MarkLogic Server

Reference Application Architecture Guide

MarkLogic 10
May, 2019

Last Revised: 10.0, May, 2019

Table of Contents

Reference Application Architecture Guide

1.0 Understanding the Reference Architecture

The MarkLogic Reference Application Architecture is a three-tier application template and set of best practices for architects, developers, and administrators designing, developing, and deploying applications that use MarkLogic Server.

This guide covers the following topics:

1.1 Purpose

MarkLogic is a flexible and powerful platform, capable of fitting into many different application solutions. The MarkLogic Reference Application Architecture provides an three-tier application template and set of best practices for architects, developers, and administrators designing, developing, and deploying applications that use MarkLogic Server.

The MarkLogic Reference Application Architecture consists of the following parts:

- A description of the application architecture, the responsibilities of each tier, and the relationship between the tiers.
- A set of recommended best practices for developing reliable, large-scale application on top of MarkLogic Server.

1.2 Why a Three Tier Architecture

Historically, much of what end users see of a web application is rendered by a back-end server that manages application and session state, and constructs HTML for the browser. Today, browsers and browser frameworks are more capable, so the balance has shifted towards browser applications managing their own views, session state, and some of the business logic. It is also now common for applications to expose business services through REST-style APIs that are easy to integrate with, making it easier to swap out parts of the application.

There are many ways to structure a MarkLogic application, including the traditional 2-tier model. For example, MarkLogic Server includes an application server, so you can easily create two-tier MarkLogic applications that generate HTML for the browser from within MarkLogic Server using XQuery or JavaScript.

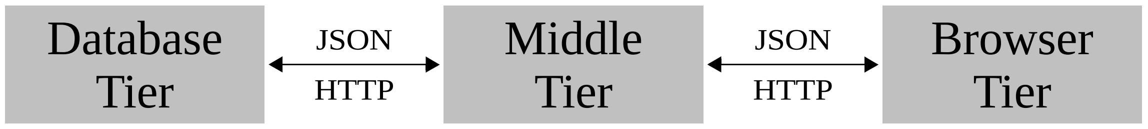
However, we've chosen a three-tier model that leverages JSON, REST over HTTP, and Java/JavaScript as the reference architecture because we believe it enables a team new to MarkLogic to become productive as quickly and easily as possible. The MarkLogic Reference Application Architecture offers the following advantages:

- The team can work exclusively with industry standard frameworks and programming languages (Java, Spring, JavaScript, AngularJS, JSON).
- Little knowledge of MarkLogic internals is required to get started.
- Separation of business logic from the data services layer makes it easier to integrate MarkLogic with existing enterprise infrastructure.

A three-tier architecture offers additional benefits such as the ability to scale and optimize the tiers independently and separation of security concerns.

1.3 Reference Architecture Overview

The MarkLogic Reference Application Architecture is a three-tier model containing database, middle, and browser tiers. As shown in the following diagram, JSON over RESTful HTTP is the transport mechanism between all tiers.



