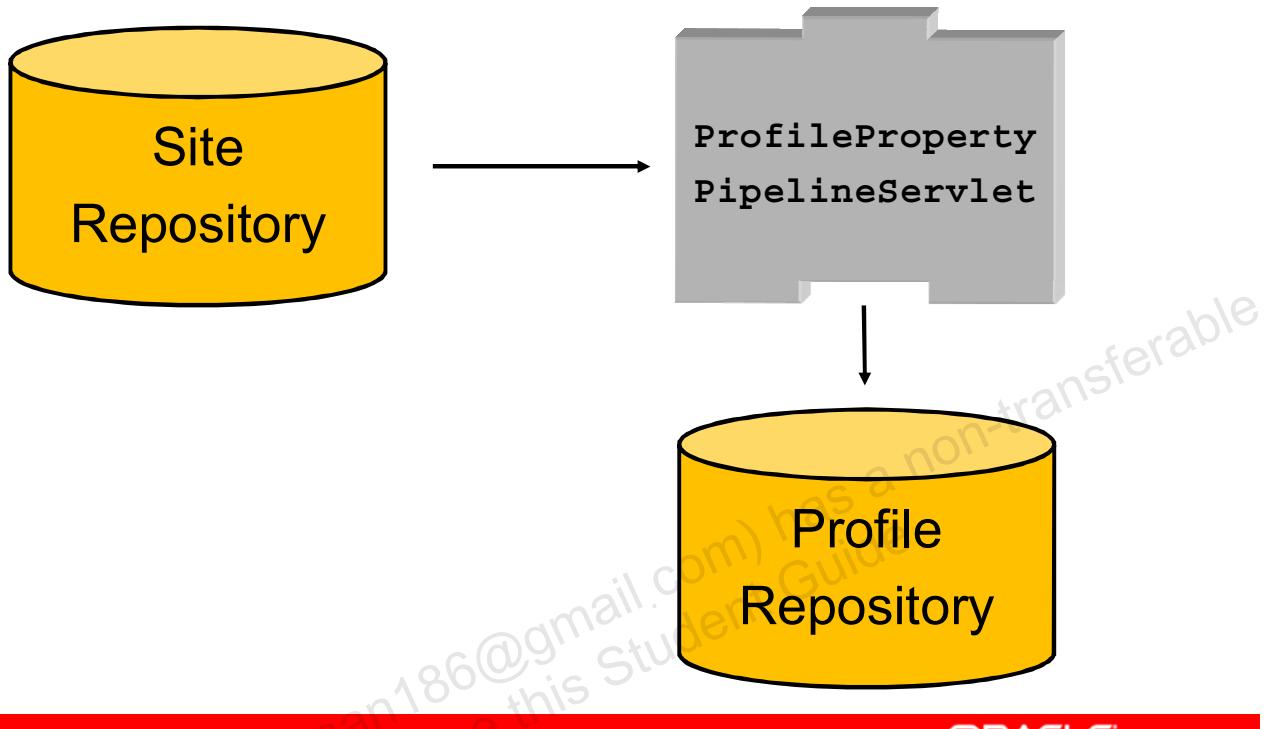
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Every HTTP request to an ATG server is processed by the servlet pipeline. One of the servlets in the pipeline is `SiteContextPipelineServlet`. This servlet uses information from the Site Repository to determine which site to assign the request to, generally based on the URL of the incoming request. If the request URL does not match with any site URLs, ATG uses the defaults set in the `CatalogTools` and `PriceListManager` components.

**Note:** You will learn more about the commerce pipelines in the lesson titled “Pipelines and Shipping.” For more information on setting the site context, see Appendix B: Additional Multisite Information.

## ProfilePropertyPipelineServlet



ORACLE

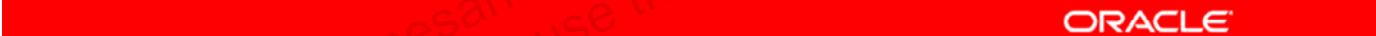
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Once the site is determined, `ProfilePropertyPipelineServlet` sets any profile properties that depend on the site, such as the catalog and price list. Out of the box, ATG sets only the catalog and price lists (based on the default catalog and default price list and sale price list settings in the site's configuration).

You can extend `ProfilePropertyPipelineServlet` to set other profile properties as needed.

## Disabling Multisite

- Multisite is enabled by default when installing or upgrading to ATG 10.x.
- If no sites are defined, ATG behaves as if multisite is disabled.
- Multisite can also be explicitly disabled by setting `SiteContextPipelineServlet.enabled` to `false`.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Multisite Responsibilities

- Developers are responsible for:
  - Creating a site-aware ATG application
  - Defining site categories (site templates)
  - Defining which components can be shared between sites
- Business users are responsible for creating:
  - Site configurations
  - Site groups

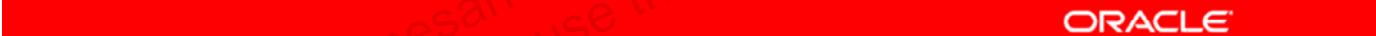


Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The default components that can be shared between sites are the shopping cart and product comparisons. You will learn more about sharing (limiting) components between sites in the lesson titled “Multisite and ATG Applications.”

## Design Decisions: Multisite

- Will the store use the multisite feature?
- If so, what types of sites will be created?
  - Brand, regional, micro-sites, other?
- Will any custom site properties be needed?
- Which components will be needed to be shared between sites?



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Summary

In this lesson, you should have learned how to:

- Describe the architecture and objects of the site repository
- Extend the site repository

## For More Information

- *ATG Multisite Administration Guide*
- *ATG Commerce Programming Guide*
  - Configuring Commerce for Multisite
- *ATG Commerce Reference Store Overview*
  - Multisite Features



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Practice 6 Overview: Customizing and Extending Multisite

This practice covers the following topics:

- Extending the `siteConfiguration` and `siteTemplate` objects to add a new Boolean `reward` property
- *Optional:* Altering the display of `siteConfiguration` properties in the BCC



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2015, Oracle and/or its affiliates.

Ganesan Sree (ganesan186@gmail.com) has a non-transferable  
license to use this Student Guide.

## Multisite and ATG Applications

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to:

- Create and extend site categories (templates)
- Limit components by site group
- Create a site-aware ATG application

# Road Map

- Site templates
- Multisite and ATG applications
- Multisite and commerce



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## New Site Templates

- You create custom site templates to present different site “categories.”
- Templates can be used to vary
  - Default values
  - Whether fields are read-only
  - Other attributes of fields, depending on options available in the JSP fragment

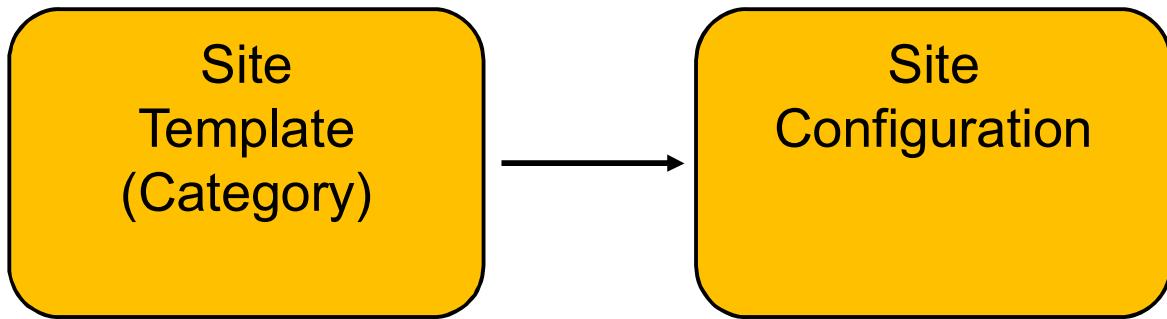


Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

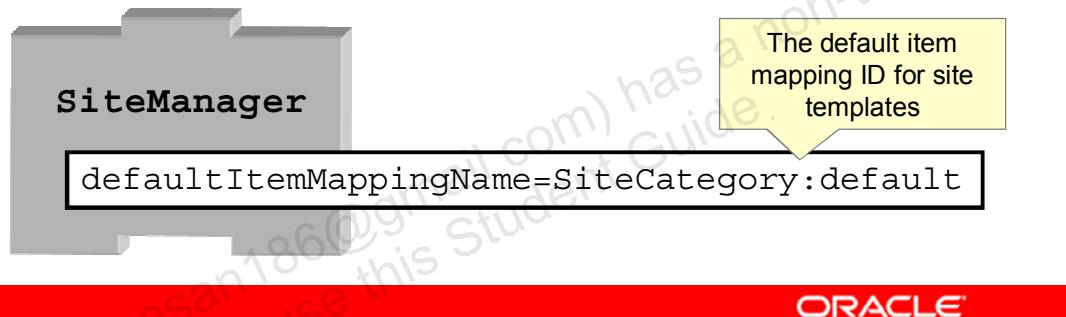
This course has referred to both site categories and site templates. They refer to the same thing, the `siteTemplate` item type in the site repository. This lesson refers to them as site templates. In the BCC, however, a site template is referred to as the Site Category. ATG provides one default site category that lists all of the properties as editable properties with no default values.

**Note:** If you want a value to be read-only on all site categories, regardless of the site template used, you can set a field to be read-only in the `siteTemplate` item.

## Site Templates



Templates use the BCC View Mapping system.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Site templates are used to create new sites (site configurations). They determine how the site configuration properties are displayed to the BCC user. The site template works in conjunction with the BCC View Mapping system. View mappings control how different types of properties appear in the BCC. For example, a Boolean property appears with radio buttons labeled "Yes" and "No," and a string property appears as a text input field. ATG provides a default item mapping for site templates called `SiteCategory:default`. That is set on the `SiteManager` component's `defaultItemMappingName` property. If you would like for certain properties to appear differently (for example, have a drop-down box with True and False options for a Boolean property), you would create your own item mapping scheme in the View Mapping system and edit the `defaultItemMappingName` property in your custom layer.

For more information about the View Mapping system, see Appendix B: Additional Multisite Information.

## Creating a New Site Template

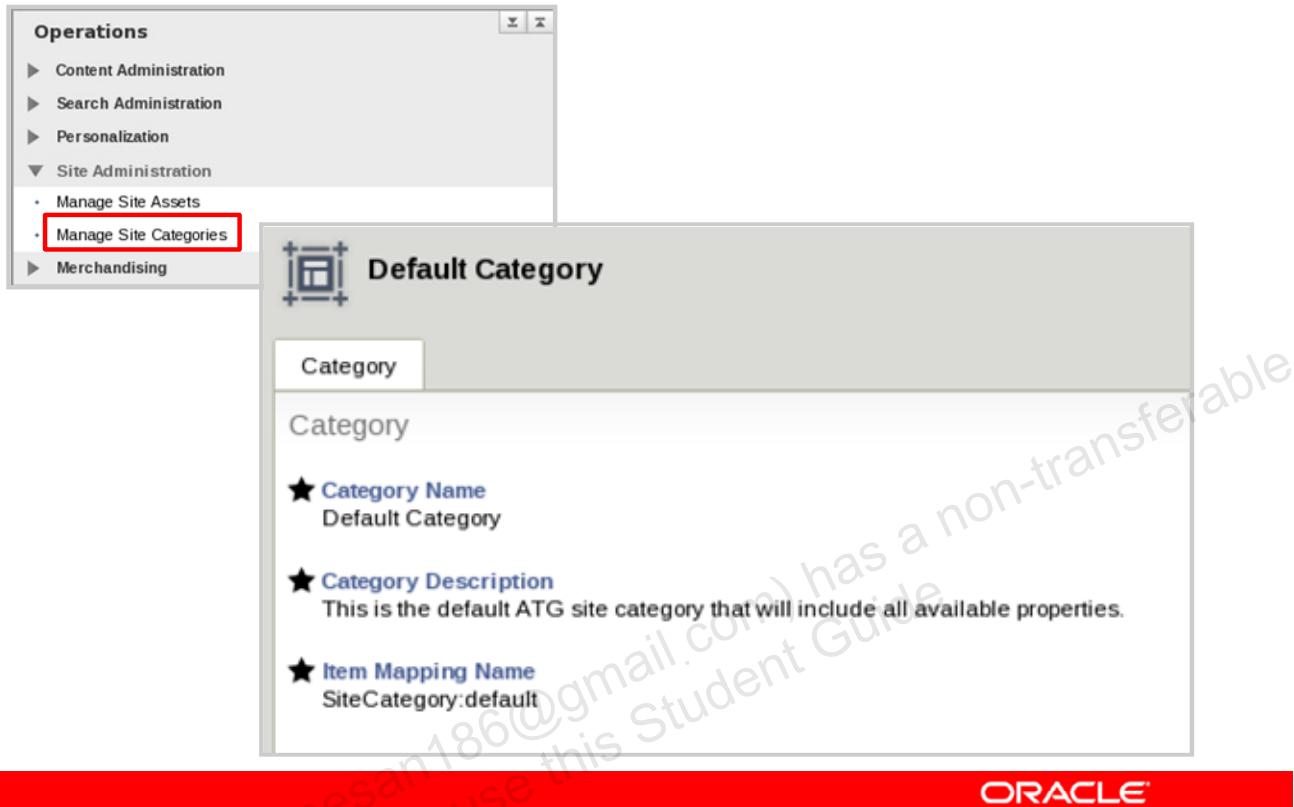
1. Create a new item mapping in the View Mapping Repository, if needed.
  - Site templates may be created solely to provide different default values, in which case a new item mapping is not necessary
2. Create a new site category item in the Site Repository.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

This lesson focuses on creating a site template that provides different default values and creating a new site category in the BCC. For more information about creating a new item mapping in the View Mapping Repository, see the appendix titled “Multisite.”

## Creating a New Site Category Item in the BCC



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

You create and manage site categories in the BCC. To access site categories, you click the “Manage Site Categories” option in the Site Administration area of the Operations menu. This action takes you to the Site Category Administration. The example in the slide shows the default category. For a new category, you specify three default properties:

- **Category Name:** The text that should be shown to BCC users when selecting the site category
- **Category Description:** An optional property to provide more details on this category to BCC users
- **Item Mapping Name:** The `itemMappingId` property to the ID of the appropriate item mapping. This is entered as “SiteCategory:<itemMappingId>.”

If you have added any custom properties to the `siteTemplate` item, those properties would also appear and could be set.

## Setting Default Values

Site templates can be used to vary default values for site configuration options. To do this, you need to:

1. Extend the `siteTemplate` repository item to have a property with same name as the relevant `siteConfiguration` property.
2. Set the default value on the site template through BCC.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

# Road Map

- Site categories
- Multisite and ATG applications
- Multisite and commerce



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Ways to Use the Multisite Feature

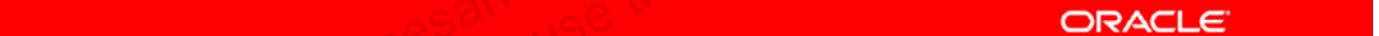
- Limit the sharing of components across sites
- Set component properties by site
- Use site droplets in JSPs
- Add custom listeners to site events
- Add processors to SiteSessionManager
- Extend custom repository items to include allowed sites

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Limiting Components by Site Group

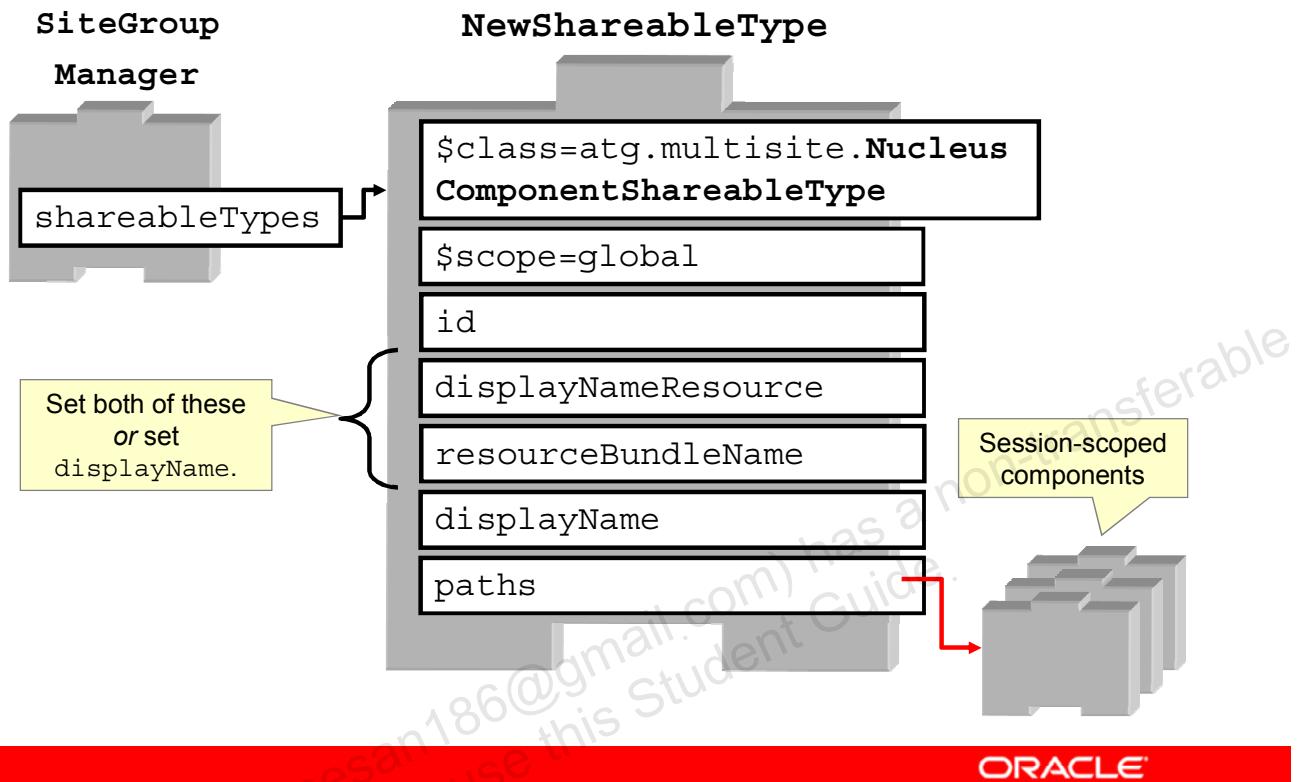
- By default, components are shared across all sites.
- Developers must define which components should be limited to sharing within their site group.
- Once a component is declared as shareable, by default it is NOT shared between sites.
  - It must be configured to be shared in a site group.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

# Creating a Component Shareable Type



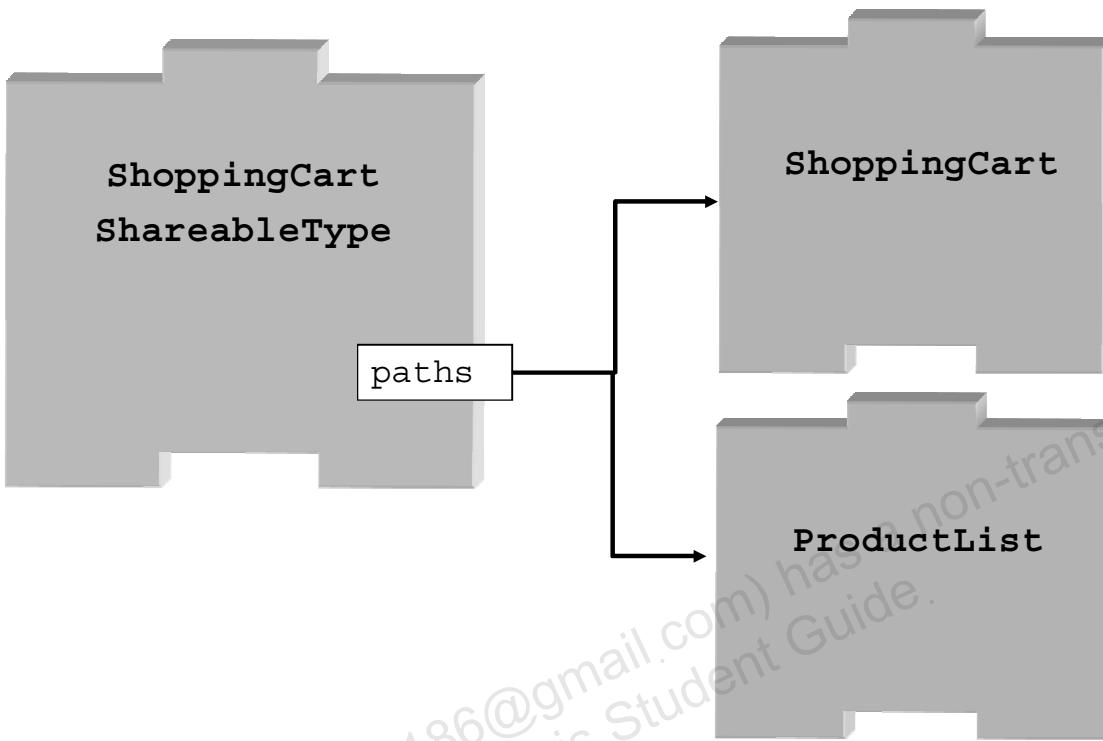
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

To create a component shareable type, you need to do the following steps:

1. Create a new global component based on the `atg.multisite.NucleusComponentShareableType` class.
2. Set the following properties of the component:
  - `id`: A unique string
  - `displayNameResource`: The key of the string in the resource bundle to use as the display name in the BCC
  - `resourceBundleName`: The resource bundle from which to look up the display name
  - `displayName`: The string to use to identify this shareable type in the BCC. It can be used instead of `displayNameResource` and `resourceBundleName` for unilingual sites.
3. Set the `paths` property to point to one or more session-scoped components that should be site-aware.
4. Add the new Shareable Type component to `/atg/multisite/SiteGroupManager.shareableTypes` property.

## Component Shareable Type: Example



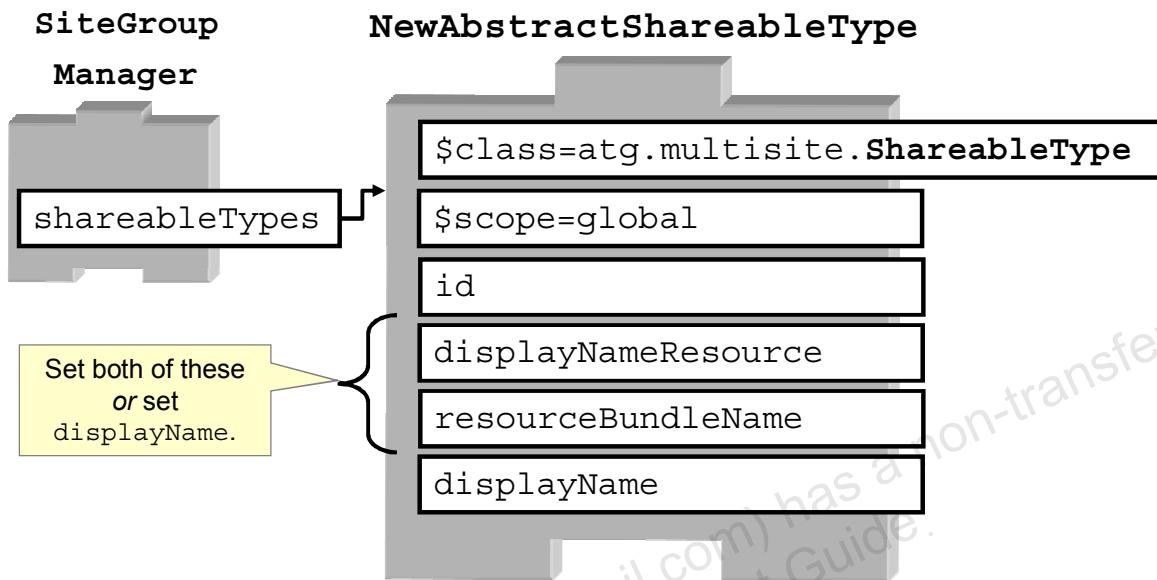
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, one `ShareableType` component, `/atg/commerce/ShoppingCartShareableType`, is pointing to the two default site-aware session-scoped components, `/atg/commerce/ShoppingCart` (a component representing what is in the user's cart) and `/atg/commerce/catalog/comparison/ProductList` (a component that handles product comparisons). Because they use the same `ShareableType` component, `ShoppingCart` and `ProductList` will be "bundled" for multisite purposes. In other words, business users are able to specify that a site group shares either both the `ShoppingCart` and `ProductLink` components, or neither of them, but not just one or the other. Developers can override this behavior to make it possible to share one without the other.

