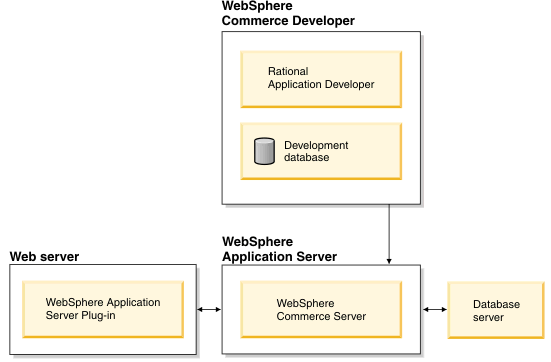
**WebSphere Commerce common architecture**



The Web server is the first point of contact for incoming HTTP requests for our e-commerce application. In order to interface efficiently with the WebSphere Application Server, it uses the WebSphere Application Server plug-in to manage the connections between the two components.

The WebSphere Commerce Server runs within the WebSphere Application Server, allowing it to take advantage of many of the features of the application server. The database server holds most of your application's data, including product and customer data. In general, extensions to your application are made by modifying or extending the code for the WebSphere Commerce Server. In addition, you may have a need to store data that falls outside of the realm of the WebSphere Commerce database schema within your database.

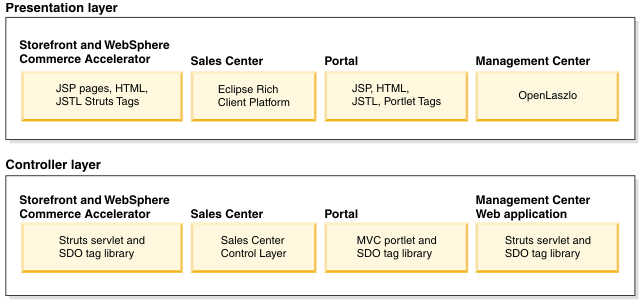
Developers use Rational Application Developer to perform the following tasks:

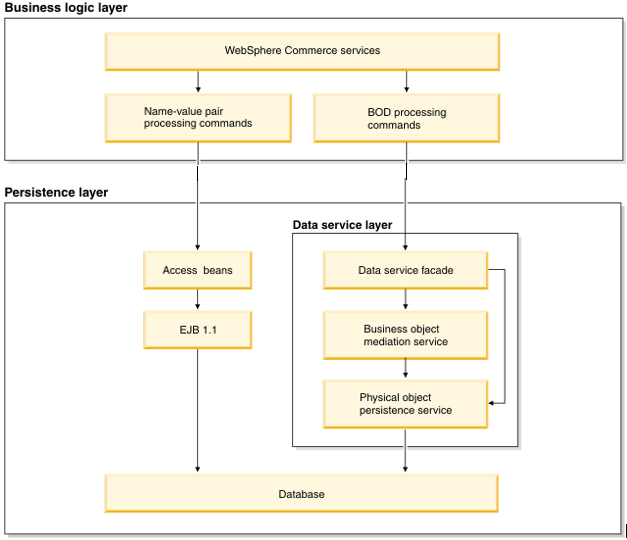
* create and customize storefront assets such as JSP and HTML pages
* create and modify business logic in Java
* create and modify access beans and EJB entity beans
* test code and storefront assets
* create and modify Web services

The WebSphere Commerce development environment uses a development database. Developers can use their preferred database tools (including Rational Application Developer) to make database modifications. WebSphere Commerce supports a one to one mapping between the WebSphere Commerce instance and the WebSphere Commerce database. Running multiple WebSphere Commerce instances against the same database is not supported.

New to this release is the further decoupling of the presentation tier from the business logic tier to better enable support for multiple sales channels. A sales channel is a method that a customer can use to purchase merchandise for example, in-store, from an online store, or from a call center.

WebSphere Commerce is multichannel-enabled, meaning that WebSphere Commerce can support transactions across various sales channels. The framework enhancements in this release support multiple presentation layers, responsible for displaying results, which decouple control logic from business logic.





The preceding diagram depicts how WebSphere Commerce supports two channels: the Web channel and the sales channel. For the Web channel the presentation is rendered using JSP pages and the Web controller layer uses Struts. For the sales channel, the display uses the Eclipse rich client technology. The presentation is rendered with Eclipse views and editors implemented using SWT components. Regardless of the channel, the business logic facade, a generic interface implemented as a stateless session bean, is used by controller calls to invoke controller commands. The command layer is implemented as WebSphere Commerce commands. The persistence layer provides EJB 2.0 support.

****

**Business models**

In WebSphere Commerce, a business model represents a sample business situation in which the WebSphere Commerce product may be used. A business model describes a scenario in which various parties use WebSphere Commerce to achieve their needs. The five business models provided by WebSphere Commerce are:

* B2B direct
* Consumer direct
* Demand chain
* Hosting
* Supply chain

Within each business model, WebSphere Commerce provides one or more samples, referred to as starter stores, which may be used as a starting point to develop online sites. You can create other business models to suit your business needs.