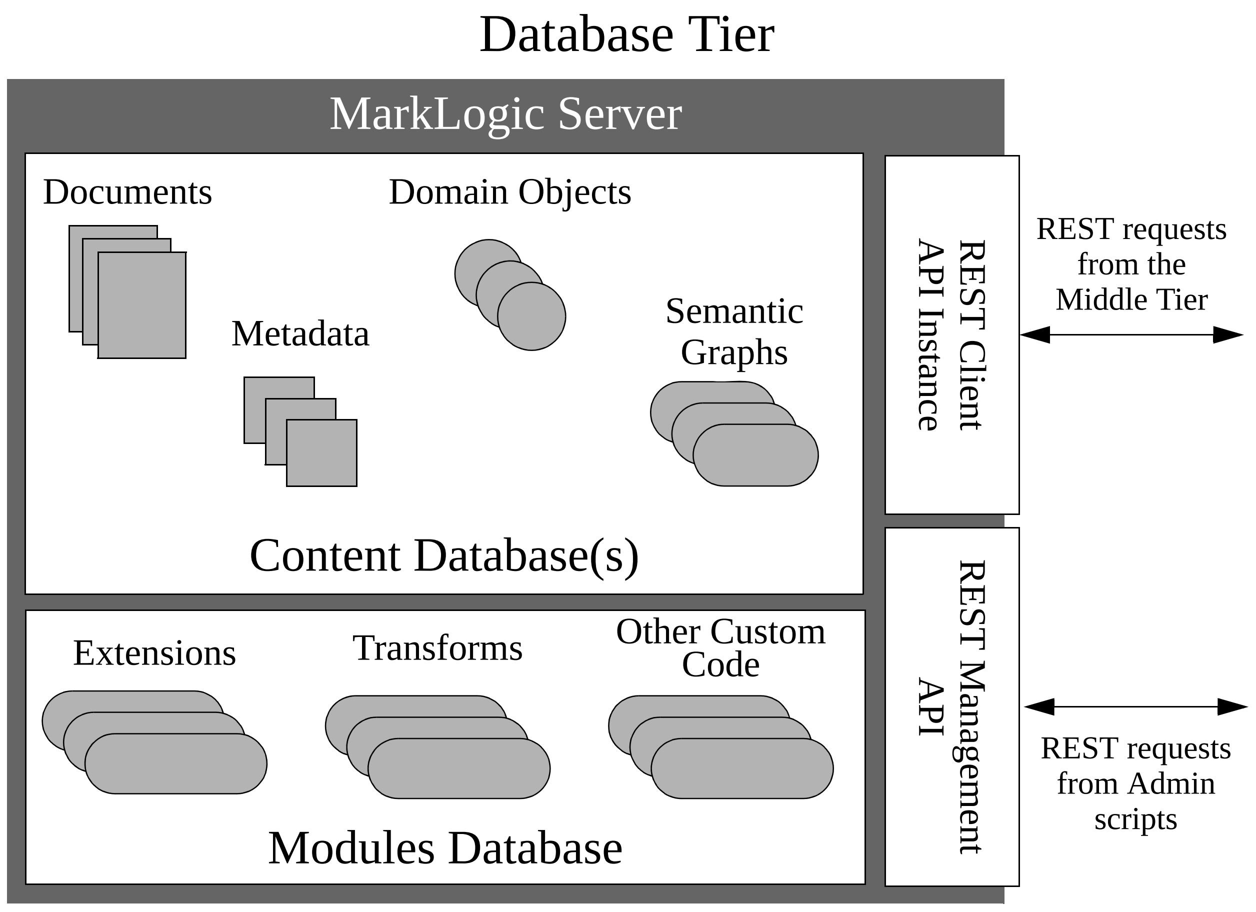
The database tier provides high availability, long-running data services to the middle tier. All the data required by the application is managed by MarkLogic Server in the database tier. Persistent application state is managed here for purposes of scalability and simplicity. This tier can include code that needs to run close to the data for performance reason or to enforce the data model. MarkLogic provides services and data to the middle tier through one of the powerful and extensible MarkLogic client APIs. For details, see "Database Tier" on page 4.

The middle tier provides data to and shares session state with the browser tier. In a MarkLogic application, it brokers exchanges between the browser and database tiers using one of the MarkLogic client APIs. The middle tier implements business logic verification and can have additional responsibilities such as rate limiting and integrations with non-MarkLogic external services. For details, see "Middle Tier" on page 6.

The browser tier contains the web application front-end that faces end users. The application includes code that runs in the browser, markup, and styles that tailor the user experience. In the MarkLogic Reference Application Architecture, the browser application is a rich client. That is, the browser tier fully owns the rendering of the UI, including decisions about how views are organized and most of transitions within a flow. The browser tier shares awareness of business logic with the middle tier. For details, see "Browser Tier" on page 7.

1.4 Database Tier

The database tier provides data and application services to the middle tier. The following diagram outlines the major components of the database tier.



MarkLogic Server manages data and code required by the application, such as the following:

- JSON, XML, Binary and Text documents
- Metadata about documents, such as permissions, quality, collections, and application-specific document properties
- Persistent application state, such as domain objects realized in the middle tier as Java or JavaScript objects
- Semantic graphs
- Schemas
- Extensions, transformations, and other application-specific code that needs runs in the database tier for purposes of performance, encapsulation, or enforcement of data rules

Storing such assets in the database provides transactional integrity to the application. The MarkLogic transaction model includes multi-statement transactions that enable applications to interleave transactional operations in the database tier with business logic in the middle and browser tiers.