You will learn more about the `ShoppingCart` component in subsequent lessons.

## Creating an *Abstract Shareable Type*



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Abstract shareable types can be used to group sites for cross-site linking when sites do not share any components. Note that there is no paths property as you saw in the component shareable type example in the slide titled “Creating a Component Shareable Type.” Commerce Reference Store (CRS) provides an example of an abstract shareable type: /atg/store/RelatedRegionalStoresShareableType. An abstract shareable type acts as a placeholder in a site group. This way you can group sites together without needing to share anything. This is useful for multisite droplets that group sites together in a list to make it easy to move from one site group to another.

## Ways to Use the Multisite Feature

- Limit the sharing of components across sites
- Set component properties by site
- Use site droplets in JSPs
- Add custom listeners to site events
- Add processors to SiteSessionManager
- Extend custom repository items to include allowed sites



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Setting Component Properties by Site

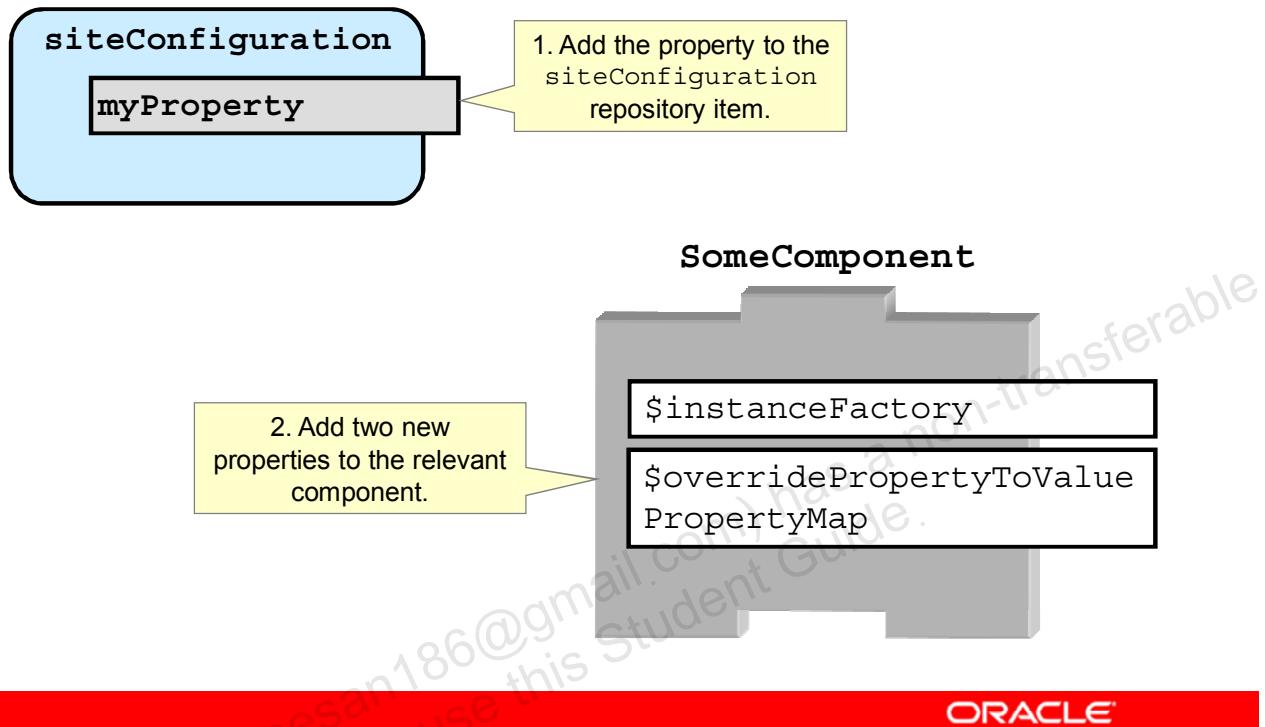
- Any component can have any property value set differently depending on the current site.
  - Best practice: Avoid setting properties by site for shareable components.
- Property values come from the specified property on the site repository item.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Steps to Configure Component Properties by Site



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

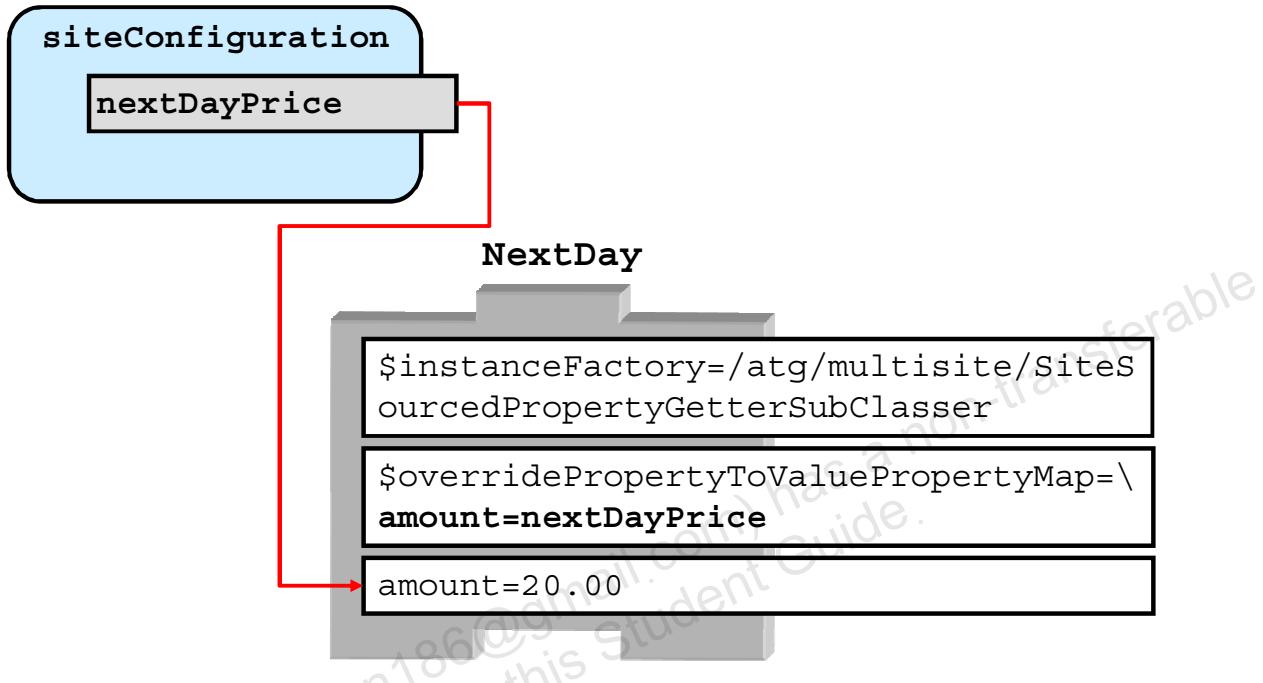
ORACLE

The steps to configure component properties by site are the following:

1. Extend the site repository item to add the property from which to read the site-specific value.
2. Add the following properties to the relevant component:
  - \$instanceFactory
  - \$overridePropertyToValuePropertyMap

The example in the slide shows a new siteConfiguration property called myProperty. In your component (SomeComponent, in the slide), you add your property to \$overridePropertyToValuePropertyMap, with the relevant component property as the key, and myProperty as the value. See the next slide for an example.

## Site-Specific Property: Example



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, you want to be able to set the price for next day shipping by site. The `siteConfiguration` object has been extended to add a new `nextDayPrice` property. The `/atg/commerce/pricing/shipping/NextDay` component's `amount` property is set to the value of the current site's `nextDayPrice` property. If the current site's `nextDayPrice` property is null, the default value specified in the component, `20.00`, is used.

The `$overridePropertyToValuePropertyMap` property ties the site configuration property and the component's property together in a property map.

## Ways to Use the Multisite Feature

- Limit the sharing of components across sites
- Set component properties by site
- **Use site droplets in JSPs**
- Add custom listeners to site events
- Add processors to SiteSessionManager
- Extend custom repository items to include allowed sites

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Using Site Droplets in JSPs

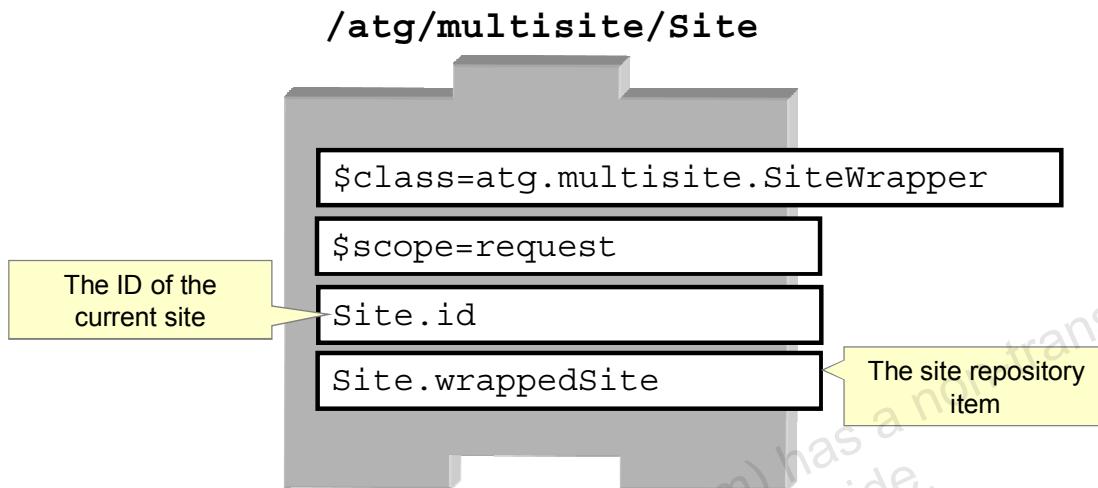
- ATG provides several site droplets for use in JSPs, including:
  - SiteLinkDroplet
  - SharingSitesDroplet
  - CartSharingSitesDroplet  A CRS extension
  - SiteIdForItem
- ATG configures a helpful out-of-the-box, request-scoped component, /atg/multisite/Site, for use in the JSPs.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The droplets listed in the slide are located in /atg/dynamo/droplet/multisite/.

## Accessing the Current Site



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Creating Links to Other Sites

```
<dsp:droplet name="ForEach">
  <dsp:param name="array"
    beanvalue="SiteManager.activeSites" />
  <dsp:oparam name="output">
    <dsp:param name="site" paramvalue="element" />
    <dsp:droplet name="SiteLinkDroplet">
      <dsp:param name="siteId" paramvalue="site.id" />
      <dsp:param name="path" value="/home/index.jsp" />
      <dsp:oparam name="output">
        <dsp:getvalueof var="siteUrl" param="url" />
        <dsp:a href="<%siteUrl%>">
          site.displayName
        </dsp:a>
      </dsp:oparam>
    </dsp:droplet>
  </dsp:oparam>
</dsp:droplet>
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The code example in the slide shows the JSP code for displaying a list of site links. It starts with a `ForEach` droplet that iterates through the `/atg/multisite/SiteManager` component's list of active sites. In the output of the `ForEach` droplet, the example first uses the `dsp:param` tag to rename the output of the `ForEach` droplet from `element` to `site`.

The `SiteLinkDroplet` generates the correct URL for a site. There are many input parameters that can be used to set various aspects of the resulting URL, including the protocol (http or https), the destination path, and the query parameters. In the example, the site's ID and path are set.

The code example assumes that all components referenced in the JSP were imported earlier in the JSP file.

**Note:** The `/atg/multisite/SiteManager` component can be used to retrieve the list of currently-active sites as repository items. Its `allSites` property would return all active and inactive sites.

## Retrieving Sites Within a Site Group

```
<dsp:droplet name="SharingSitesDroplet">
    <dsp:param name="shareableTypeId"
        value="atg.ShoppingCart" />
    <dsp:param name="excludeInputSite" value="true"/>
    <dsp:oparam name="output">
        <dsp:droplet name="ForEach">
            <dsp:param name="array" paramvalue="sites"/>
            ...
        </dsp:droplet>
    </dsp:oparam>
</dsp:droplet>
```

**Note:** CRS uses CartSharingSitesDroplet, an instance of SharingSitesDroplet, that returns sites sharing the shopping cart.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The SharingSitesDroplet can be used to retrieve the list of sites within a site's site group. Developers can optionally supply a site to the siteId input parameter; if none is supplied, the current site is used. The shareableTypeId input parameter indicates which site group to return. The excludeInputSite parameter determines whether the input site is included in the list of output sites.

This droplet will display the output op param if there are other sites in the site group. If not, the empty op param is displayed.

CartSharingSitesDroplet is an instance of SharingSitesDroplet. The CRS layer configures the following properties:

siteGroupManager=/atg/multisite/SiteGroupManager

siteContextManager=/atg/multisite/SiteContextManager

shareableTypeId=atg.ShoppingCart

## SiteIdForItem Droplet

- The SiteIdForItem droplet is used when displaying search results, cross-sells, or upsells.
- Given the ID of an item, it returns a single site on which the item is valid.
  - The item may be valid on multiple sites.
- Input parameters can influence the site returned. Examples include:
  - currentSiteFirst
  - shareableTypeId
  - includeInactiveSites and includeDisabledSites
- If the item belongs to more than one site satisfying the input parameters, the sitePriority property is used to determine which site to return.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Input parameters for the droplet include:

- currentSiteFirst: A Boolean. A value of true indicates a preference for the current site to be returned as the best possible match by looking for it in the item's site list. This is also dependent on the site being active (or the appropriate include flags being set).
- shareableTypeId: The ID of the shareable that will be used to get the list of potential sites
- siteId: A user-defined siteId value that will be used as a preference in the matching process
- includeInactiveSites: A Boolean. A value of true indicates a preference for a site in an inactive state to be considered for inclusion in the site-matching process.
- includeDisabledSites: A Boolean. A value of true indicates a preference for a site in a disabled state to be considered for inclusion in the site-matching process. By definition, a disabled site is inactive. To include a disabled site, both this flag and includeInactiveSites should be set to true.

## Ways to Use the Multisite Feature

- Limit the sharing of components across sites
- Set component properties by site
- Use site droplets in JSPs
- Add custom listeners to site events
- Add processors to SiteSessionManager
- Extend custom repository items to include allowed sites

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Add Listeners to Site Events

Custom components can be added as listeners for the following site-based events:

- StartSiteSession
- SiteChanged
- SiteVisit



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

You can register custom components to receive the event when that event is fired. That component can then take action based on the event. The events listed in the slide are site specific. To learn more about event listeners, see the section titled “Events and Event Listeners” in the chapter titled “Core ATG Services” in the *Platform Programming Guide*.

## Ways to Use the Multisite Feature

- Limit the sharing of components across sites
- Set component properties by site
- Use site droplets in JSPs
- Add custom listeners to site events
- Add processors to SiteSessionManager
- Extend custom repository items to include allowed sites

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Add SiteSessionManager Processors

- SiteSessionManager runs configured processor components when site sessions start and end, and when requests are made within a site context.
- Add a custom component to one of the following:
  - siteSessionStartProcessors
  - siteSessionEndProcessors
  - siteRequestProcessors
- The component must implement the matching interface:
  - SiteSessionStartProcessor
  - SiteSessionEndProcessor
  - SiteRequestProcessor



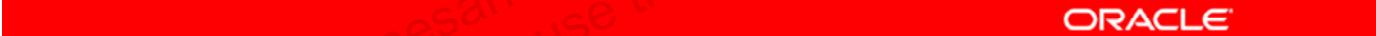
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ATG Commerce provides two examples of classes that add SiteSessionManager processors:

- One class implements the SiteSessionStartProcessor interface, `atg.multisite.ReferringSiteProcessor`. This processor sets the `referringSite` property on the requested site's SiteSession.
- Another class implements the SiteRequestProcessor interface, `atg.multisite.LastSiteVisitedProcessor`. Its implementation of `processSiteRequest()` sets the current session's last-visited site in the SiteSessionManager's `lastVisitedSite` attribute.

## Ways to Use the Multisite Feature

- Limit the sharing of components across sites
- Set component properties by site
- Use site droplets in JSPs
- Add custom listeners to site events
- Add processors to SiteSessionManager
- Extend custom repository items to include allowed sites



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Extend Custom Repository Items

- Add a property that allows content managers to specify the valid sites for this item type.
  - Example: Add the sites property to a news custom content item.
- Use the current site in an RQL query:

