2
Development Concepts

This chapter will detail the framework and design principles that govern application development. You will be presented with environment concepts so that your application can be added to the existing V3 architecture. You will also become familiar with the modularity, baseline styles and interdependencies that make V3 development a fast paced development framework.

2.1. Layers

Layers are comprised of the Logical Support Structure for all of the components of the V3 system. Layers are the way that V3 organizes resources into strict dependencies. At lower levels, the layer structure is based on hardware resources. As you move up towards domains and applications, we come across the utility classes that make higher-level classes and methods possible. Remember that the layers do not represent logical steps in system operation, but rather logic steps in system development.

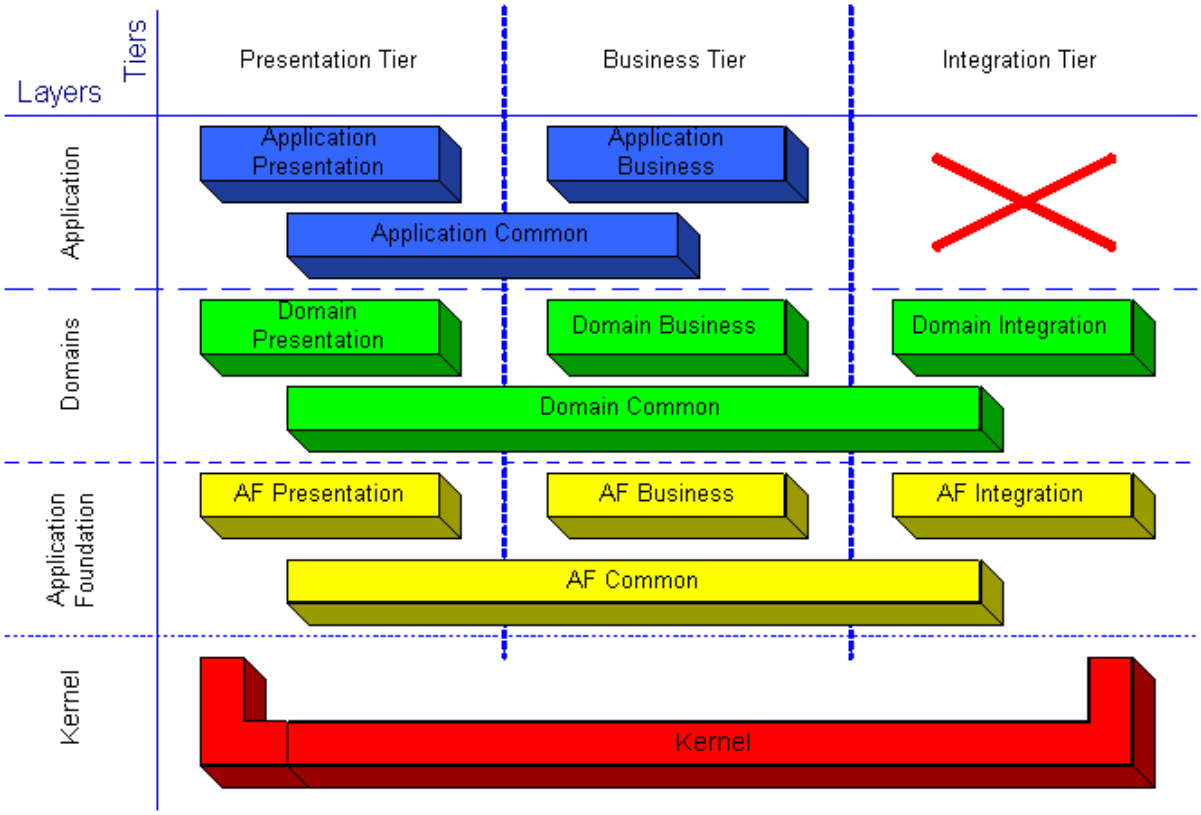

***Figure1**: A complete logical view of the Application Layers*

2.1.1. Logical Divisions

The V3 architectural layering is based on straightforward software factoring hierarchy. This factoring places a logical emphasis on the lower layers providing reusable services, subsystems and components to higher layers. The Application layers from top to bottom are:

- **Application Code** – course grained business processing & orchestration code
- **Domain Components** – course & fine grained logic (Enterprise, Domain, Semantic, Discrete)
- **Application and Components Frameworks** – (AF) eBay base classes and abstract methods.
- **Kernel** – domain diagnostic services, subsystems and support classes

The basic dependencies between the layers are that upper layers depend on lower layers – never the other way around. This is a strict requirement.