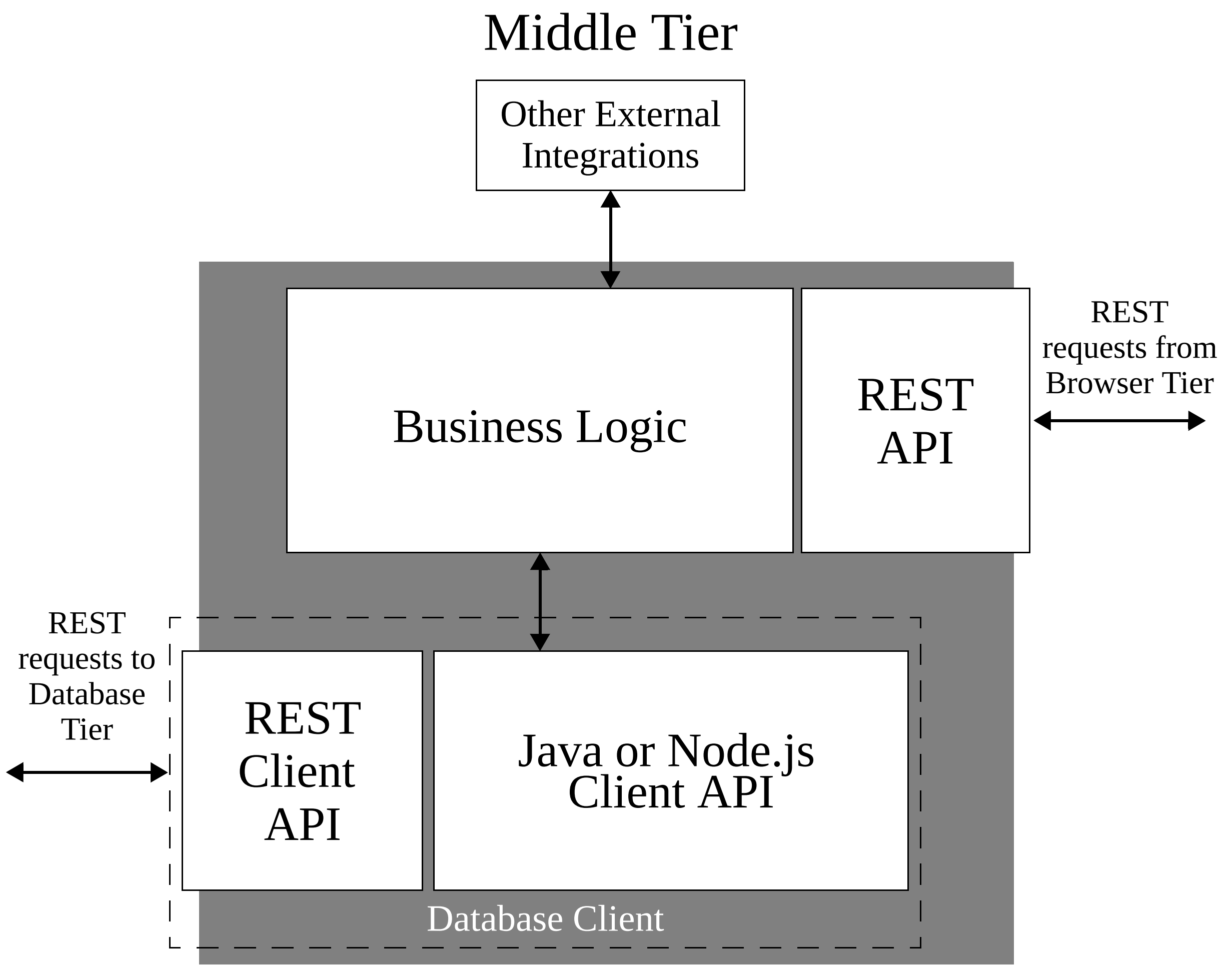
The middle tier communicates with MarkLogic Server through the REST Client API and the REST Management API. Incoming REST requests are handled by an HTTP App Server embedded in MarkLogic Server.

The REST Client API is the foundation of a family of MarkLogic client APIs that enable applications to create, read, update, delete, and search database content. The Java Client API and the Node.js Client API use this foundation. All the MarkLogic client APIs are extensible, so you can easily install and use content transformations, search customizations, REST resource services extensions, and other database tier library modules. All such database tier code is stored in a modules database and can be managed through the MarkLogic client APIs.

The REST Management API enables developers and administrators to manage, monitor, and review their MarkLogic Server configuration and status remotely through REST requests. Though MarkLogic Server has user interfaces for interactive administration and monitoring, the REST Management API makes it easy to script such tasks.

1.5 Middle Tier

The middle tier brokers inter-server communications between the browser tier and the database tier and other external services. The following diagram shows the major components of the middle tier of a MarkLogic application:


Middle Tier
Other External Integrations
Business Logic
REST API
REST requests from Browser Tier
REST requests to Database Tier
REST Client API
Java or Node.js Client API
Database Client

Even in a thick client model where the majority of the business logic may be in the browser tier and data flows between the browser tier and the database tier with little or no modification, the middle tier provides critical services such as the following:

- Coordination of inter-server communications for the browser tier, such as between the browser tier and the database tier or the browser tier and other external services.
- Security services. The middle tier can provide user authentication services and verify that data coming from the insecure browser tier adheres to the business rules.
- Network traffic optimization for the browser tier. Communication between the browser and the middle and database tier is typically a relatively long hop, so minimizing requests from the browser improves performance. The browser tier can make a single request to the middle tier that requires multiple requests between the middle tier and the database tier.

- Transactional integrity. The middle tier can coordinate application logically discrete operations that should be part of a single transaction, such as updating an account balance at the same time the user authorizes a withdrawal or deposit.