```
<dsp:param name="currentSite"
            beanvalue="/atg/multisite/Site.id"/>
<dsp:droplet name="RQLQueryForEach">
    <dsp:param name="itemDescriptor" value="news"/>
    <dsp:param name="repository"
                value="/myapp/NewsRepository"/>
    <dsp:param name="queryRQL"
                value="sites INCLUDES ITEM (:currentSite)"/>
    ...

```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The code example in the slide assumes a custom repository called NewsRepository, which has been created to contain news items. One of the properties of the news item type is sites, which stores the valid sites for a particular news item. The first two lines of the example (dsp :param) store the ID of the current site. The example continues with the RQLQueryForEach droplet, which is used to retrieve only those news items that are valid on the current site.

## Quiz

Which of the following statements describe components and sites? (Select all that apply.)

- a. By default, components are shared across all sites.
- b. You can define which components can be limited to sharing within their site group.
- c. You can configure a component's properties to be set by site.
- d. You should set properties by site for shareable components.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

**Answer:** a, b, c

# Road Map

- Site categories
- Multisite and ATG applications
- Multisite and commerce



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Multisite Extensions to Commerce

- Multisite extends the following droplets to make them site-aware:
  - CatalogItemLookupDroplet
  - OrderLookupDroplet
- Multisite adds a few new key objects:
  - CartSharingFilterDroplet
  - SiteForkProcessor



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

CatalogItemLookupDroplet has been extended to check whether the requested item belongs to current user's current site. If not, the `wrongSite` output parameter is displayed. Two site-related input parameters are:

- `sites`: Used to override which site or sites to allow
- `filterBySite`: Can be set to `false` to disable site filtering

OrderLookupDroplet (covered in more detail in the lesson titled “Orders”) has been extended to be site-aware. You can restrict order lookup based on the site context. `siteScope` can be set to `current`, `all`, or a shareable type ID, such as `atg.ShoppingCart.siteIds` can be set to specific sites, which narrows the results returned by `siteScope`.

/atg/commerce/collections/filter/droplet/CartSharingFilterDroplet filters out catalog items whose sites do not share a cart with the current site. It uses the /atg/registry/CollectionFilters/CartSharingFilter component to do the filtering.

The `atg.service.pipeline.processor.SiteForkProcessor` class can be used to create pipeline links that fork based on the site. You configure the `sitesMap` property to map site IDs to integers, and configure the pipeline XML to indicate which fork to take based on which integer is returned.

## Multisite and Scenarios

- Scenarios have two site-specific events:
  - Site visit
  - Switches site context
- Most events have had site and site's added as parameters.

 Switches site context  
site is switched to is  
ATG Home

 Site visit site is  
storeSiteUS

 Session starts Site is  
ATG Home

 Order submitted Site is  
ATC Store

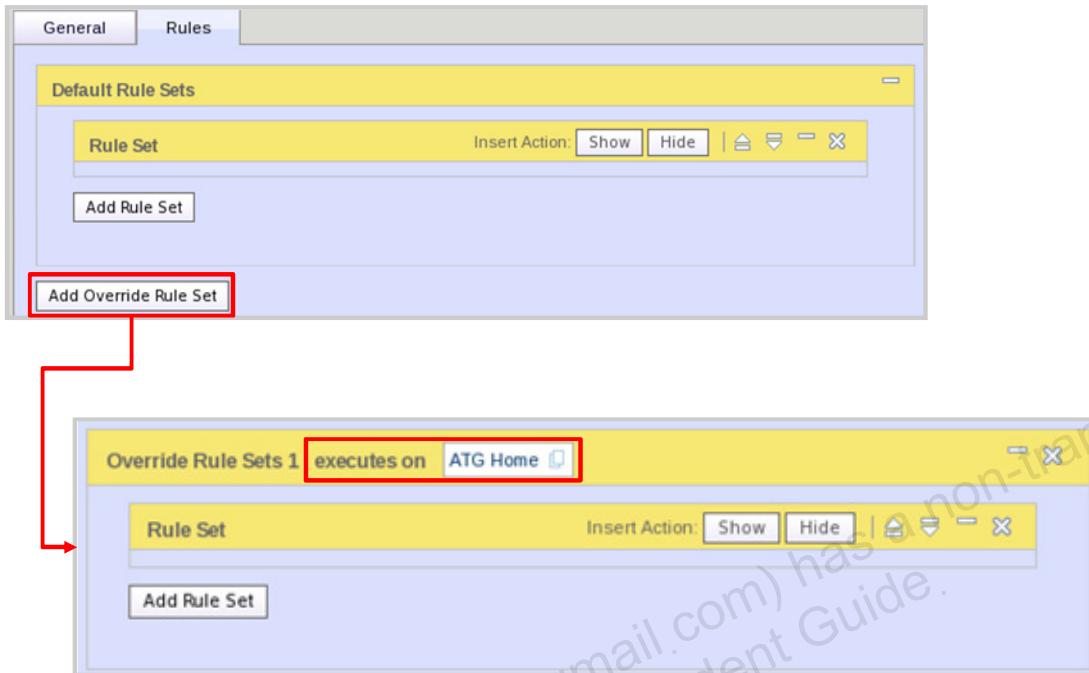


Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Some of the events that are not site-aware:

- An email is received
- Email is sent
- Clicks a link
- Price changed
- Profile property updated by user
- Uses promotion
- Views (a repository item)
- Visits (a page)

# Site-Aware Targeters

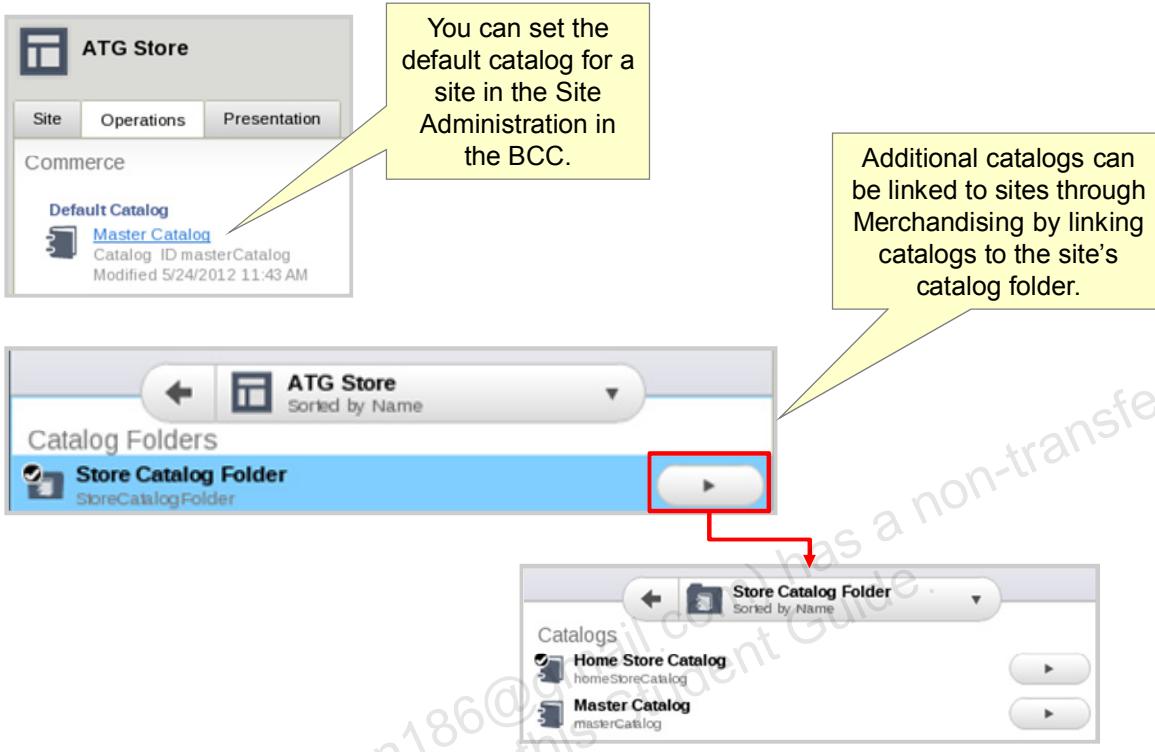


ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

To make site-aware targeter rules, click **Add Override Rule Set** on the targeter's **Rules** tab. Select the site or sites that will trigger the override rule. Build the targeter rules as usual with the override rule set. You can add as many override rule sets as desired.

## Sites and Catalogs

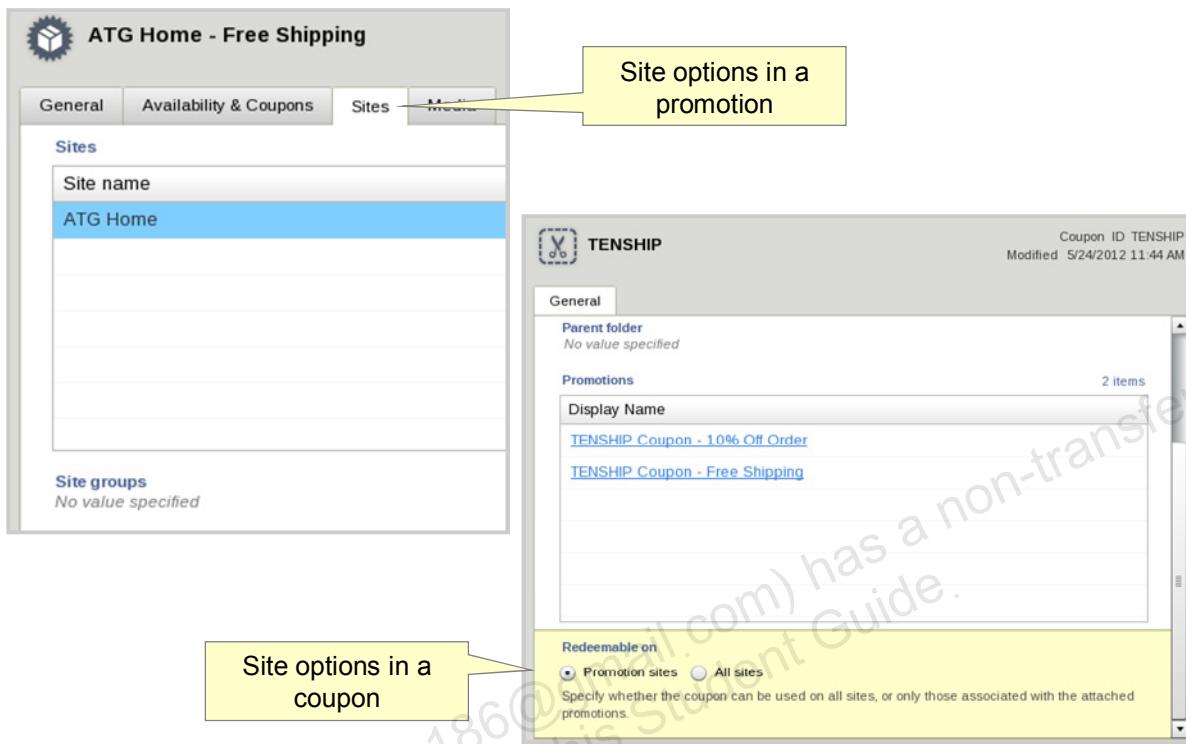


Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

Sites can have multiple catalogs and catalogs can be linked to multiple sites. In the example in the slide, the Home Store Catalog has been linked to the ATG Store site through the ATG Store's Store Catalog Folder. Adding a catalog folder under a site makes it possible to have more than one catalog linked to the site. If you do not create a folder, you will only have one catalog in the site.

## Site-Aware Promotions and Coupons



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

You can specify the site or sites on which a promotion is active. You can also specify the site group. This is a way to limit a promotion to specific sites, if desired. If no sites or site groups are specified, promotions will be active on all sites.

Coupons have a property to specify whether the coupon can be used on all sites, or only those associated with the attached promotions.

## Summary

In this lesson, you should have learned how to:

- Create and extend site categories (templates)
- Limit components by site group
- Create a site-aware ATG application

## For More Information

- *ATG Commerce Programming Guide*
  - Configuring Commerce for Multisite
- *ATG Programming Guide*
  - Multisite Request Processing
- *ATG Multisite Administration Guide*



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Practice 7 Overview: Creating and Using New Site Categories

This practice covers the following topics:

- Extending the siteTemplate item to add a Boolean reward property
- Creating a new site category for reward sites that sets the new reward property to true
- Creating a new rewards site
- Creating a new Related Reward Site abstract shareable type
- Creating a new Rewards Sites site group



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

# 8

## Orders: Overview

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to describe:

- The default life cycle of an order
- The out-of-the-box shipping and payment options

# Road Map

- Orders
- Shipping and payment methods

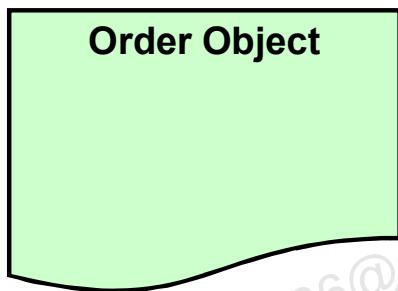


ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Orders

- ATG uses the Order object to represent a single purchase.
  - It may contain many items.
  - Items within the order may be shipped to different addresses or paid for by different methods.
- Other commerce products often use the word “cart” for this object.
  - ATG uses “cart” differently, as shown in the next slide.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Shopping Cart

ATG Commerce's shopping cart is a session-scoped component that:

- Contains a reference to the current order and last order in this session, if any
- Is also used to configure the persistence of incomplete orders

/atg/commerce/ShoppingCart



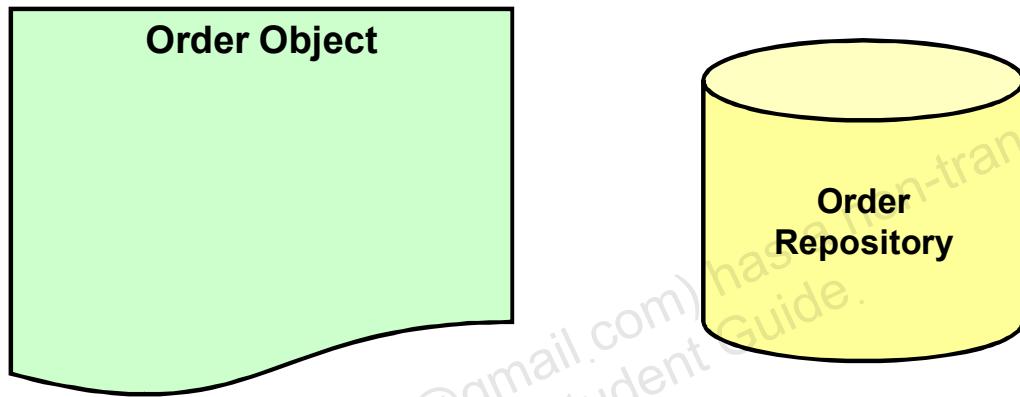
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

# Order Object Versus Order Repository Item

The Order data is stored in two places:

- The Order Java object in the server memory
- The Order repository item in the database



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The application interacts with the Order object in memory. The Order is persisted to the repository item when certain events occur (for example, the user logs in, items are added to an order, the order is submitted).

## Order Life Cycle

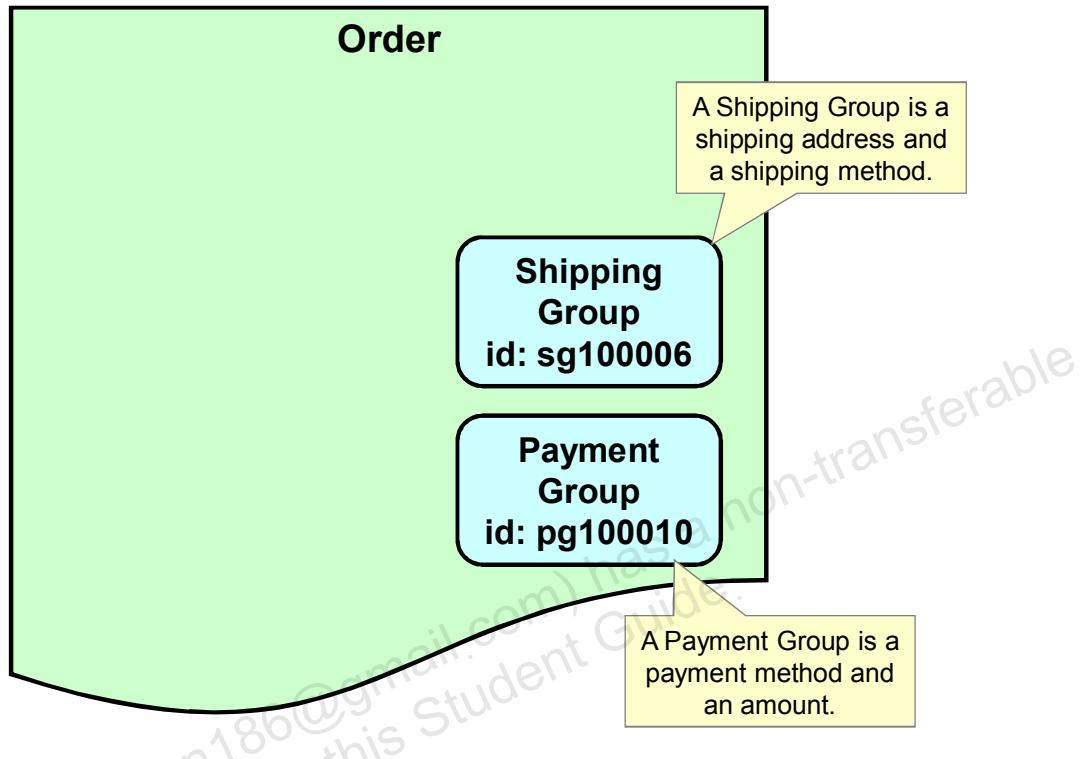
- The following slides illustrate the default life cycle of the Order object.
- Notes:
  - Many of these behaviors can be configured through the OrderTools component.
  - Not all sites take advantage of all order features, such as splitting shipping or saving multiple incomplete orders.

 ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full path to OrderTools is /atg/commerce/order/OrderTools.

# The Session Starts



ORACLE

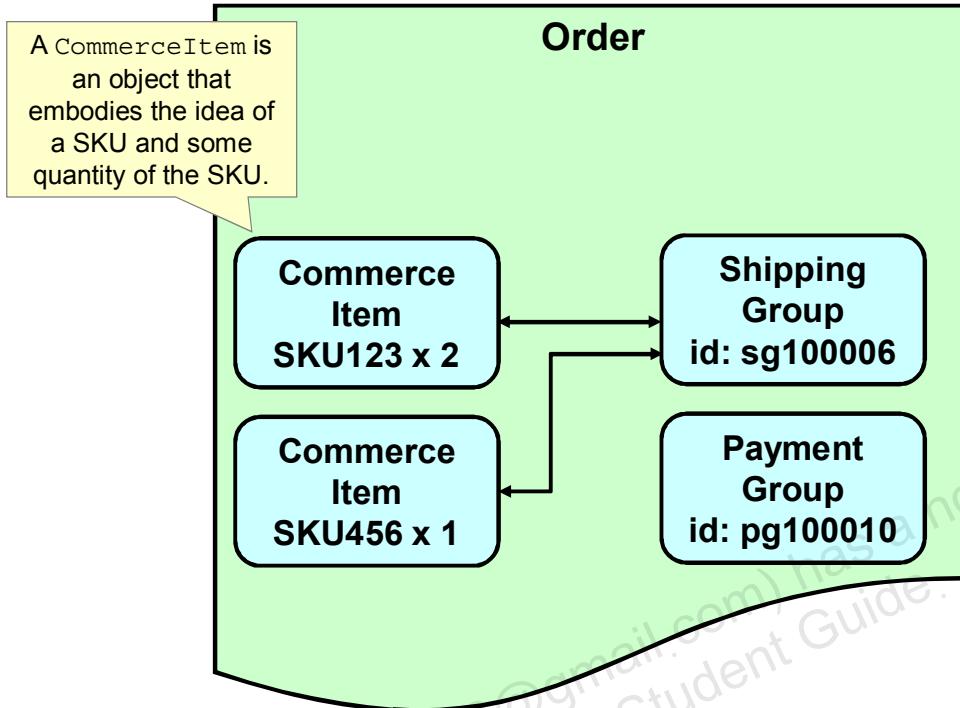
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

When a Commerce session starts (when a user opens the first page of a Commerce site), an Order object is created with links to one payment group and one shipping group. The types of the payment and shipping groups can be configured in the `OrderTools` component, in the `defaultShippingGroupType` and `defaultPaymentGroupType` properties. By default, they are `HardgoodShippingGroup` and `CreditCard`. A shipping group holds a shipping address and a shipping method (such as Next Day Air). A payment group holds a payment method (such as a credit card) and an amount.

A new Order will also be created when a user changes from one site to another, if the sites do not share the shopping cart.

**Note:** The ID numbers shown in the slides are examples for the purposes of illustration. The IDs of the actual objects may not match these numbers.

## SKUs Added to the Cart

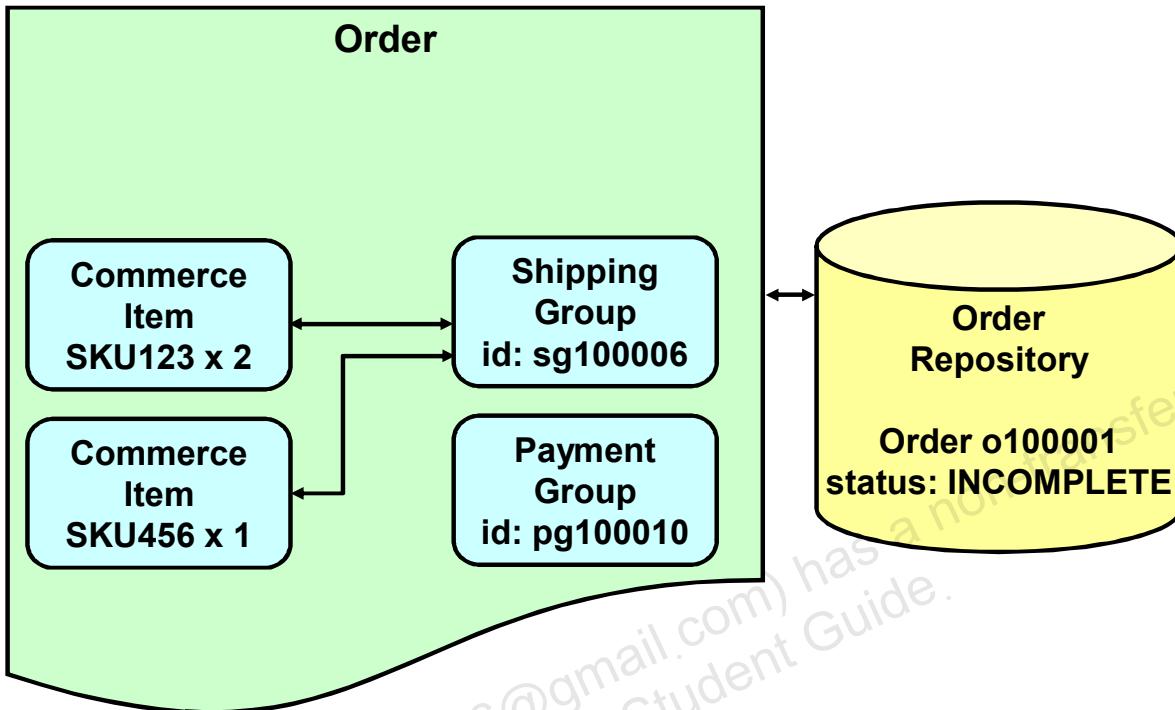


ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

As the shopper adds SKUs to the order, `CommerceItem` objects are created. The objects are linked to the default shipping group, but not to the default payment group.