In development, every layer (and its respective components) except the application layer code for your feature will (usually) already exist. Some Domain components and Frameworks should be modified to support new applications, but they should also be as general and reusable as possible.

Each layer is partitioned into tiers (see section 2.2: Tiers). An application can be thought of as being decomposed into the various layers and tiers, but may be further partitioned on the basis of various systemic qualities, as in Domains (see section 2.4: Domains). The Kernel, AF and domain code are pure Java and do not rely on a J2EE application server. Thus, this code can be moved to any Java environment and function as is.

Each layer will in general have its own common code that is shared across the tiers. By shared we mean physical code sharing and not remote access sharing. Code that is *not* common across all the tiers must *not* live in the common code base for that layer. Kernel code, for example, that is passed (or marshaled) across two tiers is isolated into jars. These Kernel jars contain the remote interfaces and serialized class definitions that will travel back and forth across those tiers.

2.2. Tiers

Tiers, unlike layers, represent a logical subdivision of Application Logic. No matter which layer is being modified, the code is tiered so that general logical steps in data manipulation, assimilation, and recovery are always encapsulated in a standard way.

2.2.1. Logical Divisions

The tiered format divides the application into logical steps that can be built separately across the layer hierarchy. In any application, data from the database flows through the Integration tier, is manipulated by the Business tier, and packaged and unpackaged by the Presentation tier. This format flows in both directions – if a user makes a request, the request is unpackaged by the presentation tier, manipulated by the business logic tier, and the database is queried through the integration tier.