**Business processes**

Represent the processes available in WebSphere Commerce divided by business model. The business processes are divided into three areas:

**Administrative processes**

Processes that are used to administer a site, a store, or an organization. Administrative processes are described generically. These processes are generally used as-is, that is a change to or an addition of an additional administrative process will usually entail customizing WebSphere Commerce.

**Starter stores**

Starter stores contain sample processes that would be followed by customers of the store. Many different kinds of stores, satisfying a wide range of business needs can be created with WebSphere Commerce. Use the processes described in starter stores as a guideline, or a starting point, for site development. Changing or adding a process to a starter store processes will require changes to the site design. Frequently, this type of change does not require customizing the underlying infrastructure.

**Solution**

A solution describes the high-level view of how all the administrative processes and starter store processes fit within the framework of the overall business model. A solution combines processes into a coherent picture which explains the relationship between the various process groups.

**Presentation layer**

The presentation layer is responsible for displaying results. By default, there are two supported types of presentation layers supported: Web and rich client. For the Web presentation layer the display is rendered using JSP files, whereas, for the rich client the presentation is rendered with Eclipse views and editors implemented using SWT components.

**Service layer**

The service layer, implemented using OAGIS messages, is a channel-independent mechanism that can access WebSphere Commerce business logic. The service layer segregates the implementation of business logic such as order and catalog. This segregation permits the underlying implementation to change without requiring that the caller change. All clients, including Web clients and back-end services, go through the service layer to run business logic. The service layer supports two transport mechanisms: local Java binding and Web services.

**Business logic**

The business logic layer is where business rules are implemented independent of the presentation layer. Business logic is implemented using the command pattern. Two types of commands are implemented:

* controller commands - accessible by the presentation layer and used as a coordinator of tasks.
* task commands - not accessible by the presentation layer but called from the controller commands. This command type is used to implement business rules.

**Persistence layer**

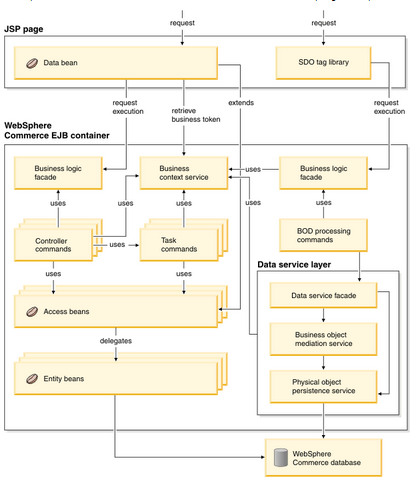
The persistence layer records the data and operations of the WebSphere Commerce system. The persistence layer represents entities within the commerce domain and encapsulate the data-centric logic required to extract or interpret information contained within the database. These entities comply with the Enterprise JavaBeans specification.

These entity beans act as an interface between the business components and the database. In addition, the entity beans are easier to comprehend than complex relationships between columns in database tables.

**Database schema**

WebSphere Commerce database schema, which includes over 600 tables, is designed specifically for e-commerce applications and their data requirements. The database schema supports persistence requirements for the WebSphere Commerce subsystems (Order, Catalog, Member, Marketing, Trading). WebSphere Commerce supports both DB2 and Oracle relational databases. Oracle is not supported on WebSphere Commerce - Express. In addition, to DB2 and Oracle, the development environment also supportsApache Derby.

# WebSphere Commerce functional overview



Although there have been enhancements in the WebSphere Commerce runtime, task commands, controller commands, access beans, and entity beans should continue to function as in previous releases.

**Controller layer**

The conductor of operations for a request. It controls the transaction scope and manages the session related information for the request. The controller first dispatches to a command and then calls the appropriate view processing logic to render the response.

**Presentation layer**

The presentation layer displays the result of command execution. The presentation layer can use JSP pages, or other rendering technologies.

**Business Context Service (BCS)**

A service that manages contextual information used by business components. The contexts include such information as globalization and entitlement.

**Business logic facade**

This generic interface is implemented as a stateless session bean which the controller calls to invoke controller commands.

**Controller commands**

A controller command business process logic such as OrderProcess. It invokes task commands to accomplish different unit of work in the business process. By default, access control is enabled for controller commands.

**Task commands**

A task command is an autonomous task that accomplishes a specific unit of application logic such as check inventory. A task command usually works with other task commands to complete processing of a controller command. By default, access control is not enabled for task commands.

**Access beans**

Access beans are simple persistent objects with setters and getters. The access bean behaves like a Java bean and hides all the enterprise bean specific programming interfaces, like JNDI, home and remote interfaces from the clients. Rational Application Developer provides tooling support to generate access beans from the schema.

**Entity beans**

Entity beans are used in the persistence layer within WebSphere Commerce. The architecture is implemented according to the EJB component architecture. The EJB architecture defines two types of enterprise beans: entity beans and session beans.