## (Optional) The User Logs In

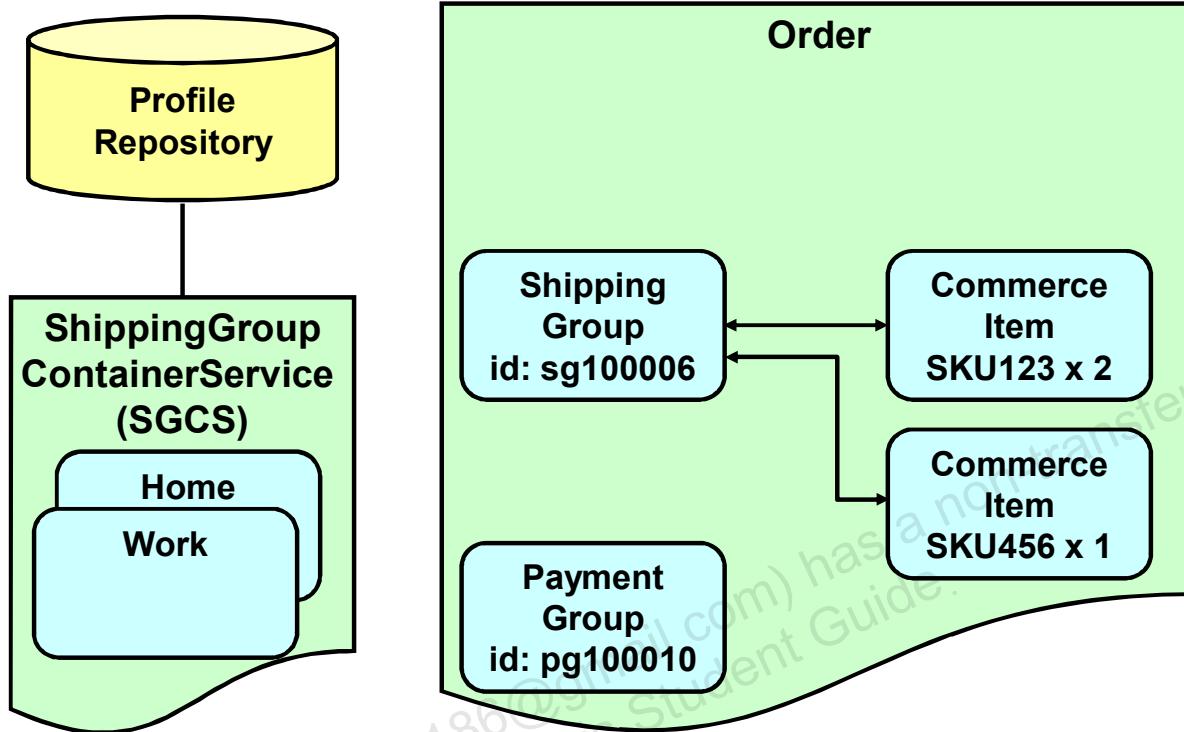


ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

By default, orders for anonymous users are not persisted in Order Repository until they are submitted. Orders for registered users are, however, saved at every stage, such as when the user logs in. These behaviors can be configured in the ShoppingCart component's `persistOrders` and `persistOrdersForAnonymousUsers` properties.

## The User Opens the Shipping Page

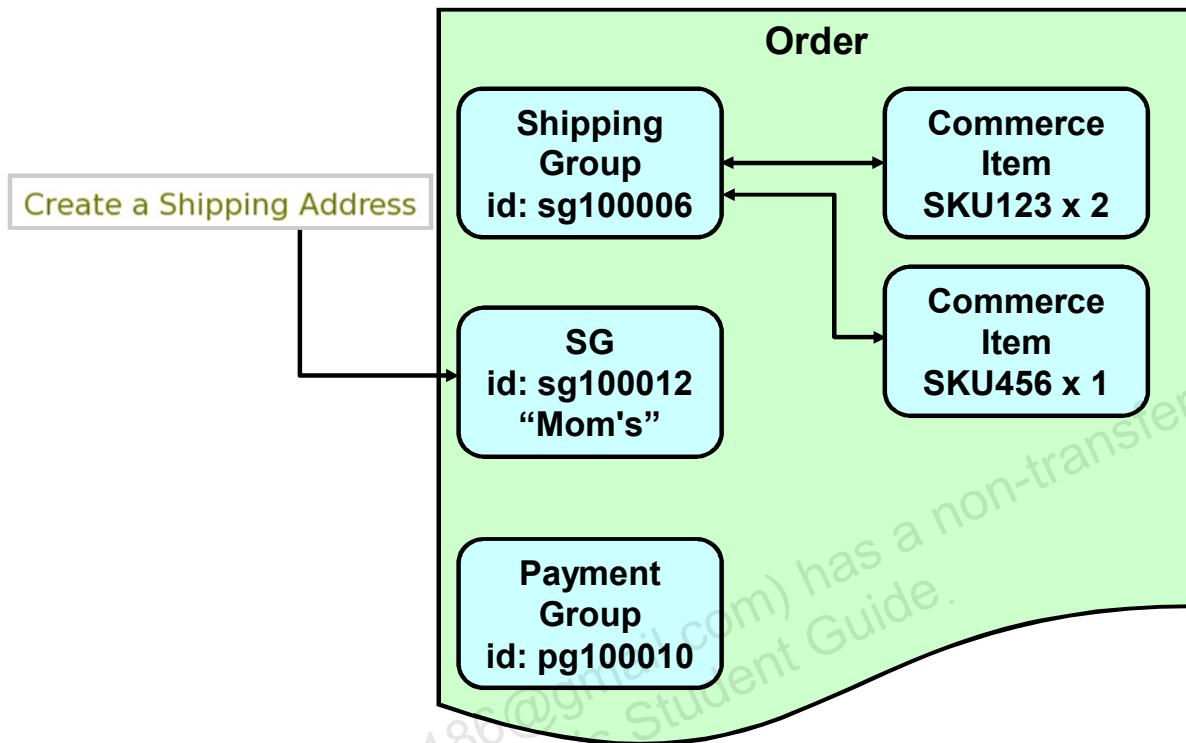


ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

If users are logged in, when they open the shipping page, any saved addresses they have are loaded into a component called  
`/atg/commerce/order/purchase/ShippingGroupContainerService (SGCS).`

## (Optional) The User Adds a New Address

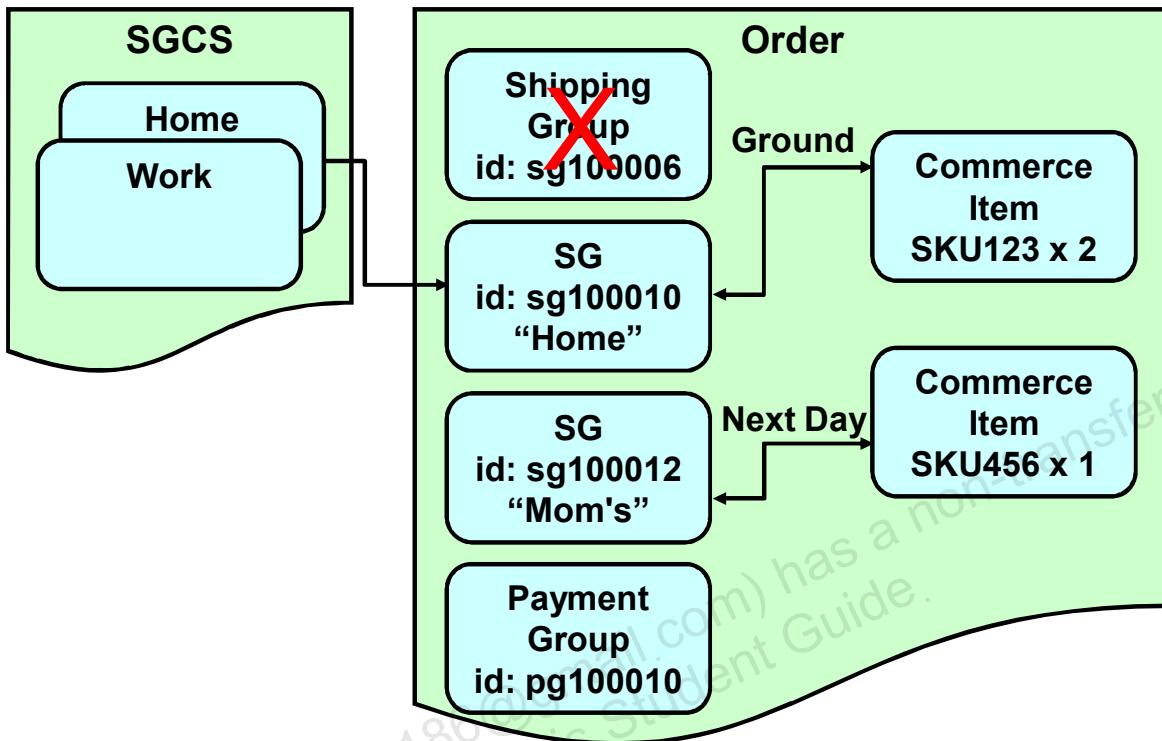


ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Users can also add new shipping groups by entering addresses in forms provided on the store pages. Registered and anonymous users can both add shipping groups this way. Anonymous users, who have no saved addresses, must add at least one shipping address. When the user creates a new address ("Mom's" in the example in the slide), a new shipping group is created. The user can now add SKUs to the new shipping group.

## The User Allocates Shipping

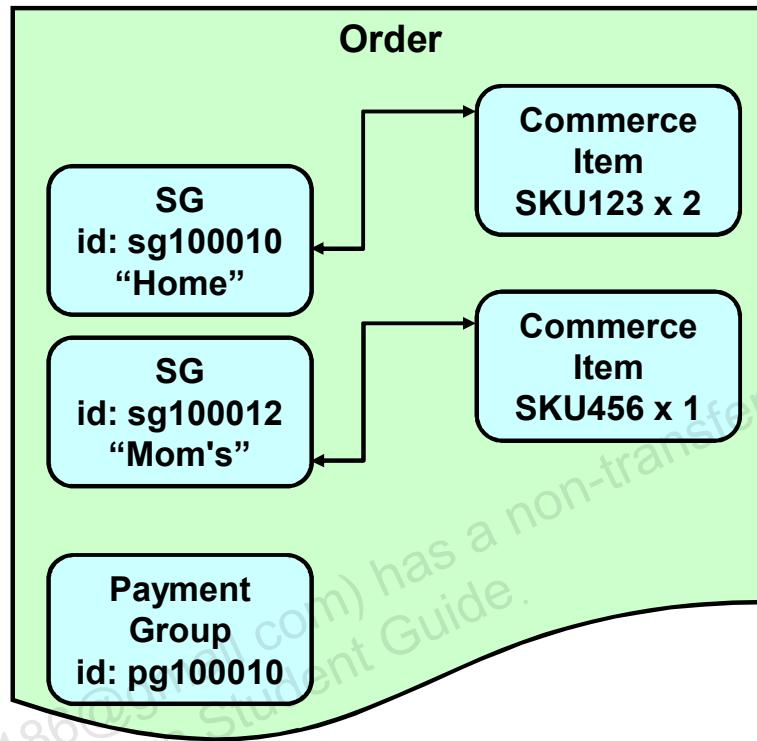


ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Users can then set which commerce items are to be shipped to which addresses, and select a shipping method for each shipping group. If they have ordered more than one of a particular item, they may choose to ship some of that item to one address and some to another. The links between `CommerceItems` and shipping groups change.

## The User Opens the Billing Page

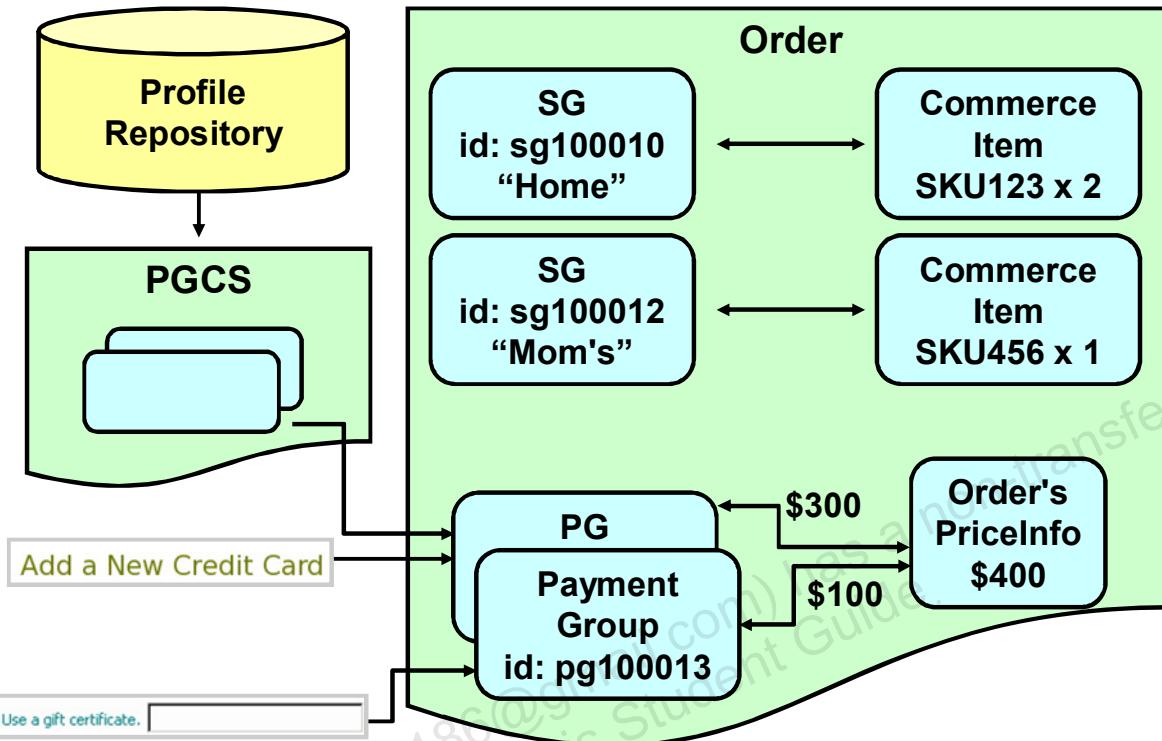


ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

When the user moves on to the billing page, the default shipping group is deleted from the order.

## User Allocates Billing



**ORACLE**

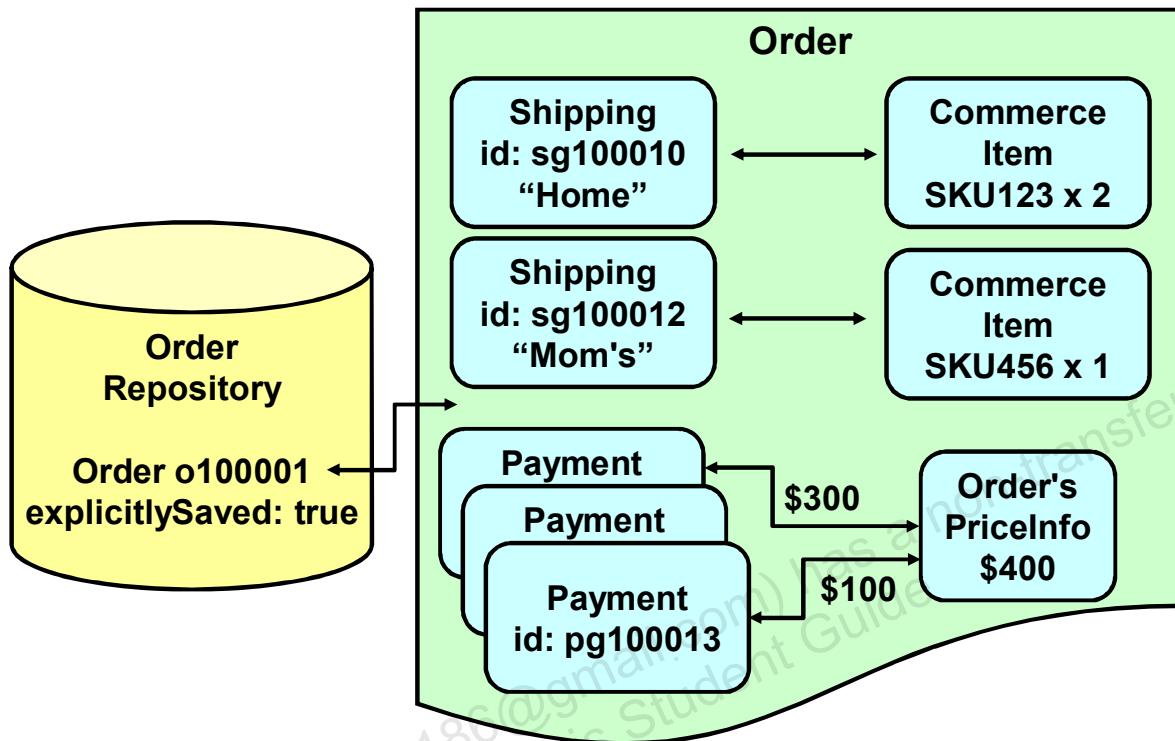
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

A similar process is used to create and allocate the order among payment groups. If there are any saved credit cards or store credits (or other custom payment types) in a registered user's profile, they will be loaded into the PaymentGroupContainerService (PGCS in the slide) for possible use in this order. Users can also add payment groups as needed, which may be credit cards, gift certificates, store credits, or purchase orders (by default).

Again, similar to allocating shipping, users may be able to select how much of an order should be paid for by each payment group. Depending on how the application is designed, users may split payment based on the total order amount (for example, an order with a total of \$400 for all items, tax, and shipping is paid for by a \$100 gift certificate and a \$300 credit card charge) or based on the individual line items (for example, the tax, shipping, and cost of SKU123 is paid for by one purchase order, and the total cost of SKU456 is paid for by another).

The ability to manually adjust the amount covered by each payment group is not currently exposed in CRS.

## (Optional) The User Explicitly Saves the Order



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

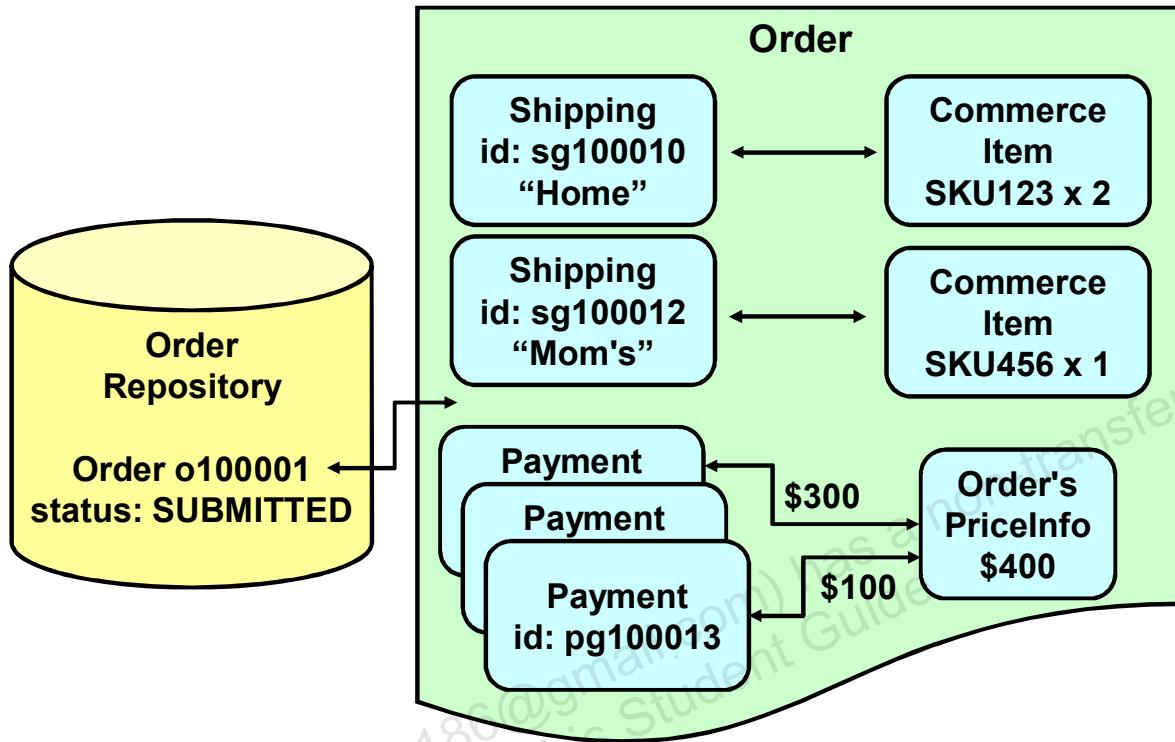
If supported by the application, registered users can choose to save an incomplete order at any time; all the information given so far will be saved. The order's `explicitlySaved` property is set to `true`. This property setting prevents a saved order from being marked as abandoned after the passage of a configured amount of time.

The saved order is available through `ShoppingCart . saved`, and the current order becomes empty. Users can have as many saved orders as they want. The application must provide pages that allow the user to view and retrieve saved orders for later submission.

Remember that the current order is automatically saved for registered users. The saved-order feature allows them to save multiple incomplete orders.

This feature is not currently implemented in CRS.

## The User Submits the Order

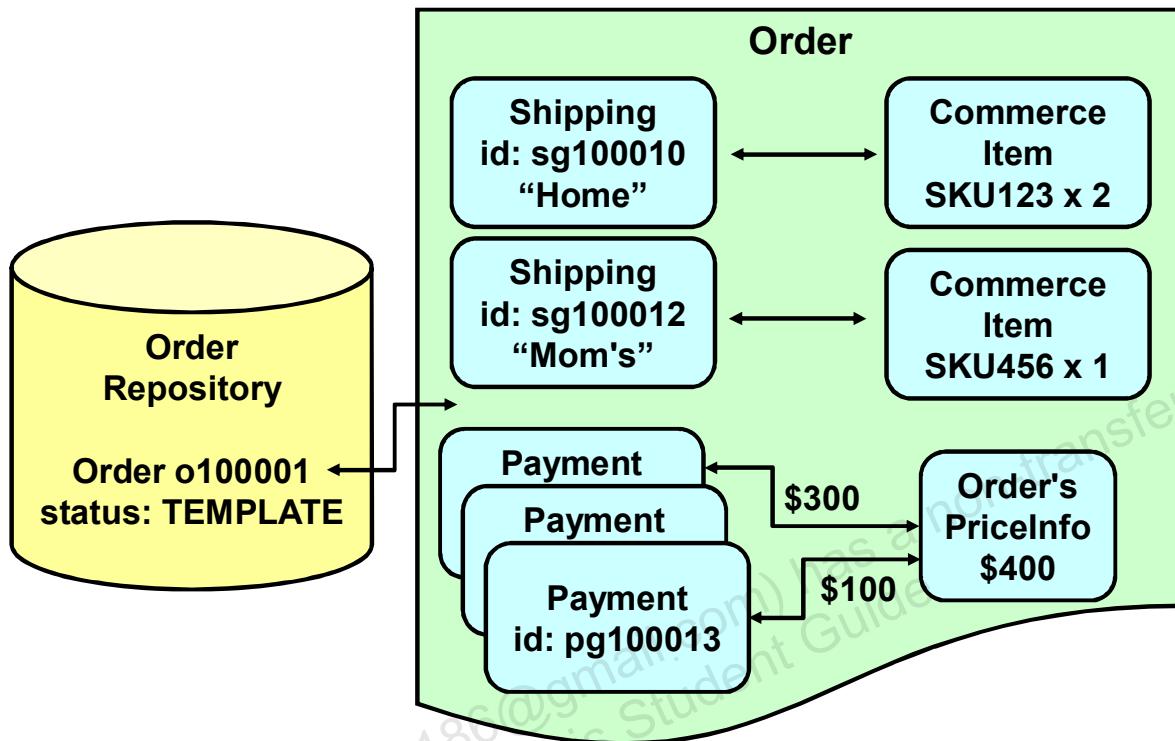


ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

By default, orders from anonymous users are not persisted in the Order Repository until they are submitted. Once an order is submitted, its status changes from INCOMPLETE to SUBMITTED.

## (Optional) The User Creates a Scheduled Order



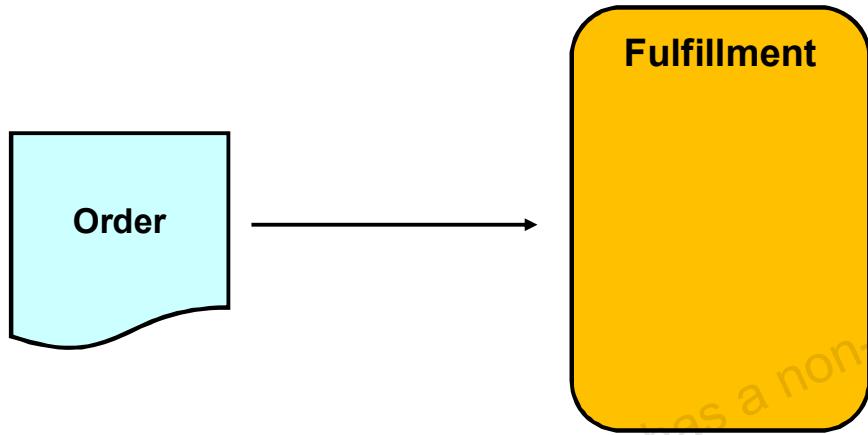
ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

If desired, stores can allow users to create scheduled orders. Scheduled orders allow users to set up a recurring order (for example, 100 widgets on the first of every month) or delay the submission of a single order (for example, submit a single order for 100 widgets on May 1). The completed order is saved as a “template” order, which is used to create the actual later orders as needed. A scheduled order must be complete, including all shipping and billing information.

This functionality is not currently implemented in CRS.

## The Order Is Sent to Fulfillment

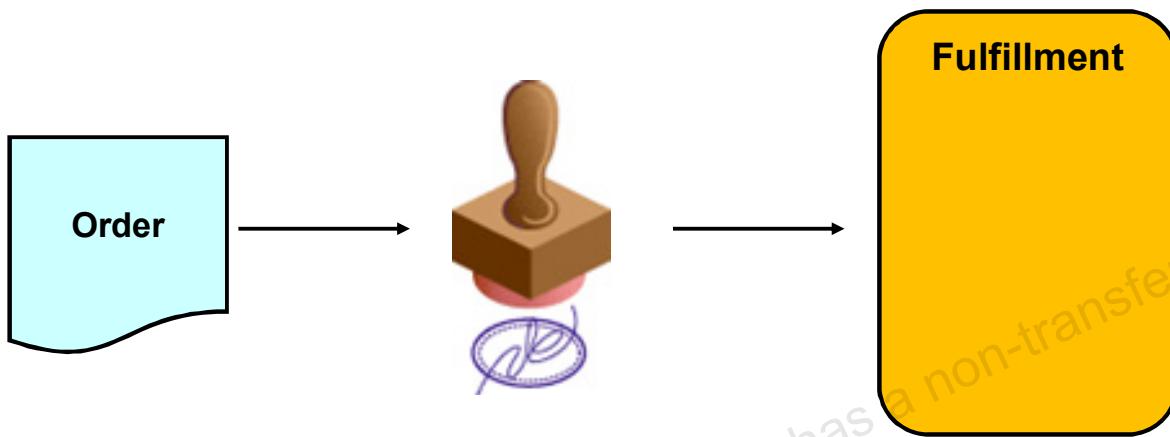


ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Fulfillment is handled by a separate module that may or may not be running on the same servers as the store. The Fulfillment system will be covered in more detail in the lesson titled “Inventory and Fulfillment.”

## (Optional) The Order Is Held for Approval



ORACLE

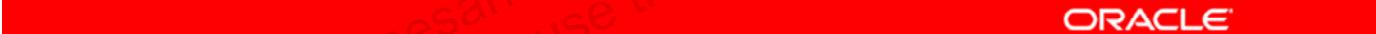
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You can choose to implement the order approval system. When this feature is used, some orders are held for approval before they are sent to Fulfillment. Depending on how the store implements approvals, permission for the order to proceed may come from external users (for example, in a B2B situation, the shopper's manager) or from internal users (for example, the fraud prevention department). Stores define the criteria used to hold orders. For instance, an order may be held if it is over a certain amount, or if the countries of the billing and shipping addresses do not match.

Implementing order approvals may require some extensions to the profile. For example, stores may want to be able to mark some users as exempt from the approval process. If order approvals are determined by external users, a profile property is needed to list which users are designated approvers for a particular customer, which may be inherited from a custom property on the customer's designated organization.

## Design Decision: Order Process

- How will your organization configure the order life cycle?
  - Persist incomplete orders for registered users? For anonymous users?
  - Allow saved incomplete orders?
  - Allow scheduled orders?
  - Allow splitting shipping or payment?
- Will some orders be held for approval? What criteria will be used?
- Which order management system are you integrating with, and at what point do you send the order detail to it?



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Quiz

Which of the following statements best describes orders and the ShoppingCart component in ATG Commerce?

- a. The Order object and the ShoppingCart component are the same object, referred to by different names.
- b. The Order object and the Order repository item are both persisted in the database.
- c. The ShoppingCart component contains references to the current and last order in the user session.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

**Answer: c**

# Road Map

- Orders
- Shipping and payment methods



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Shipping Methods

- Each possible shipping method (for example, ground or next day) is represented by a component based on a shipping calculator class.
- Out-of-the-box classes provide the ability to calculate shipping based on:
  - A fixed price (for example, ground shipping is \$10)
  - An attribute of the shipping group (for example, 0–10 kilograms = \$10, over 10 kilograms = \$20)
- Custom shipping methods can be added.
  - For example, calculate the shipping price by using the shipping company's web service.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

You will learn more about shipping and shipping methods in the lessons titled “Shipping and Shipping Groups” and “Pipelines and Shipping Methods.”

## Filtering Shipping Methods

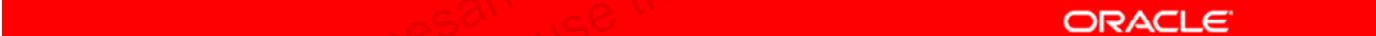
- In some cases, not all shipping methods should be available for all addresses.
  - For example, courier shipping is not allowed for PO boxes.
- Each shipping method can be extended to use information from the shipping group, order, or profile to determine whether this method is allowed for this shipping group.

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Design Decisions: Shipping Methods

- Which shipping methods will your organization require?
- Which shipping calculators will you use?