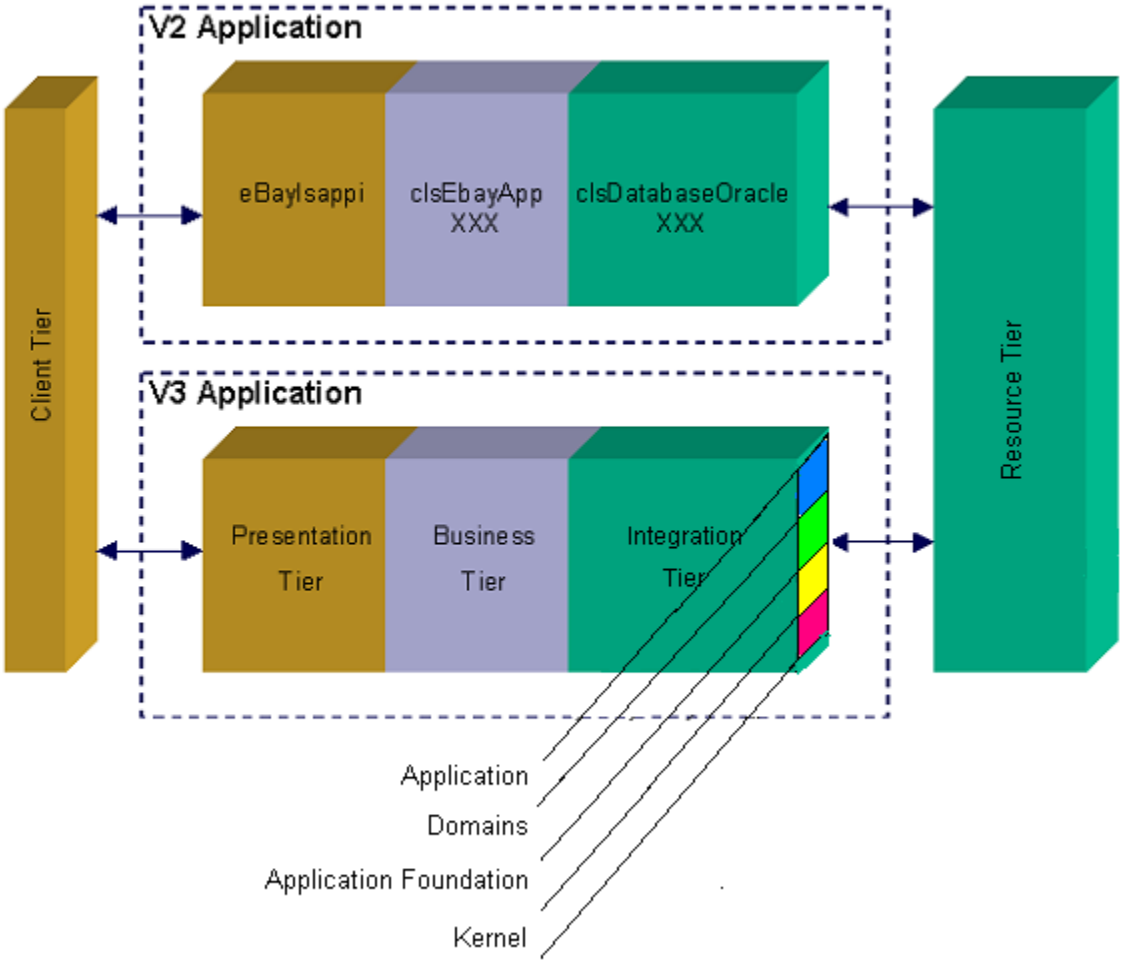

***Figure2**: A tiered comparison of V2 and V3 formats*

There are four basic tiers within the architecture:

- Client –browsers/apps connected over HTTP(s), API and batch clients
- Presentation - Request handling, XSL, JSP, etc…
- Business - High level application flow, Components, Rules Processing
- Integration - Data Access Layer, etc…
- Resource – network infrastructure, Websphere, etc…

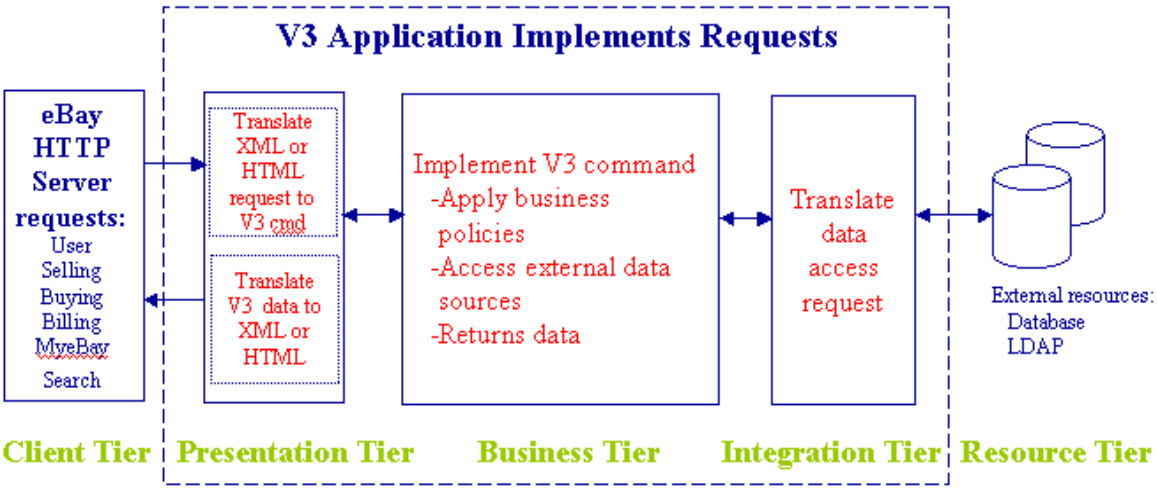

Figure3: *The basic functionality of the Tier structure*

There is another element that may be confused with a tier but is not. The common components of a layer are comprised of common code used throughout the logic flow across the tiers on any given layer. These elements are necessary but should not be confused with anything else.

The majority of the V3 system is concerned with the Presentation, Business and Integration tiers. Over time the architecture will extend out to the Client tier with various thick clients and various device types. The common tier, as discussed before, is common code and is, for the most part, already in place.

The tiers have a strict dependency order: client depends on presentation depends on business depends on integration. For example the Business Tier will never depend on the Presentation Tier. These dependencies are a strict requirement.