- Will any custom shipping calculators or custom SKU properties be needed?
- Will shipping methods be filtered? If so, how?



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Payment Methods

- Four payment methods are available out of the box:
  - CreditCard
  - StoreCredit
  - GiftCertificate
  - InvoiceRequest
    - For use with purchase orders
- Custom payment methods can be added.
  - For example, Bill Me Later, PayPal

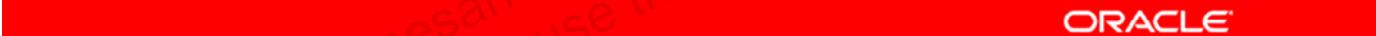


ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Design Decisions: Payment Methods

- Which out-of-the-box payment methods will your store use?
- Will any custom payment methods be needed?

A solid red horizontal bar located at the bottom of the slide.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Summary

In this lesson, you should have learned how to describe:

- The difference between the order object and the shopping cart
- The order life cycle
- Shipping and payment methods

## For More Information

### *ATG Commerce Programming Guide*

- Working with Purchase Process Objects
- Managing the Order Approval Process

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

# Extending Orders

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to:

- Describe the design pattern of components used to work with orders
- Extend order-related processes and order persistence

## Road Map

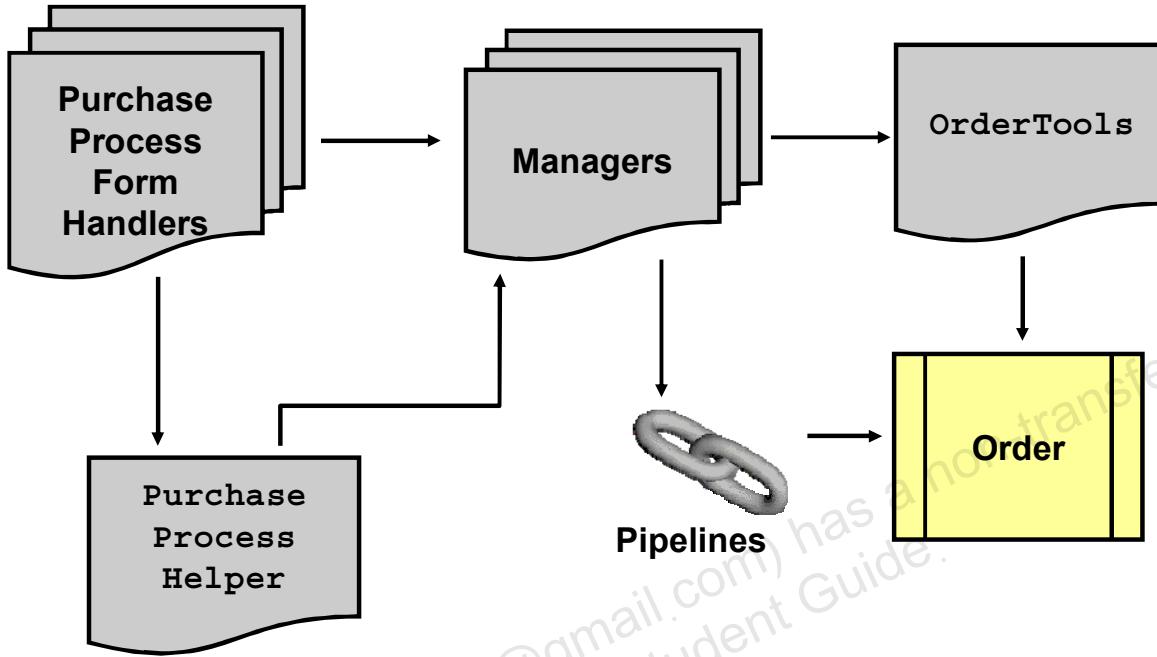
- Order-related classes and components
- Pipelines
- Modifying the Order programmatically
- Extending the Order and Order persistence



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Order-Related Components



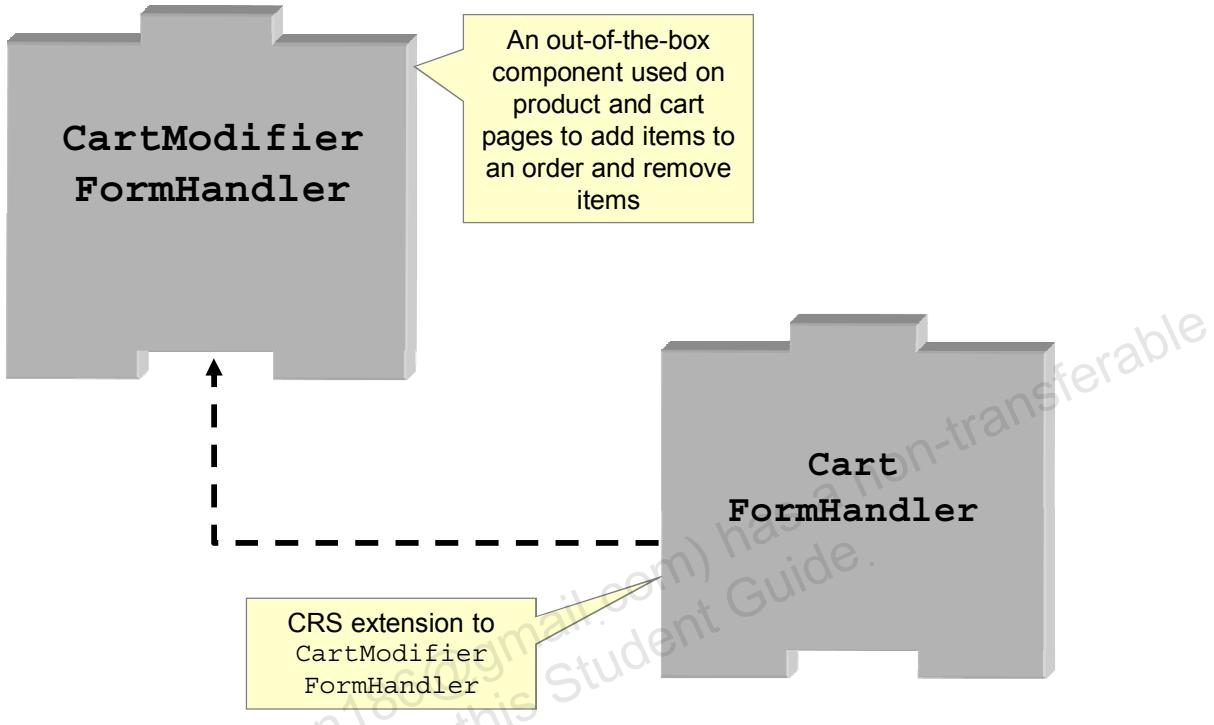
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Shoppers interact with order through the various purchase process form handlers. These form handlers use a variety of Manager components, such as the OrderManager and the CommerceItemManager, to make order changes. These components contain all the business logic governing these changes (for example, whether a given item is allowed to be added to the current order). The Manager components then call on methods in the OrderTools component, which directly change the Order object.

Some of the common actions that form handlers need to perform are centralized in the PurchaseProcessHelper component. Also, the Manager components may call pipelines to perform some functions, such as persisting orders to the Order Repository.

## Purchase Process Form Handlers



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

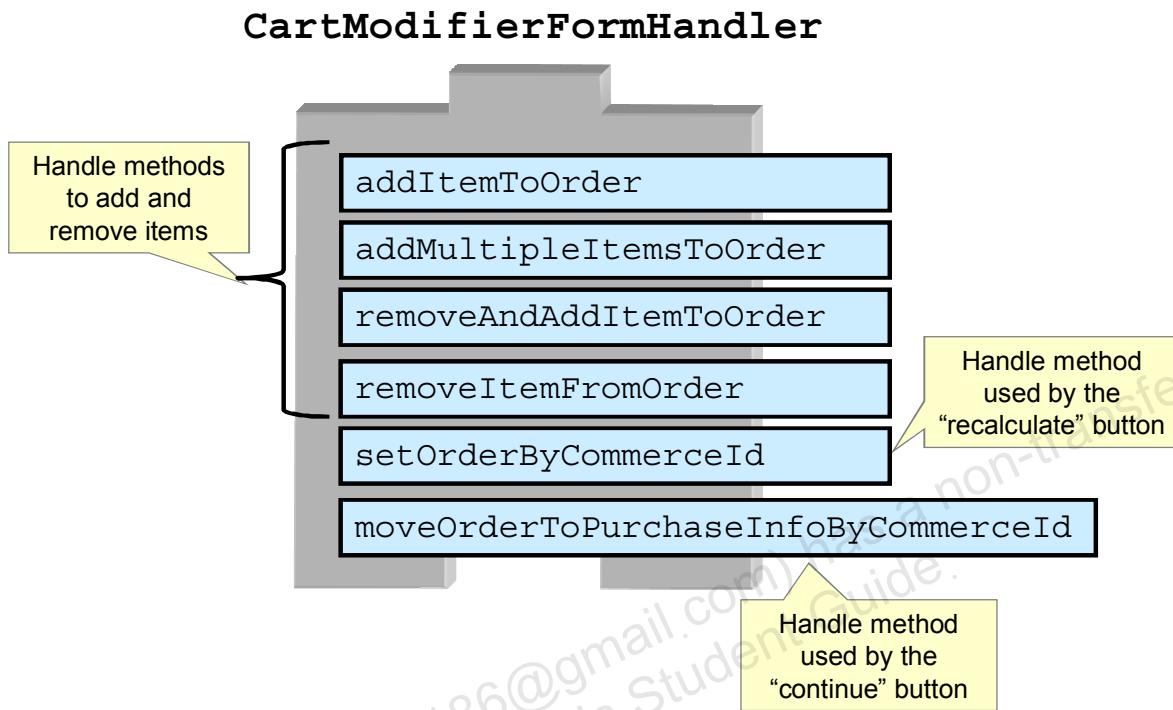
The full path for `CartModifierFormHandler` is `/atg/commerce/order/purchase/CartModifierFormHandler`. It is request-scoped. `CartModifierFormHandler` is used on product pages and the cart page to:

- Add items to the order
- Remove items from the order
- Change quantities of items in the order
- Move the order forward in the purchase process to start collecting payment and shipping information

The component is an example of a purchase process form handler. Its class, `atg.commerce.order.purchase.CartModifierFormHandler`, extends the `atg.commerce.order.purchase.PurchaseProcessFormHandler` class.

The CRS extension is a component called `/atg/store/order/purchase/StoreCartFormHandler`, whose class, `atg.projects.store.order.purchase.StoreCartFormHandler`, extends `CartModifierFormHandler`. It simplifies the handle methods and adds the functionality to add a gift list item to the cart.

## CartModifierFormHandler Handle Methods



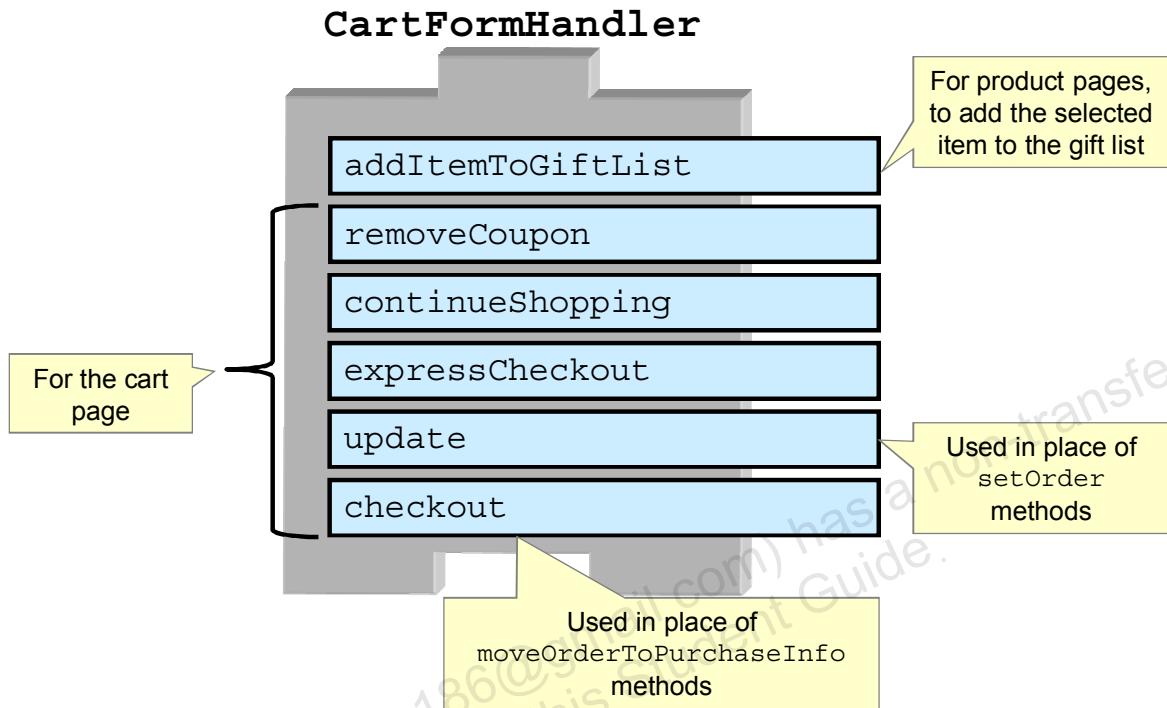
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Some of the handlers are used on product pages to add items to the order. Cart pages typically include a form that allows shoppers to change item quantities or remove items, and then continue to the rest of the purchase process.

The class has many properties and handle methods. For more information, see the *API Platform Reference*.

## CartFormHandler Extensions (CRS)



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full path to component is `/atg/store/order/purchase/CartFormHandler`. It is based on the `atg.projects.store.order.purchase.StoreCartFormHandler` class. The slide shows the handle methods that are added by the `StoreCartFormHandler` class:

- `addItemToGiftlist`
- `removeCoupon`
- `continueShopping`: Makes changes to the order and redirects to the specified “continue shopping” URL
- `expressCheckout`: Moves the order forward to express checkout
- `update`
  - Used in place of `setOrder` methods
  - Calls `super.setOrderByCommerceId`
- `checkout`
  - Used in place of `moveOrderToPurchaseInfo` methods
  - Calls `super.moveOrderToPurchaseInfoByCommerceId`

## Extending Purchase Process Form Handlers

- As with all out-of-the-box form handlers, purchase process form handlers include empty “pre” and “post” methods for each handle method.
- **Best practice:** Consider extending `CommerceItemManager` or `OrderManager`, if the extension includes business logic that should also apply when changes are made either
  - Programmatically or
  - From another application, such as CSC



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ATG Commerce Service Center (CSC) is an agent-facing application that lets agents view and change customer orders and profile data, issue returns or exchanges, and perform other activities related to customer service. An important note to developers is that, in general, CSC uses a different set of form handler components, but these CSC form handlers make calls to the same “Manager” components used by the base ATG Commerce framework. This means that customizations made at the Form Handler level will likely NOT be picked up by CSC whereas customizations made at the Manager level will.

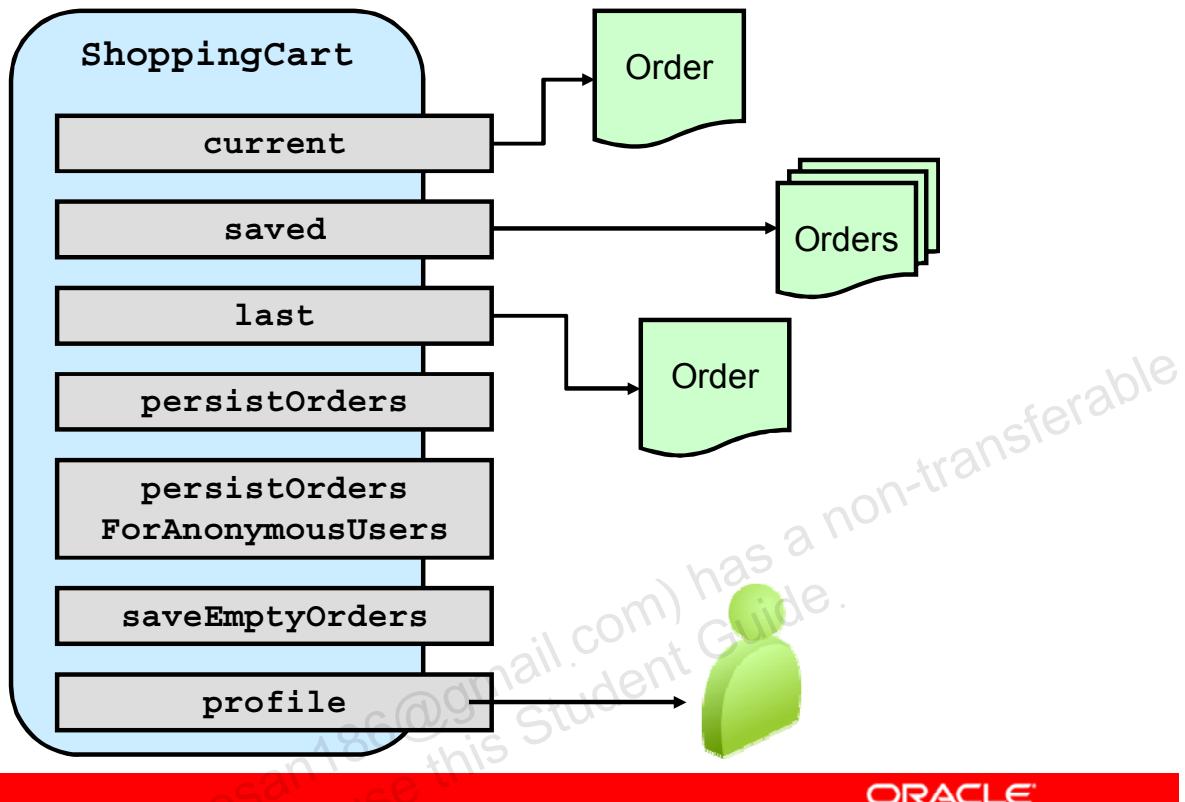
## Extending CRS Form Handlers

- If using CRS, always check when extending an out-of-the-box class whether CRS has already extended it.
  - If so, base your extension on the CRS class.
- Not all “pre” and “post” methods in CRS form handler extensions are empty, so to be safe, call the superclass method.

**ORACLE®**

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Shopping Cart



ORACLE

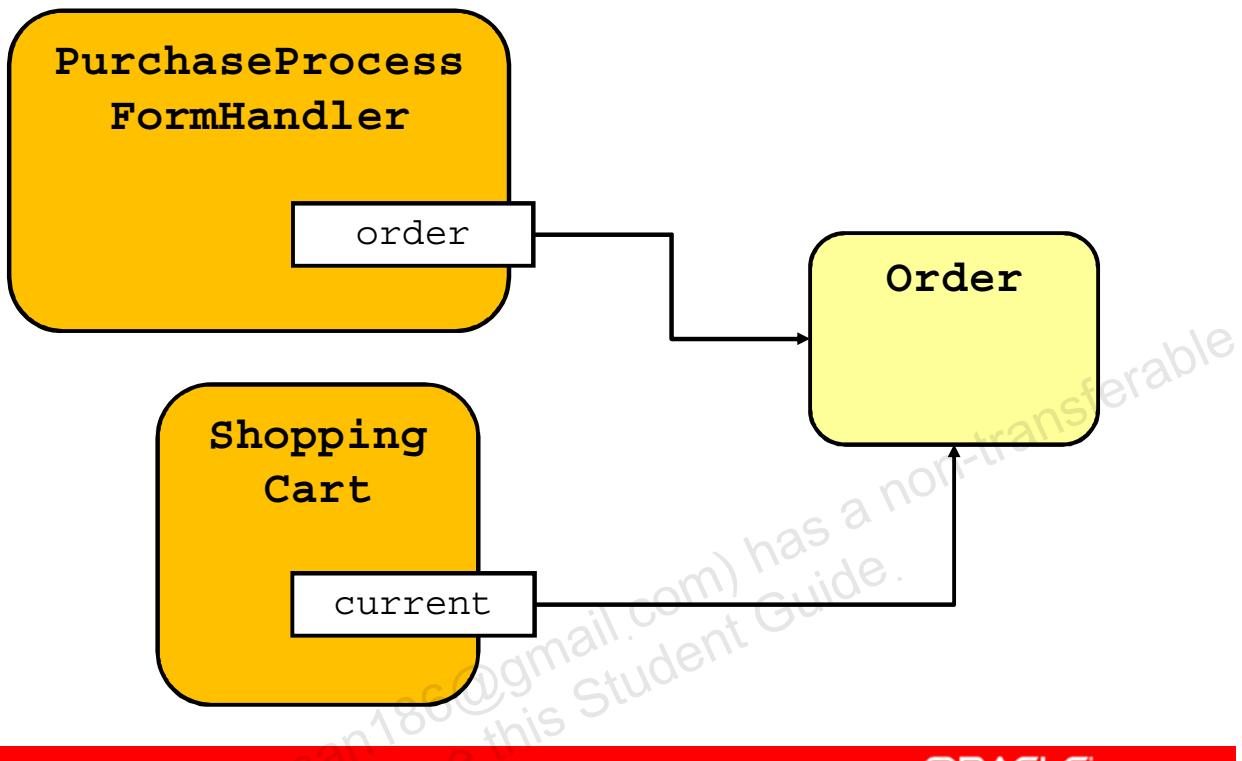
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The /atg/commerce/ShoppingCart component is based on the atg.commerce.order.OrderHolder class, and is session-scoped. Properties include:

- **current**: The current Order. If the current order is null, a new order is automatically created when this property is retrieved.
- **saved**: A collection of Order objects that are the user's saved list of shopping carts
- **last**: A reference to the previous order placed during this session, if any
- **persistOrders**, **persistOrdersForAnonymousUsers**, and **saveEmptyOrders**: Boolean properties that control the types of orders that get persisted
- **profile**: A reference to the Profile object for the user

**Note:** Persisting orders for anonymous users is turned off by default, because the size of your order repository can grow quite quickly, if you keep orders from anonymous users. If you choose this option, you need to have a plan for regularly clearing out old anonymous orders.

## Ways to Access the Current Order



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Both `PurchaseProcessFormHandler.order` and `ShoppingCart.current` point to the current active order.

**Note:** `PurchaseProcessFormHandler` is the class name (in the `atg.commerce.order.purchase` package). There are several default form handlers based on it. You will learn more about those in the lesson titled “Other Purchase Process Form Handlers.”

## Manager Components

- The purchase process form handlers use several manager components:
  - ClaimableManager
  - CommerceItemManager
  - OrderManager
  - PaymentGroupManager
  - ShippingGroupManager
- Manager components contain all the business logic regarding changes to orders.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

PaymentGroupManager and ShippingGroupManager will be covered in more detail in later lessons.

## CommerceItemManager

CommerceItemManager contains methods for working with commerce items, such as:

- Create a commerce item
- Add the item to the order
- Remove the item from the order



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Commerce Items

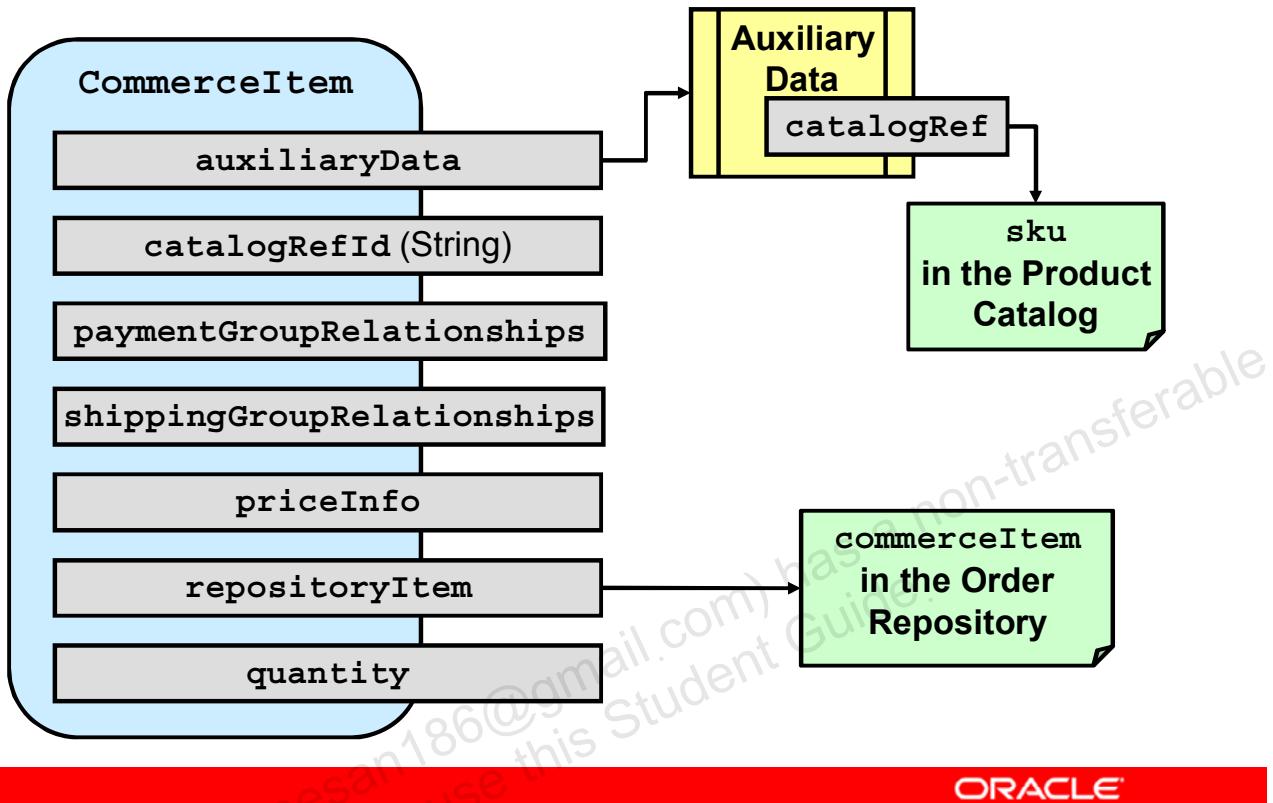
- When SKUs are added to the order, they become `CommerceItem` objects.
- `CommerceItem` objects represent each line item in the order.
  - For example: 2 x SKU123
- `CommerceItem` objects are persisted in the Order Repository as `commerceItem` repository items.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## CommerceItem Object



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The slide illustrates selected properties of the `CommerceItem` class. In particular, it shows the properties that link the `CommerceItem` object to the `sku` repository item.

The `catalogRefId` property holds the `sku` item's ID. In general, `catalogRefId` is a synonym for `skuId`.

Every `CommerceItem` must be associated with a payment group and a shipping group. The relationships are stored in the `CommerceItem` as one-ended relationships. There will be one `CommerceItem` per shipping group or payment group, unless the customer splits the shipping or payment across multiple groups. You will learn more about payment group and shipping group relationships and `priceInfos` later in the course.

**Note:** The `CommerceItem.repositoryItem` property does *not* represent the SKU. It provides a pointer to the repository item used to persist `CommerceItem` in the Order Repository. This allows you to access the `sku` repository item, and not just the SKU's ID, through `CommerceItem`.

## OrderManager

- OrderManager contains methods for working with orders, such as:
  - Create the order
  - Save and load the persisted order
  - Allocate the order amount to a payment group
- The CRS extension is StoreOrderManager.
  - Adds gift list functionality and the adjustInventoryOnCheckout property



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full path to the OrderManager component is  
/atg/commerce/order/OrderManager.

## Helper Components

- The purchase process form handlers use the `PurchaseProcessHelper` for many operations.
- `PurchaseProcessHelper` centralizes functionality needed by more than one purchase process form handler.



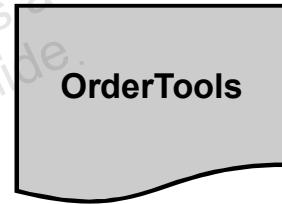
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full path to the `PurchaseProcessHelper` component is  
`/atg/commerce/order/purchase/PurchaseProcessHelper`.

## OrderTools Component

- The OrderTools component is used by Manager components.
- It contains raw operations only, no business logic.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full path to the OrderTools component is /atg/commerce/order/OrderTools.

# Road Map

- Order-related classes and components
- **Pipelines**
- Modifying the Order programmatically
- Extending the Order and Order persistence

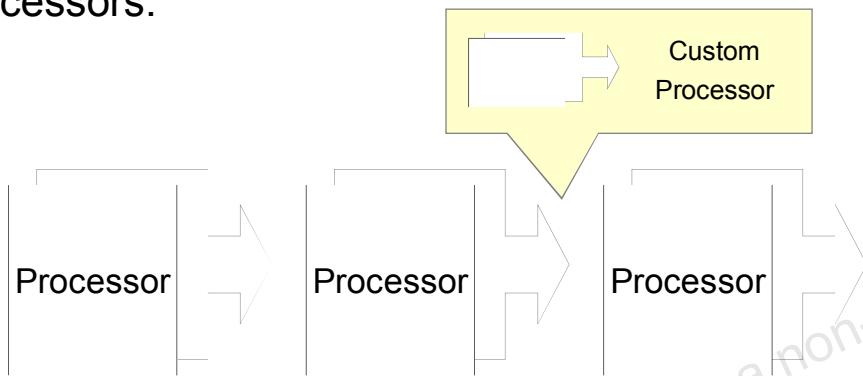


ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Pipeline Basics

- ATG Commerce has several pipeline chains, typically called by the business layer classes, made up of processors.



- Pipeline Manager components, configured with an XML file, run processor chains.



**ORACLE**

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

You may be familiar with the concept of pipelines in ATG through the servlet pipeline, which is a request-handling pipeline. Whenever a HTTP request comes in the request is passed through a series of servlets. A response object is returned. In the servlet pipeline, each servlet knows what follows.

The ATG Commerce pipeline architecture is similar, but uses *processors* instead of servlets. ATG Commerce has many pipeline chains. Each chain is made up of processors, each performing a specific function. A processor does not know what processor follows. This is configured through XML. You can easily customize a pipeline chain by inserting your own processors.

The PipelineManager is responsible for the creation, registration, and removal of pipeline chains. It uses the Java Transaction API. Each processor in the chain can mark the chain to be rolled back. If none mark the transaction, then it is committed when the chain finishes execution.

PipelineManager calls its `runProcess()` method with the chain's ID to execute a chain.

Processors return status codes, which may determine what processor to execute next.

Chain IDs are found in pipeline XML files. Examples: `processOrder`, `validateForCheckout`, and `validateShippingInfo`.

## Commonly Extended Pipelines

- ATG Commerce defines several pipelines out of the box.
- Some of the commonly extended pipelines are:
  - updateOrder
  - loadOrder
  - processOrder



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full list of pipelines, and the details on the processors that make them up, can be found in “Processor Chains and the Pipeline Manager” in the *Commerce Programming Guide*.

Commonly extended pipelines include the following:

- **updateOrder:** Saves the order to the repository
- **loadOrder:** Loads the order from the repository
- **processOrder:** Is triggered when the user submits the order

## Pipeline Manager Components

- Most pipelines in Commerce are configured in
  - /atg/commerce/PipelineManager
  - /atg/commerce/payment/PaymentPipelineManager
- Use the Component Browser of the Dynamo Administration UI to see detail on the live pipelines.
- Other pipeline manager components exist for other modules (for example, the Fulfillment module).



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The Fulfillment module's pipeline manager component is  
/atg/commerce/fulfillment/FullfillmentPipelineManager.

## Commerce Pipeline Processors

- Commerce pipeline processor classes are found in the following packages:
  - atg.commerce.order.processor
  - atg.commerce.payment.processor
- Components are found in the following name spaces:
  - /atg/commerce/order/processor
  - /atg/commerce/payment/processor

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Example: ProcessOrder Chain

After a customer adds all necessary information to an order:

1. The customer submits the order for checkout.
2. OrderManager.processOrder() is invoked.
  - PipelineManager calls the processOrder pipeline chain.
  - When the pipeline is finished, PipelineManager returns a PipelineResult object to OrderManager, which then returns it to whatever called processOrder().



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

processOrder is defined in the /atg/commerce/commercepipeline.xml definition file. You can view versions of the file in the config.jar file in the B2CCommerce module or in the <ATGdir>\B2BCommerce\src\config\atg\commerce\ directory.

The PipelineResult object has a Boolean property called errors to indicate whether any errors were returned by the pipeline links, as well as additional properties to hold the errors and error messages.

## ProcessOrder Chain Processors



This chain includes:

- Validating the order
- Checking for expired promotions
- Marking used promotions
- Persisting the Order
- Sending the Order to the fulfillment system



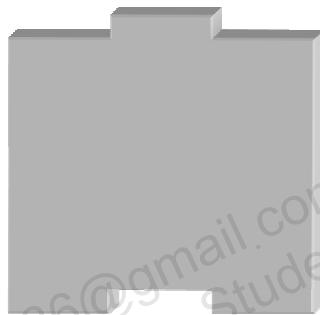
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Each processor in this chain is a component that lives in /atg/commerce/order/processor. Some key processor components include:

- ExecuteValidateForCheckoutChain simply executes a nested chain that is identified by its chainToRun property. In this case, the nested chain is validateForCheckout, which makes sure there is at least one CommerceItem, ShippingGroup, and PaymentGroup in the Order.
- CheckForExpiredPromotions checks whether any expired promotions were used during the pricing of any part of the order.
- MoveUsedPromotions moves any used promotions defined as being “use once” from the user’s activePromotions list to the usedPromotions list in their Profile.
- AddOrderToRepository makes sure the order is made persistent (stored in the repository), if the order is transient.
- SendFulfillmentMessage sends a message to the fulfillment engine when an order has been processed. The fulfillment engine may then begin fulfilling the Order.

## Customizing a Pipeline: Steps One and Two

- Pipeline definition files (XML) define pipeline chains.
- Steps to customize a pipeline chain
  1. Create a processor:
    - Implement the `atg.service.pipeline.PipelineProcessor` interface.
    - Implement the `runProcess()` method, returning a status code.
  2. Create a nucleus component for your processor.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

There are three steps to customizing a pipeline. The first two steps are shown in the slide: creating a new processor and creating a new processor component. The third step involves inserting the new processor into a pipeline by editing the pipeline's XML file. This is shown in the next slide.

The status code that is returned by the `runProcess()` method is used in the XML file to determine which processor is called next.

You might want to customize the `processOrder` pipeline, for example, by adding a processor that keeps a record of how often customers attempt to use promotions that have expired.

Source code is provided for most of the delivered pipeline processors at `<ATG-Dir>\DCS\src\Java\atg\commerce\order\processor`.

**Note:** In addition to using XML, you can also dynamically configure pipelines by using the `atg.service.pipeline` API.

## Customizing a Pipeline: Step Three

3. Configure your pipeline link in /atg/commerce/commercepipeline.xml in one of two ways:
  - a. Write the XML manually (as shown in the slide notes).
  - b. Use the ACC's graphical pipeline editor (pictured on the next slide).
    - **Note:** To use the ACC editor, the new pipeline processor must be registered in /atg/registry/pipelineRegistry.xml.

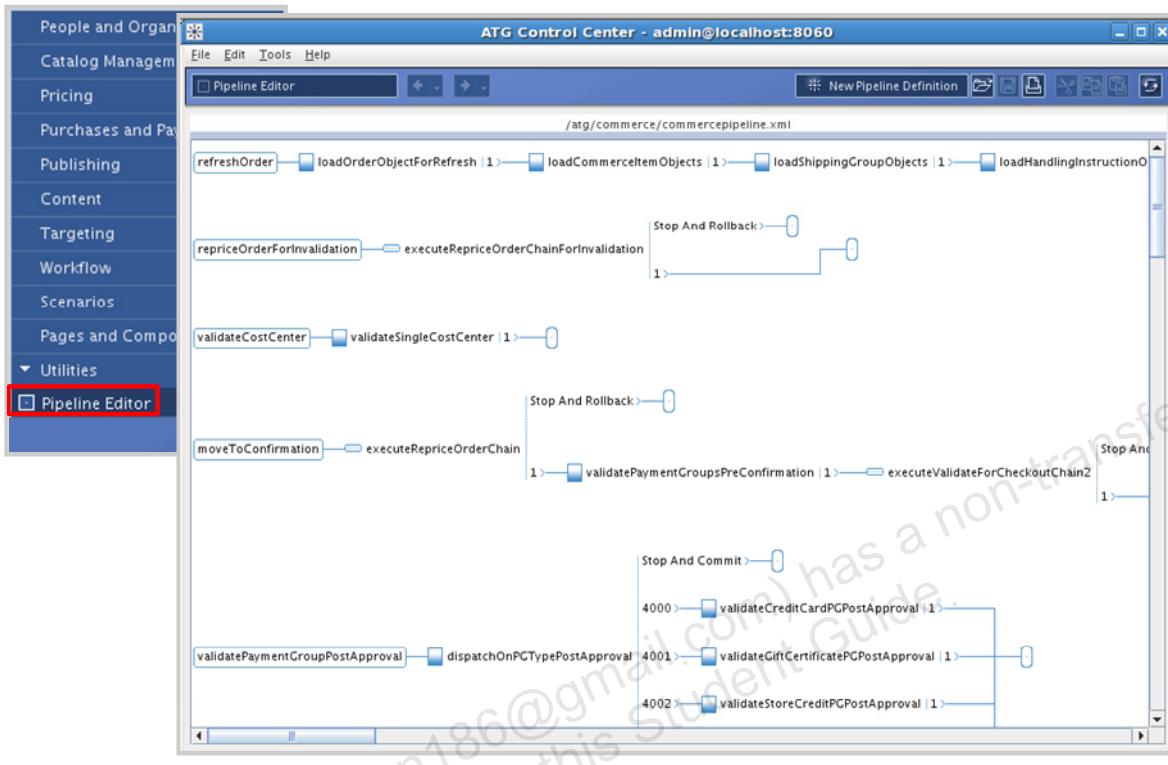


Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The Commerce Reference Store has made two extensions to the Commerce pipelines. An excerpt is shown in the following.

```
<pipelinemanager>
  <pipelinechain name="processOrder">
    <pipelinelink name="sendPromotionUsedMessage"
      transaction="TX_MANDATORY"
      xml-combine="replace">
      <processor
        jndi="/atg/commerce/order/processor/
          SendPromotionUsedMessage" />
      <transition returnvalue="1"
        link="sendFulfillmentMessage" />
    </pipelinelink>
  </pipelinechain>
</pipelinemanager>
```

# Editing Pipelines in the ACC



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

Pipelines may also be manipulated by using the ATG Control Center (ACC). To edit a pipeline in the ACC, select the Utilities task. Then, select the Pipeline Editor subtask. From the File menu, you may:

1. Create a New Pipeline
2. Open an existing Pipeline

To open an existing Pipeline, you select the pipeline manager that contains the Processor Chain Definition you want to edit.

# Quiz

Which of the following steps are valid when customizing a commerce pipeline? (Select all that apply.)

- a. Create a pipeline class that extends the `atg.commerce.pipeline.CommercePipeline` class to include your processor code.
- b. Create a Nucleus component for the custom processor.
- c. Register the new component in the `PipelineProcessor` component's `processor` property.
- d. Configure your pipeline link in `/atg/commerce/commercepipeline.xml`.
- e. Create a processor that implements the `atg.service.pipeline.PipelineProcessor` interface.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

**Answer: b, d, e**

The steps are not shown in the correct order in the quiz. The three steps to customize a commerce pipeline, in order, are:

1. Create a processor that implements the `atg.service.pipeline.PipelineProcessor` interface and implements the `runProcess()` method, returning a status code.
2. Create a Nucleus component for the custom processor, based on the class created in step one.
3. Configure your pipeline link (the component created in step two) in `/atg/commerce/commercepipeline.xml`.

## Road Map

- Order-related classes and components
- Pipelines
- **Modifying the Order programmatically**
- Extending the Order and Order persistence

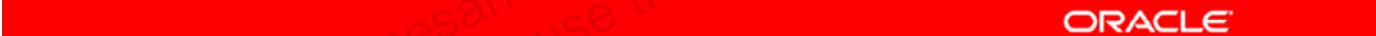


ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Modifying the Order Programmatically

- Stores sometimes need to write custom code to modify orders outside of the normal purchase process.
- This code *must* follow specific steps to avoid errors.
  - Possible errors include concurrent update exceptions, deadlocks, and corrupted orders.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Steps to Safely Modify an Order

The code that modifies the order object should:

1. Obtain a local lock on the profile ID
2. Begin the transaction
3. Synchronize on Order
4. Perform all modifications to the Order object
5. Call `OrderManager.updateOrder`
6. End the synchronization
7. End the transaction
8. Release the local lock on the profile ID



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Code Examples

- For examples of the code needed, see
  - PurchaseProcessFormHandler's beforeSet, afterSet, acquireTransactionLock, and releaseTransactionLock methods
  - Any handle method of any form handler that subclasses PurchaseProcessFormHandler
- More details are also given in the knowledgebase articles that are available in the My Oracle Support Community:
  - “Process for updating orders in ATG commerce code [ID 1362812.1]”
  - “Managing Transactions Involving the ATG Commerce Order Object [ID 1362761.1]”



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The source code for PurchaseProcessFormHandler is available in the DCS module, in DCS/src/Java/atg/commerce/order/purchase/PurchaseProcessFormHandler.java. The source code for some of the form handlers that subclass PurchaseProcessFormHandler are also available in this folder.

The knowledgebase articles are available through support.oracle.com. You can search for them by name or by entering “PurchaseProcessFormHandlers” in the search form. Oracle ATG Support has created several additional articles on this topic.

## Road Map

- Order-related classes and components
- Pipelines
- Modifying the Order programmatically
- Extending the Order and Order persistence



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Order Object Versus Order Repository Item

- The Order data is stored in two places:
  - Java objects in the server memory
  - Repository items in database
- The application interacts with Java objects.
- The Order is persisted to the repository (by default)
  - On update, for registered users
  - On submit, for anonymous users
- The structure of repository items is not identical to Java object properties.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Order-Related Java Objects

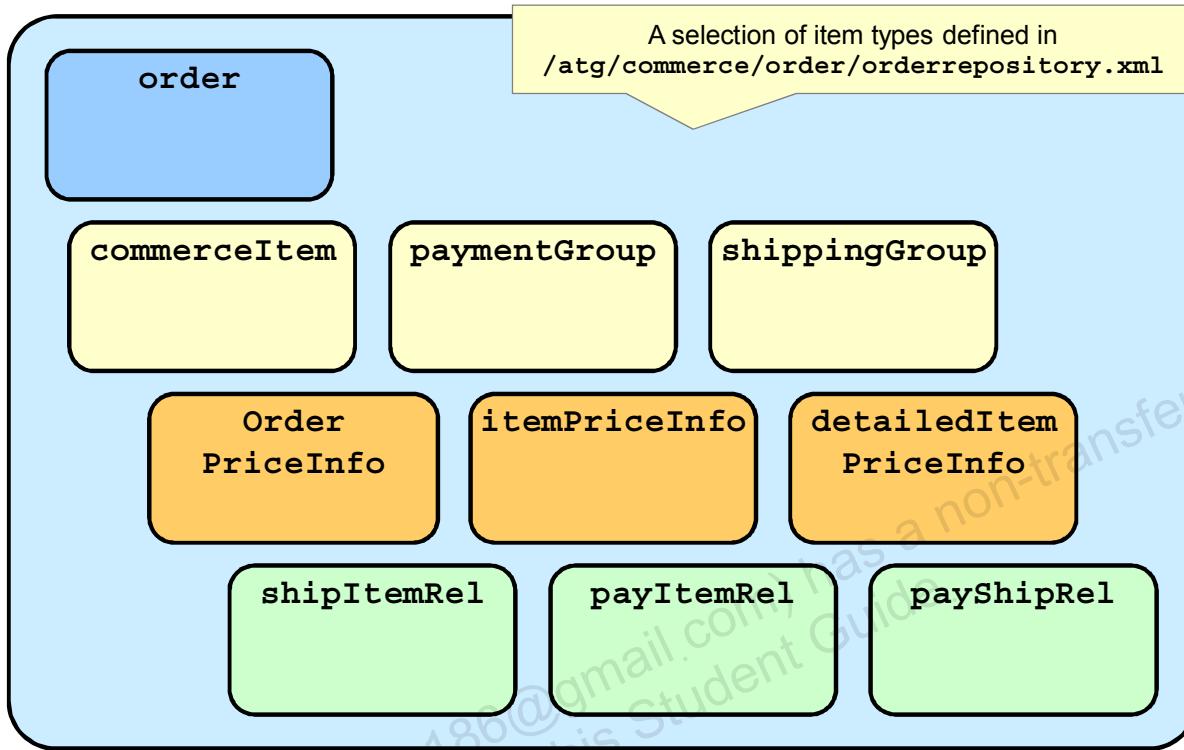
- Classes and interfaces that represent the Order are in the `atg.commerce.order` package.
- Each piece of an order has an interface and an implementation.
  - Examples:
    - The `Order` interface is implemented in `OrderImpl`.
    - The `CommerceItem` interface is implemented in `CommerceItemImpl`.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Other examples of classes with an interface and an implementation are `ShippingGroupImpl` and `PaymentGroupImpl`.

# Order Repository



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The item descriptors for the order repository are configured in /atg/commerce/order/OrderRepository.xml. The slide lists some of the item types defined in the XML file.

## Mapping of Classes to Repository Items

- OrderTools has properties that
  - Map Java classes to repository item types
  - Map repository item types to Java classes
  - Set default object types
  - Set default classes for price info objects
- These mappings determine which classes are used to instantiate order-related objects.
- If order classes are extended, these mappings must be altered for new classes to be used.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Types can be used by an application to differentiate objects that should be handled differently. For instance, CRS includes four types of commerce item: giftWrapCommerceItem, default, configurableCommerceItem, and subSkuCommerceItem. Each one of these types maps to its own implementation of the CommerceItem interface.

## Selected OrderTools Mapping Properties: Java Class to Repository Item Mapping

- beanNameToItemDescriptorMap

Maps Java classes to repository item descriptors

- itemDescriptorToBeanNameMap

Maps repository item descriptors to Java classes

itemDescriptorToBeanNameMap is the reverse of beanNameToItemDescriptorMap. The values are automatically filled in from there.

This is not an exhaustive list of OrderTools' mapping properties. The item descriptor in the maps refers to the cached repository item and the bean name is the actual bean.

## Selected OrderTools Mapping Properties: Object Type Mapping

- `orderTypeClassMap`
  - Example:  
`default=atg.projects.store.order.StoreOrderImpl`
- `defaultOrderType`
- `commerceItemTypeClassMap`
  - Examples:  
`default=atg.commerce.order.CommerceItemImpl`  
`giftWrapCommerceItem=atg.projects.store.order.GiftWrapCommerceItem`
- `defaultCommerceItemType`

A map of the application's order types to Java classes

The default type to use when creating new orders

A map of the application's commerce item types to Java classes

The default type to use when creating new commerce items

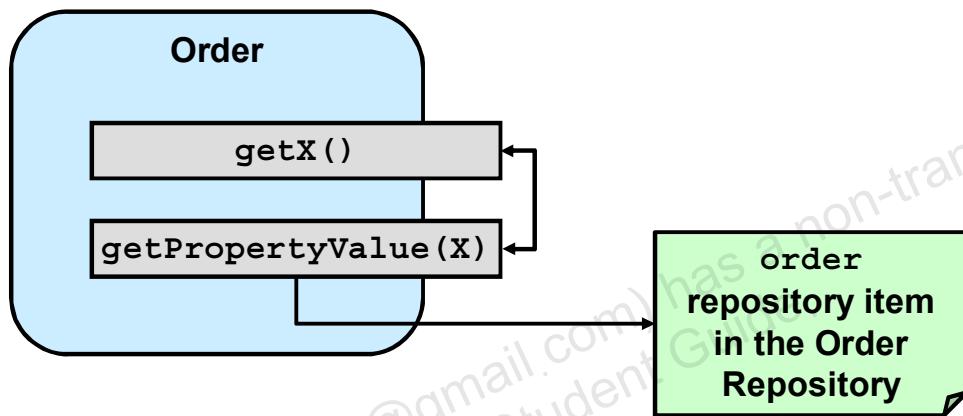
**ORACLE**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

This is not an exhaustive list of OrderTools' mapping properties. There are `TypeClassMap` and `defaultType` properties for each piece of an order, such as shipping groups and handling instructions.

## Order Property “Flow Through”

Many properties of the Java classes “flow through” to the underlying repository item by using `getPropertyValue()` and `setProperty()`.



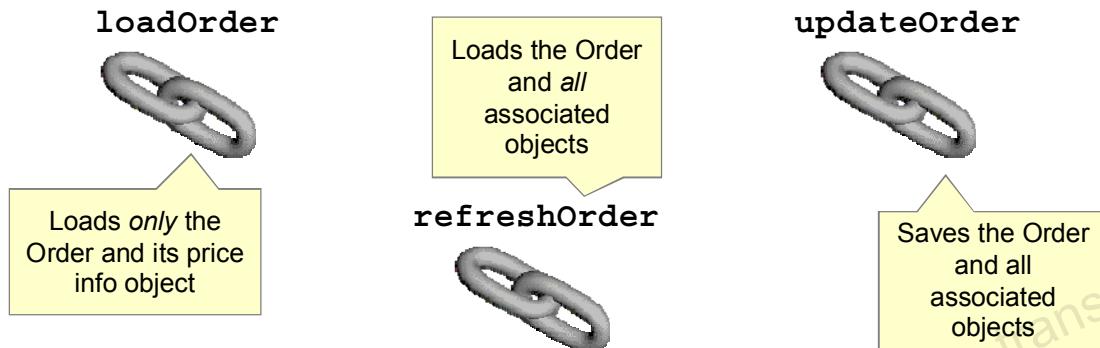
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The Order is a complex object that is updated throughout the shopping experience. ATG Commerce provides a number of wrapper classes, such as the Order, that provide a layer of abstraction to the components that manipulate them (OrderTools and OrderManager). The example in the slide is a simplistic representation of gathering and setting order information through the Order object, using `Order.getX()` for a property called x. Just as a reminder the order repository item shown in the slide is the cached repository item. The purchase process form handlers update the repository item in the database at certain points in the process.

## Order Persistence

- Order persistence is handled by pipelines.



- Different pipeline processors handle the saving and loading different pieces of the order.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

`loadOrder` loads the Order from the repository item. `updateOrder` saves the Order information to the order repository.

## Extending Order Persistence

To extend order persistence, perform the following steps:

- Add necessary tables and columns to the database.
- Add properties to the appropriate item descriptor in the Order Repository.
  - Best practice: Give the repository property the same name as the Java property.
- Use `Order.getPropertyValue()` and `Order.setPropertyValue()` to read and write properties.
- Add the property name to the appropriate load processor's `loadProperties` property.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The default load processors are found in the `/atg/commerce/order/processor` namespace. Example load processors include `LoadCommerceItemObjects`, `LoadPriceInfoObjects`, and `LoadShippingGroupObjects`.

**Note:** There is a similar `savedProperties` property on the save processors.

## Summary

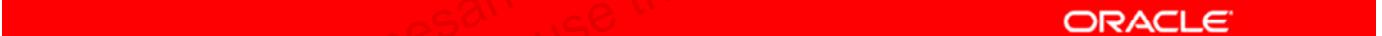
In this lesson, you should have learned how to:

- Describe the design pattern of components used to work with orders
- Extend order-related processes and order persistence

## For More Information

### *ATG Commerce Programming Guide*

- Working With Purchase Process Objects
- Configuring Purchase Process Services
  - Working with ATG Commerce Form Handlers
  - Managing Transactions in the ATG Commerce Form Handlers
- Processor Chains and the Pipeline Manager

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Practice 9 Overview: Extending Orders

This practice covers extending:

- The Order class to include the `reward` property
- The Order Repository and order persistence to include the `reward` property
- The creation of orders to set the `reward` property



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

# 10

## Pricing Architecture

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to describe:

- How ATG Commerce calculates prices
- What options are available for setting base prices
- How ATG Commerce handles dynamic discounts

# Road Map

- Pricing engine
- Dynamic discounts



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Pricing Engine

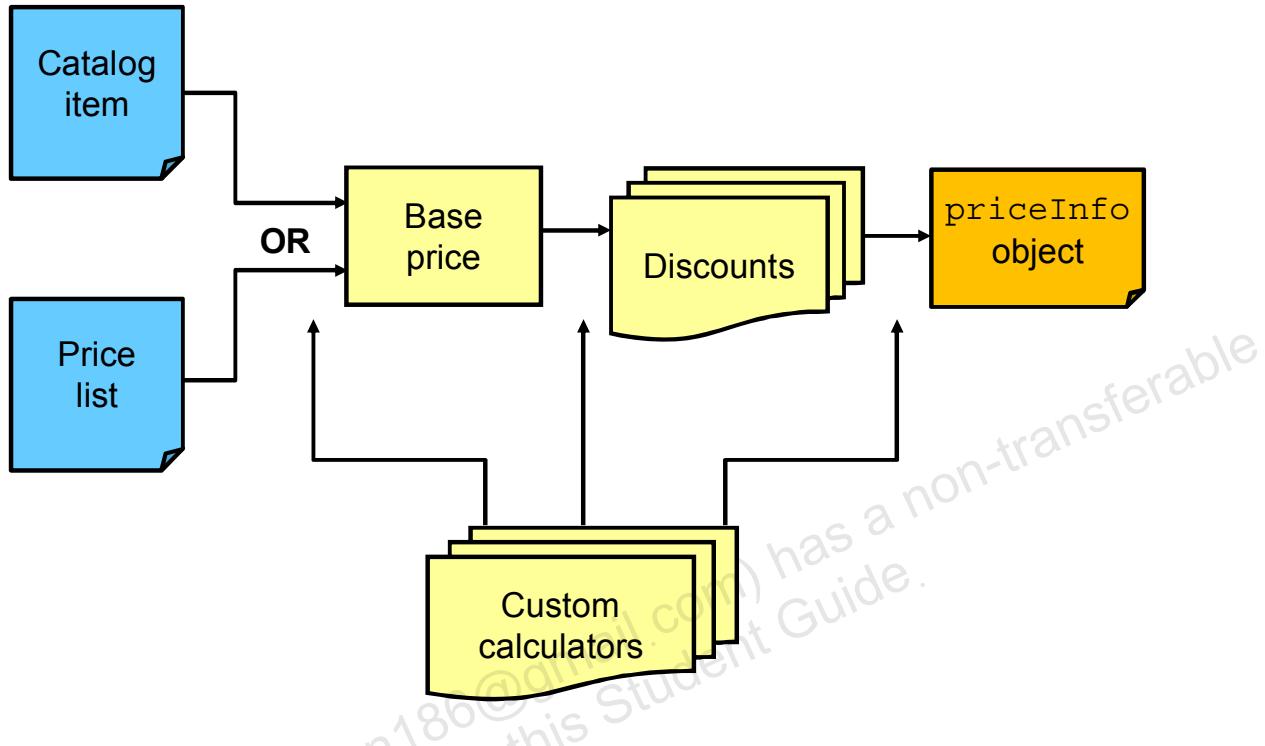
- Prices are calculated by ATG's pricing engine. The pricing engine:
  - Calculates the base price and then any discounts, based on the current user and site
  - Can easily be extended with custom calculators
- The pricing engine does not include a tax calculator.
  - Designed to be integrated with a third-party application such as CyberSource



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Pricing Process for Items



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

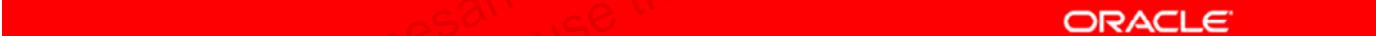
Pricing can be set either on the catalog items (products or SKUs) or by price lists. If the price is set on the catalog items, there is one base price for all customers. If price lists are used, different groups of customers can have different base prices.

The base price is then modified by any applicable discounts (also known as promotions or pricing adjustments). The result is a `priceInfo` object that contains the final price, the base price, and information about any adjustments made to calculate the final price.

The store can add custom calculators before or after the base price is calculated, or after the discounts have been calculated. Custom discount types can also be created.

## Pricing in Catalog Objects

- If pricing is set in the catalog, there will be one base price for all users.
- Prices can be set in either SKU or product objects.
- Pricing is set in SKUs by default.
  - Recommendation: Keep pricing at the SKU level for greater flexibility.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Pricing in Price Lists

- Using price lists
  - Separates pricing from the catalog
  - Allows different base prices for different groups of users
    - Examples: By locale, by organization, and by site
- Prices in price lists
  - Can be set by SKU or by product
  - Can be inherited from other price lists
  - Allow volume pricing
- SKUs not on the price list use the price from the catalog item.

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Road Map

- Pricing engine
- Dynamic discounts



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Discounts

- ATG supplies a powerful discount engine.
- Discount objects are called “promotions.”
  - Also referred to as “pricing models” or “pricing adjustments”
- Discounts are defined by the
  - Size of the discount
  - Thing or things being discounted
  - Conditions to qualify for the discount, if any
- You can add custom discount types.

## Discounts: Examples

- If the order contains three or more shirts, one shirt is free.
- For every three shirts in the order, one shirt is free.
- If the user has more than 5000 loyalty points, the order total is reduced by \$10.
- If the order contains one or more sweaters, one skirt is 20 percent off.
- For all orders, all skirts are 20 percent off.
- If the order total is over \$400, the shipping is \$1.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Note the difference between the first two examples. In the first example, the user will only get one shirt for free, even if there are six, nine, or twelve shirts in the order. In the second example, the user gets one free shirt for each three shirts in the order, so if the order contains six shirts, two of them will be free.

## Determining Who Gets a Potential Discount

Discount use can be limited in two ways:

- In the distribution of the discount
- In the conditions of the discount



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Limiting by Distribution

- Discounts can be
  - *Global*: Available automatically to all users
  - *Individual*: Only given to specific users
- Individual discounts have a range of properties that can limit when and how often a discount is used.

ORACLE®

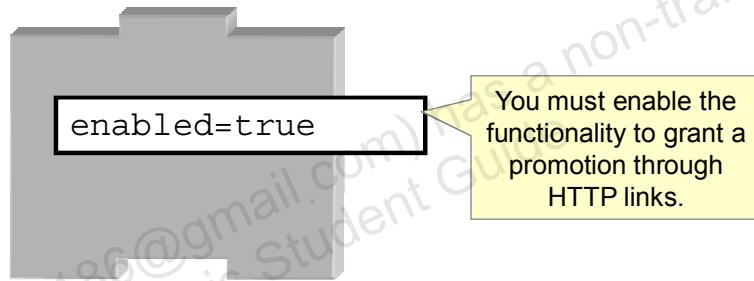
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Granting Individual Promotions to Users

Individual discounts can be given to users by:

- Creating and distributing a coupon code
- Creating a scenario that reacts to certain actions by granting discount to user
- Providing an HTTP link
  - The URL of the link includes the promotion ID.

### PromotionServlet



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

If a user receives a coupon code, that user can redeem the code on the site to claim the potential discount.

An example scenario is one that listens for an event where a new user registers and then grants the user 10% off the first order. Scenarios to distribute promotions are created through the ACC. (If your organization purchased ATG Outreach in a previous version, you may also have the ability to create scenarios through ATG Outreach as well.)

When providing an HTTP link, the URL of the link includes the promotion ID. The ability to grant promotions through HTTP links is not enabled by default. To use this feature, set the `enabled` property of `/atg/commerce/promotion/PromotionServlet` to `true`. You then pass a parameter in the HTTP link called `PROMO`.

## Limiting by Conditions

- Discounts can have conditions for use, such as:
  - An order over \$x
  - The order contains certain items
- Conditions can also refer to any profile properties, such as:
  - The number of loyalty points
  - The amount of past purchases
  - The organization
- Even if shoppers are given a potential discount, they still have to meet the conditions to get the discount.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

If shoppers receive a coupon code for \$10 off on an order over \$100, they can enter the coupon code on the order. However, the discount only applies if the order meets the condition. In other words, the order must contain over \$100 for the customer to receive the discount.

## Ways to Limit Promotions

- The dates the promotion can be used, such as:
  - From July 1 to 31
  - For 30 days after being granted
- The number of uses per customer
  - Example: Only one use per registered user for each time the promotion is granted
- The number of times the user can receive the promotion
  - Example: The user can be given the promotion five times.
- Usable by anonymous users



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

With the exception of the last bullet point (usable by anonymous users), all of these options are only available for individual promotions, not global promotions.

## Discounts Are Linked to Users

- Discounts are linked to *users*, not orders.
- Consider the following example:
  - Bob gets a coupon code for a promotion for free shipping on one order over \$100.
  - Bob comes to site and enters coupon code.
  - The free shipping promotion is added to Bob's list of available discounts on his profile.
  - His next order is not over \$100, so he does not get free shipping.
  - The order after that is over \$100, so he gets free shipping on that order.

**ORACLE**

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Differential Pricing Methods

- Two ways to implement differential pricing:
  - Price lists
  - Promotions
- Promotions may be better when the discount is rule-based.
  - Example: All internal employees receive 20% off.
- If discounts are not easily defined as rules, prices lists are better.
  - Price lists also allow pricing in multiple currencies.
- You can use both options in combination.

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Design Decision: Pricing

- Will prices be set
  - On the catalog objects or on price lists?
  - By SKU, product, or a combination of both?
- Will any custom calculators be needed to alter prices?
- How will differential pricing be implemented?
  - Using price lists?
  - Using promotions?
  - Both?



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

When running CIM, you can select to set prices by SKU or by product. The recommendation is to set prices by SKU. This allows for more flexibility in pricing. Even if you select to set prices by SKU, you can still set the price on the product. Any SKU price, however, would override the product's price setting.

## Summary

In this lesson, you should have learned how to describe:

- How ATG Commerce calculates prices
- What options are available for setting base prices
- How ATG Commerce handles dynamic discounts

Unauthorized reproduction or distribution prohibited. Copyright© 2015, Oracle and/or its affiliates.

Ganesan Sree (ganesan186@gmail.com) has a non-transferable  
license to use this Student Guide.

# 11

## Displaying and Extending Pricing

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to:

- Configure price lists
- Display static or dynamic prices on product pages
- Display order pricing on the cart page
- Understand the functioning of the pricing engine
- Extend the pricing engine to add pre- or post-calculators



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Road Map

- Price lists
- Item pricing
- Order pricing
- Extending pricing



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Price Lists

- Provide a means to define a set of prices that are applied to products, SKUs, or a combination of both
- Allow different price lists to be assigned to different groups of users according to business rules

You can overlay price lists.

Sku	Inherit	Price	Vol.	Inherit	Price	Vol.
A-Line Skirt	<input type="checkbox"/>	52.5	<input type="checkbox"/>	<input type="checkbox"/>	55.0	<input type="checkbox"/>
A-Line Skirt	<input type="checkbox"/>	52.5	<input type="checkbox"/>	<input type="checkbox"/>	55.0	<input type="checkbox"/>
A-Line Skirt	<input type="checkbox"/>	52.5	<input type="checkbox"/>	<input type="checkbox"/>	55.0	<input type="checkbox"/>
A-Line Skirt	<input checked="" type="checkbox"/>	55.0	<input type="checkbox"/>	<input type="checkbox"/>	55.0	<input type="checkbox"/>
A-Line Skirt	<input checked="" type="checkbox"/>	55.0	<input type="checkbox"/>	<input type="checkbox"/>	55.0	<input type="checkbox"/>

ORACLE

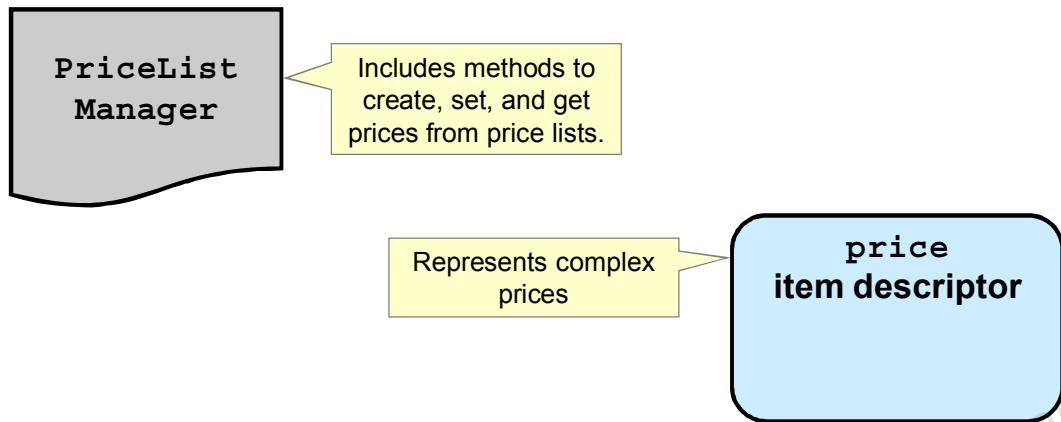
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Price lists provide a means of overlaying price lists. A price not found in the current price list can be looked up in the base price list, or on the catalog object. In the example in the slide, you see the Preferred Customer Price List. It has List Prices as its base price list. You can set the price for the Preferred Customer Price List (\$52.50 for three of the A-Line Skirt SKUs) or inherit the price from the base price list (\$55.00 for two of the A-Line Skirt SKUs). If the price had not been set in either price list, you could look at the A-Line Skirt in the catalog and get the price from the catalog object.

Using the example in the slide, you can assign the List Price price list to most users as the default price list. Your preferred customers could be assigned the Preferred Customer Price List with some more advantageous pricing.

You can view price lists in the BCC in two places: In the Price List area of Merchandising or on a particular SKU.

# Price List Architecture



Supported means of pricing include

- Price by product ID
- Price by SKU ID
- Price by product/SKU pairing

**ORACLE**

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Assigning Price Lists to Users

- The user profile has `myPriceList` and `mySalePriceList` properties.
- The price list can be set in several ways:
  - Inherited from an organization (typically, B2B), site, or the default price list set in `PriceListManager`
  - Set explicitly at the user level



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

`myPriceList` is a derived property. It uses the `firstNonNull` derivation. If the property is null, then the organization tree is checked for a price list. In the example in the slide, the user `alex@example.com` inherits the price list from his organization.

# Road Map

- Price lists
- Item pricing
- Order pricing
- Extending pricing

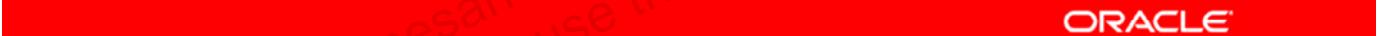


ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Item Pricing Droplets

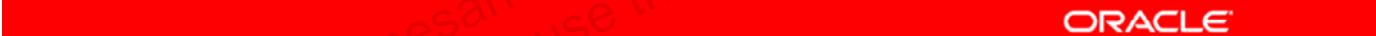
- Several droplets are available to retrieve individual SKU prices for display.
- Which one to use depends on:
  - Whether to display a price that includes dynamic discounts (promotions)
  - If volume pricing is being used
- All Commerce-related droplets are documented in the *Commerce Guide to Setting Up a Store*.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Static Pricing Droplets

- Two droplets available to retrieve static prices from price lists:
  - PriceDroplet
  - PriceRangeDroplet
- These droplets do not use the pricing engine to calculate possible discounts (promotions)
  - Pro: Price lookup is faster.
  - Con: The price may not reflect the actual cost.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## PriceDroplet

- PriceDroplet retrieves a single price from a price list.
- Input parameters:
  - sku or product (one is required)
  - priceList (optional; used to override the current shopper's price list)
- Output parameter
  - price (repository item)



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full class name for this droplet is atg.commerce.pricing.priceLists.PriceDroplet. This droplet also includes the error output parameter and the empty open parameter that can be used in case there is an error retrieving the price, or there is no price listed for the given item.

## PriceRangeDroplet

- PriceRangeDroplet retrieves the highest and lowest prices from product's list of SKUs on a price list.
- Input parameters
  - productId (required)
  - priceList and salePriceList (optional; used to override the current shopper's price lists)
- Output parameters
  - lowestPrice and highestPrice (doubles)



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full component name for the droplet is /atg/commerce/pricing/PriceRangeDroplet. It is based on the atg.commerce.pricing.PriceRangeDroplet class.

## Volume Pricing

- Volume pricing is possible when using price lists.
- Prices may be “bulk” or “tiered.”
  - **Bulk:** The price for all units changes when the quantity crosses thresholds.
    - For example: Buy 100 at \$10 apiece, or 200 at \$9
    - If you purchase 200, you pay **\$1800**.
  - **Tiered:** The price per unit in the tier changes when the quantity crosses thresholds.
    - For example: Buy the first 100 at \$10 apiece, and buy the next 100 at \$9
    - If you purchase 200, you pay **\$1900**.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## ComplexPriceDroplet

- ComplexPriceDroplet processes the price repository item for SKUs with volume prices.
  - The price repository item is returned by PriceDroplet.
- Input parameter
  - complexPrice (required; ID of price repository item)
- Output parameters
  - levelMinimums and levelMaximums
  - numLevels (integer)
  - prices

**ORACLE**

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Dynamic Pricing Droplets

- Two droplets are available to retrieve prices by using the pricing engine:
  - PriceItemDroplet
  - PriceEachItemDroplet
- The items are priced in isolation.
  - The contents of the cart are *not* taken into account when calculating discounts.
  - Pro: The price will include some discounts.
  - Con: The price lookup is slower.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Whether an item's price will be discounted when using these droplets depends on the type of promotion that is generating the discount.

If the promotion is not dependent on the contents of the cart (for instance, “10% off all bags”), the price returned will reflect the discount.

However, if the promotion does have conditions related to the cart contents (for instance, “buy two bags, get one free”), the discount will not appear until the entire order is priced (on the cart page).

## PriceItemDroplet

- PriceItemDroplet prices a single SKU by using the pricing engine.
- Input parameters
  - item: The SKU to price (required)
  - product: The product object (optional; may affect pricing, if discounts are product-based)
- Output parameter
  - element: A PricingCommerceItem object, with a completed priceInfo object



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full component name is /atg/commerce/pricing/PriceItem.

This is a partial listing of input parameters. There are also several input parameters that allow default values to be overridden, such as `locale`.

The only required input parameter is named `item`. The `item` parameter can be either `RepositoryItem` (such as a SKU object), which represents the item to be priced, or `CommerceItem` that can be directly priced. If the item supplied is `RepositoryItem`, the droplet will create a new temporary `CommerceItem` to hold the quoted price, the product, and the SKU.

## PriceEachItemDroplet

- PriceEachItemDroplet prices a collection of SKUs.
  - Each SKU is priced in isolation, as with PriceItemDroplet.
- Input parameters
  - items: The SKU items to price (required)
  - product: The product object (optional; may affect pricing, if promotions are product-based)
- Output parameter
  - element: A collection of priced PricingCommerceItem objects



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full component name is /atg/commerce/pricing/PriceEachItem.

This is a partial listing of input parameters. There are also several input parameters that allow default values to be overridden, such as `locale`.

This droplet is an alternative to calling PriceItemDroplet once for every SKU of a product.

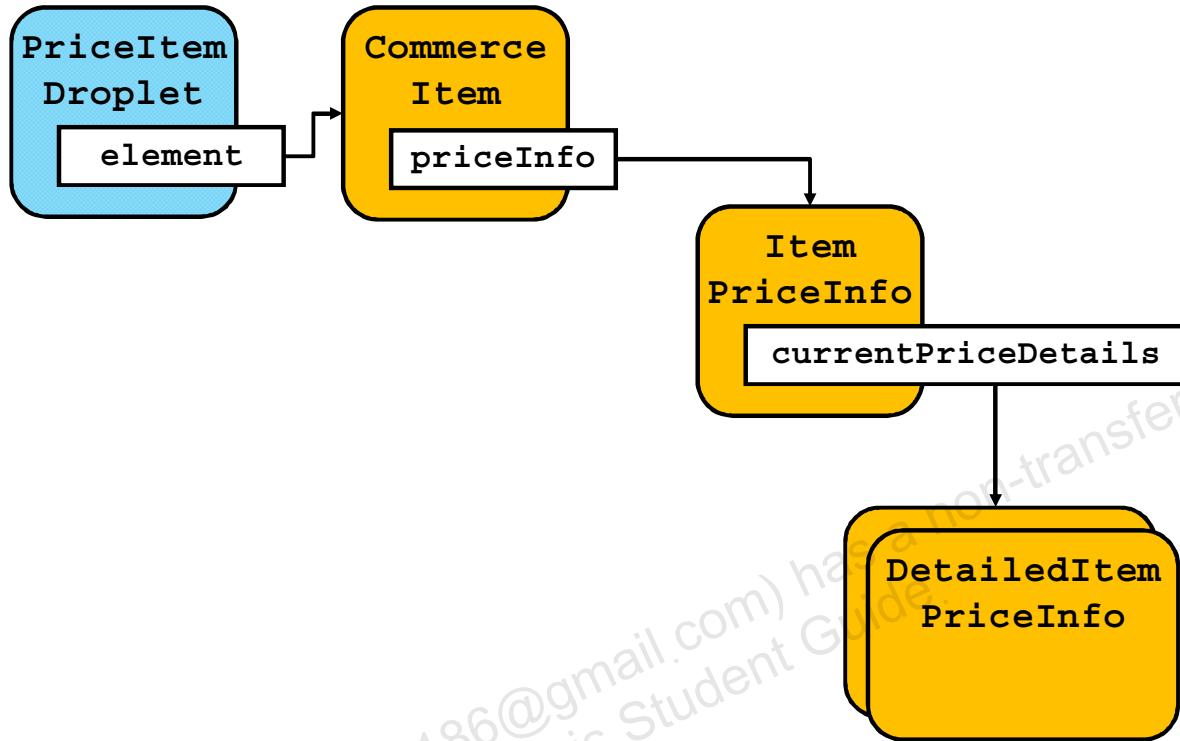
## PricingCommerceItem

- Dynamic pricing droplets create a temporary CommerceItem object for ItemPricingEngine.
  - The object is an instance of atg.commerce.pricing.PricingCommerceItem
- The class to use is not configurable, so it is not possible to extend PricingCommerceItem.

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## PriceInfo Objects

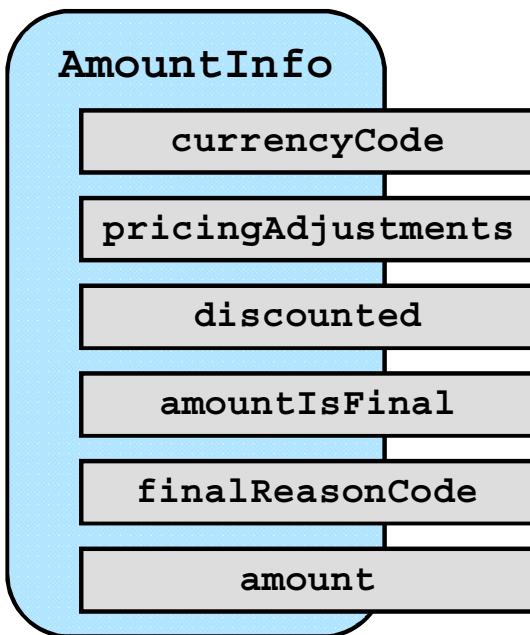


ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

PriceItem and PriceEachItem return a CommerceItem object, which points to an ItemPriceInfo object, which points to a list of DetailedItemPriceInfo objects. Each of these objects will be described in more detail on subsequent slides.

## Base Class: AmountInfo



Classes that extend  
AmountInfo:

- ItemPriceInfo
- DetailedItemPriceInfo
- ShippingPriceInfo
- TaxPriceInfo
- OrderPriceInfo

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

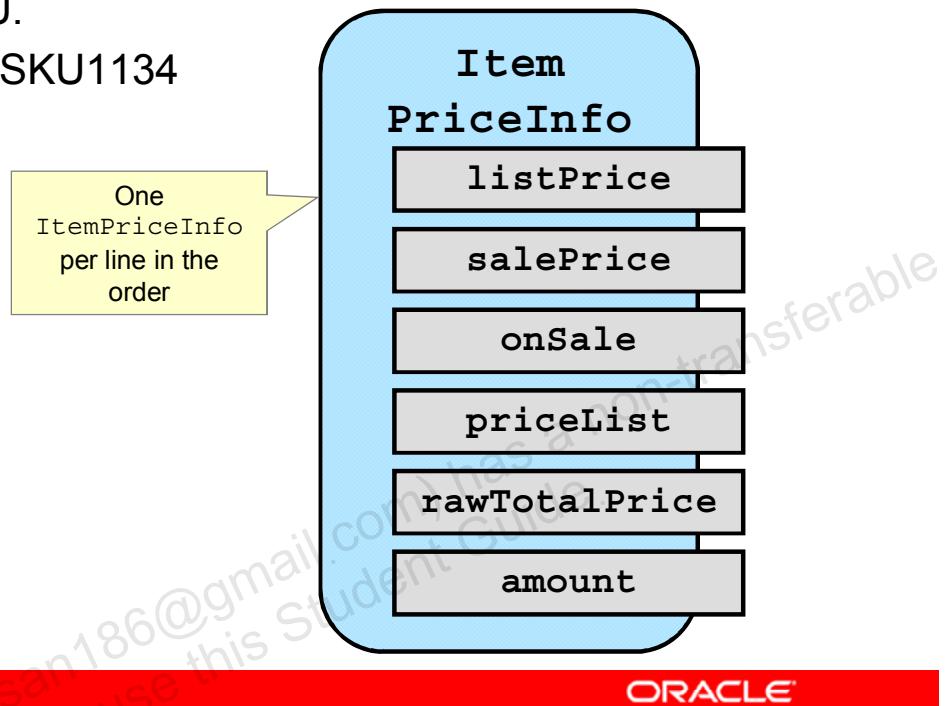
AmountInfo is the base class that all price information classes extend. Properties include:

- **currencyCode**: The currency of this amount
- **pricingAdjustments**: The list of discounts applied
- **discounted**: A Boolean that is set to true if the amount is discounted
- **amountIsFinal**: A Boolean that is set to true if the amount should not be recalculated
- **finalReasonCode**: The reason the amount is final
- **amount**: The total, with discounts

## ItemPriceInfo

ItemPriceInfo represents the total price for the full quantity of a particular SKU.

- Example: 3 x SKU1134



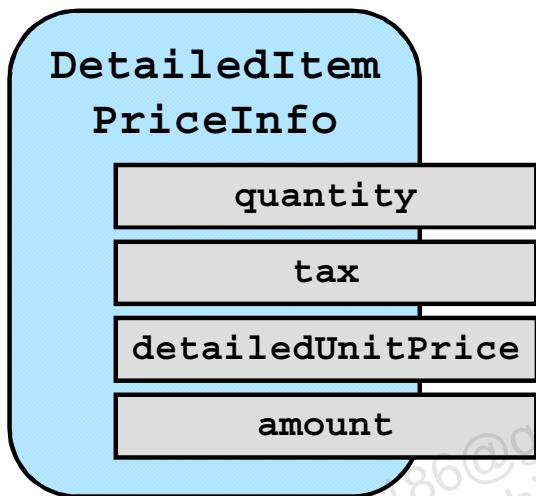
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full classpath to ItemPriceInfo is atg.commerce.pricing.ItemPriceInfo. The properties of the ItemPriceInfo object are:

- listPrice: The base price from the price list or catalog
- salePrice: The sale price from price list or catalog
- onSale: A Boolean that determines whether listPrice or salePrice is used as the starting price
- priceList: The price list used to calculate the price
- rawTotalPrice: The quantity x price, without any discounts
- amount: The quantity x price, with discounts

## DetailedItemPriceInfo

- In some cases, not all units of a SKU will have the same price.
- DetailedItemPriceInfo objects represent prices for the different groups of SKUs.



Example: 3 x SKU1134 with a “buy 2, get 1 free” promotion would result in two `DetailedItemPriceInfo` objects.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full classpath to `DetailedItemPriceInfo` is `atg.commerce.pricing.DetailedItemPriceInfo`. In example in the slide, there would be two `DetailedItemPriceInfo` objects:

1. 2 x SKU1134 at \$11
2. 1 x SKU1134 at \$0

There is at least one `DetailedItemPriceInfo` per `ItemPriceInfo`.

Some key properties of the `DetailedItemPriceInfo` class include:

- `quantity`: The quantity covered by this `DetailedItemPriceInfo` object
- `tax`: The tax applied to these items
- `detailedUnitPrice`: The price per unit
- `amount`: The quantity x `detailedUnitPrice`

For the full list, refer to the API.

## Road Map

- Price lists
- Item pricing
- Order pricing
- Extending pricing



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Order Pricing

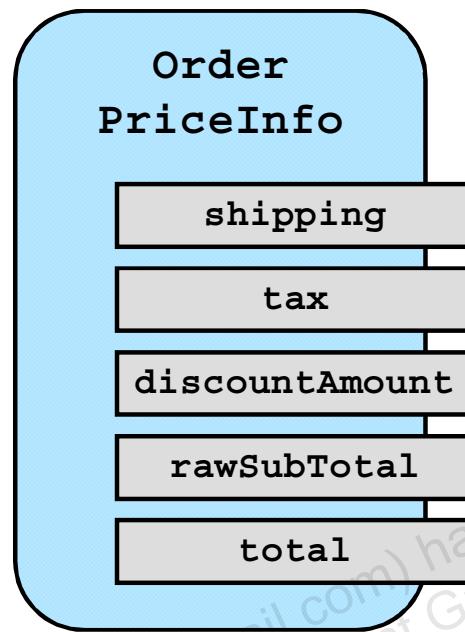
- The Order is automatically repriced when commerce items are added or removed.
- Order repricing can also be triggered by using `RepriceOrderDroplet`.
  - This droplet should be included on any pages that display the order total or subtotal, such as the
    - Cart page
    - Shipping page
    - Billing page

**ORACLE**

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full path to the `RepriceOrderDroplet` component is  
`/atg/commerce/order/purchase/RepriceOrderDroplet`.

## OrderPriceInfo



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Properties of the OrderPriceInfo object include:

- **shipping:** The shipping amount
- **tax:** The tax amount
- **discountAmount:** The amount the order has been reduced by discounts
- **rawSubTotal:** The total without order-level discounts
- **total:** The total with discounts

For the full list of the OrderPriceInfo class' properties, refer to the API.

## Road Map

- Price lists
- Item pricing
- Order pricing
- Extending pricing



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Objects That Can Be Priced Dynamically

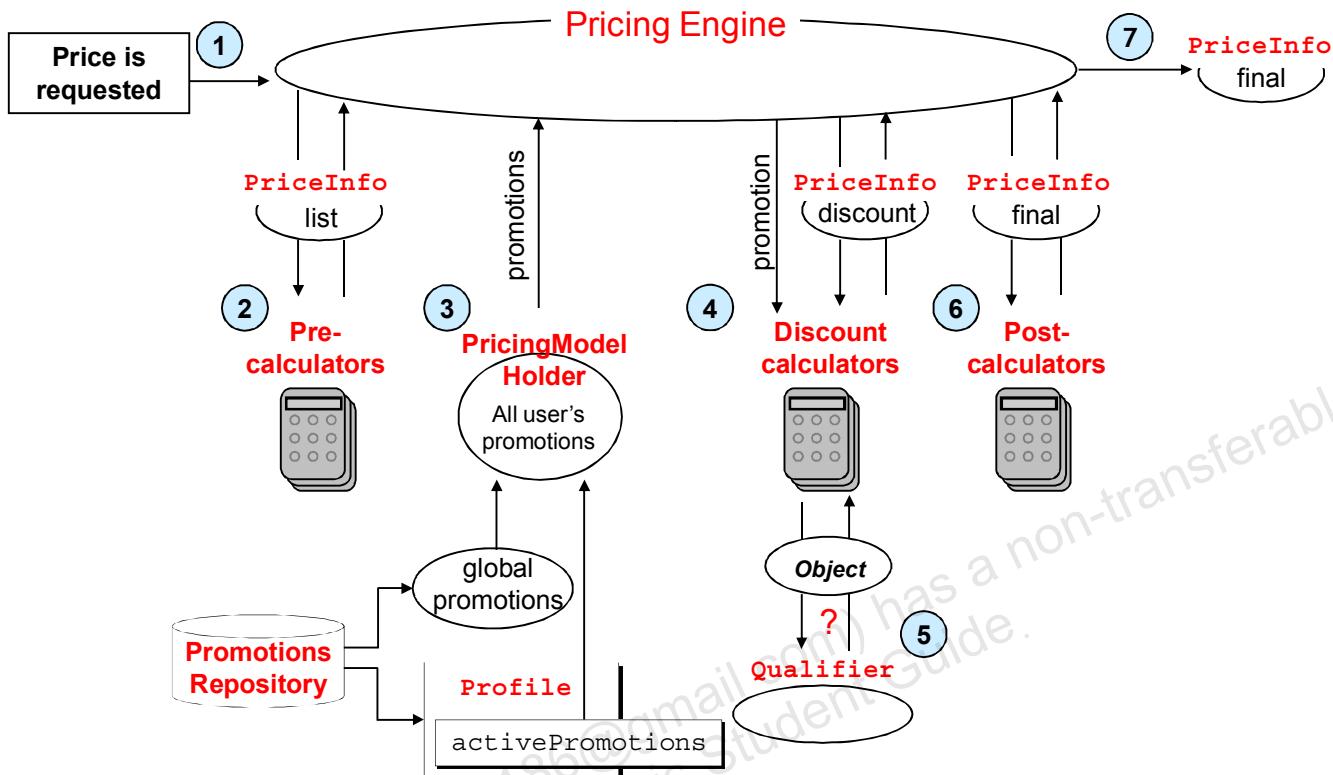
- Four commerce object types out-of-the-box can be priced dynamically:
  - Item
  - Order
  - Shipping
  - Tax
- The pricing mechanism is similar for each.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## How Pricing Works



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

1. A pricing operation is requested and the **PricingEngine** is invoked.
2. **PricingEngine** constructs a new **PriceInfo** object, and passes it to pre-calculators, which set an initial, undiscounted price in the **priceInfo**.
3. All discount promotions associated with the user are retrieved from **PricingModelHolder**, which contains all promotions specific to the current user (from the **Profile**), and global promotions available to all users.
4. **PricingEngine** iterates through the user's promotions and invokes each promotion's discount calculator one at a time.
5. Each discount calculator uses the **Qualifier** service to check whether the object being priced qualifies for the promotion associated with that calculator. If so, the price is discounted, and qualified objects are marked as having been used.
6. **PricingEngine** sends **PriceInfo** to post-calculators, which may further alter the price.
7. **PricingEngine** returns the modified **PriceInfo**.

## Ways to Extend Pricing

- Extend the pricing engine
- Extend existing pre-calculators
- Add pre- or post-calculators
- Extend or add discount calculators (promotion types)



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The slides in this lesson look at the existing calculators and how to add a pre- or post-calculator. The lesson titled “Promotions Architecture” covers adding custom discount calculators.

## Extending the Pricing Engine: Default Classes

- The default pricing engine classes are:
  - ItemPricingEngineImpl
  - OrderPricingEngineImpl
  - ShippingPricingEngineImpl
  - TaxPricingEngineImpl
- All classes are in `atg.commerce.pricing`.

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Extending the Pricing Engine: Critical Methods

- Critical methods of ItemPricingEngineImpl
  - priceItem()
  - priceEachItem()
  - priceItems()
- Critical method of OrderPricingEngineImpl
  - priceOrder()



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The methods listed in the slide do the following:

- priceItem(): Prices a single item in a context
- priceEachItem(): Prices each of a List of items in a context
- priceItems(): Prices a List of items together in a context
- priceOrder(): Prices an order within a context

ShippingPricingEngineImpl has a priceShippingGroup() method that prices the shipping group in a context. TaxPricingEngineImpl has a priceTax() method that taxes an order within a context.

The four default pricing engine implementations share many other method names around pre- and post-calculations.

# Extending Existing Item Pre-Calculators

Existing item pre-calculators:

- ItemPriceCalculator
- **ItemListPriceCalculator**
- ItemSalePriceCalculator
- ConfigurableItemPriceCalculator
- ConfigurableItemPriceListCalculator

The superclass of the other calculators

Typically, the first in a series of calculations, providing the starting price for other calculators

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

There are several existing item pre-calculators:

- ItemPriceCalculator: This is the superclass of the other calculators.
- ItemListPriceCalculator: Determines and sets the list price
- ItemSalePriceCalculator: Determines and sets the fixed sale price, if any. Dynamic discounts (promotions) are calculated later.
- ConfigurableItemPriceCalculator: Prices configurable items without price lists
- ConfigurableItemPriceListCalculator: Prices configurable items with the list price list
- ConfigurableItemPriceListSaleCalculator: Prices configurable items with the sale price list

## Extending Existing Order Calculators

- Existing order pre-calculator:
  - OrderSubtotalCalculator
    - Calculates the raw subtotal by summing all the item amounts
- Existing order post-calculator:
  - OrderAdjustmentCalculator
    - Adjusts the amount based on manual adjustments (generally made by customer service agents from CSC)



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Adding Pre- or Post-Calculators

To add pre- or post-calculators:

1. Create a calculator that implements the appropriate interface
  - ItemPricingCalculator
  - OrderPricingCalculator
  - ShippingPricingCalculator
  - TaxPricingCalculator
2. Add the calculator to the preCalculators or postCalculators property of the relevant pricing engine.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Extending the PriceInfo Class

If your pricing customizations extend one of the PriceInfo classes, you need to update the following properties:

- OrderTools.beanNameToItemDescriptorMap
- OrderTools.defaultItem/Order/Shipping/TaxPriceInfoClass
- The relevant pricing engine's priceInfoClass property



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Quiz

Ways to extend pricing in ATG Commerce include extending the pricing engine, existing pre-calculators, or discount calculators.

- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

**Answer: a**

## Summary

In this lesson, you should have learned how to:

- Use droplets to retrieve and display static or dynamic prices
- Describe the key pricing objects and the pricing engine for pricing items and orders
- Extend the pricing engine to add new calculators or override existing ones



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## For More Information

- *ATG Commerce Guide to Setting Up a Store*
  - Managing Price Lists
  - Appendix: ATG Commerce Servlet Beans
- *ATG Commerce Programming Guide*
  - Using and Extending Pricing Services



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

## Practice 11 Overview: Displaying and Extending Pricing

This practice covers the following topics:

- Extending `OrderPriceInfo` to include points earned
- Implementing a custom pricing calculator to calculate points earned for an order
- Extending the persistence of `OrderPriceInfo`



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.