A rich browser client is usually the first line implementer of business logic, but the middle tier injects business logic as necessary, to support the services listed above.

The middle tier communicates with MarkLogic Server through a database client using one of the MarkLogic client APIs, such as the Java Client API or the Node.js Client API. These APIs access MarkLogic through REST requests over HTTP, while providing a fluent interface natural to your application. Using these APIs means developers do not need to learn a new programming or query language and injection risks via unsafe evaluation of strings of code can be eliminated.

The MarkLogic client APIs enable you to do the following:

- Insert, read, update, and delete documents, metadata, semantic triples, and domain objects, singly or in batches.
- Query documents and other data in the database using MarkLogic's powerful search features. Choose from among several query styles, depending on what best fits your needs.
- Extend the built-in services through several extension points. You can use and manage your extensions through the MarkLogic client APIs.

The data structures that are returned by MarkLogic through the client APIs can be passed through directly to the browser tier as JSON or XML. Similarly, domain objects stored in the database can be retrieved as native Java and JavaScript objects.

The business logic in the middle tier validates data delivered by the browser tier, as well as handling integration with other systems. The middle tier may publish a REST interface with which to communicate with the browser tier in a way natural to the application. This API should be flexible enough to support swapping in a different front-end implementation. Typically, data flows between the middle tier and the browser tier as JSON.

1.6 Browser Tier

MarkLogic does not require any particular web application architecture. The reference architecture promotes a thick client such as a Single Page Application (SPA), where the browser tier is responsible for all view rendering and transient application state. However, other browser tier solutions can be plugged into the architecture.

The browser tier uses JavaScript to interact with the middle tier through JSON services provided by an application-specific REST API. The services provided by this REST API model the tier interactions in a way that is natural to the application. The existence of this API does not necessarily require data transformations.

In the thick client, SPA solution, the browser is an Model-View-Controller (MVC) application that includes business logic and shares a high degree of fidelity with the data model exposed by the database tier. This enables the browser tier to consume data from the database tier (by way of the middle tier) with minimal transformation.

Though the browser tier implements much of the business logic, the browser is inherently more vulnerable to risks like injection and denial of service attacks. The middle and database tiers provide protection in the form of security, verification of the business and data rules, rate limiting.

The model can perform JavaScript object validation against a JSON schema. The view is HTML and CSS based.

1.7 Next Steps

Refer to the following table for suggestions of further reading and activities to continue learning about MarkLogic Server application development.

If you want to	Then see
Learn more about the Reference Architecture	"Samplestack: A Reference Architecture Instantiation" on page 12
Explore a full-feature MarkLogic Application based on the Reference Architecture	The `samplestack` application at http://github.com/marklogic/marklogic-samplestack
Learn about specific MarkLogic capabilities such as document operations, search, or semantics.	Tutorials on the MarkLogic developer community site, http://developer.marklogic.com/learn
Learn more about the MarkLogic client APIs	*Java Application Developer's Guide* *Node.js Application Developer's Guide* *REST Application Developer's Guide*
Learn about MarkLogic Server internals	The "Inside MarkLogic Server" paper on the MarkLogic developer community site, http://developer.marklogic.com/inside-marklogic

2.0 Recommended Best Practices

When designing your MarkLogic application, you should consider incorporate at least the following best practices into your development process:

- Organize your project around logical components
- Use source control for code, tests, and automation drivers
- Make project development and deployment easy to configure
- Automate development, testing, and deployment tasks
- Incorporate automated testing into all phases of development

2.1 Organize your project around logical components

You should organize your project in a way that preserves the logical separation of concerns in your application. This makes it easier to share the project among teams working in the different tiers and makes it easier to find source files and other assets.

2.2 Use source control for code, tests, and automation drivers

Your source code, tests, automation scripts, and other application assets should all be under source control. Source control preserves historical changes to your application and enables multiple people to work on a project concurrently. Putting all your application development and deployment components under source control also makes it easy to know goes into each release of your product. Even the smallest team benefits from source control.

2.3 Make project development and deployment easy to configure

The building, testing, and deployment of your application should be configuration driven.

For example, if you move your testing infrastructure to a new host, you should only need to update a single configuration file, not every test. Similarly, you should be able to change product build dependencies with minimal configuration file changes.

Consolidating such variables into configuration files makes it easier to track and less prone to error when change is required.

2.4 Automate development, testing, and deployment tasks

Project development involves many repetitive tasks. For example, a developer goes through the edit-test-debug cycle many times in a single day. Automating common tasks improves productivity and reproducibility.

Similarly, using automation to simplify testing removes barriers to frequent developer testing. Frequent testing improves product stability because problems are easiest to diagnose and fix close to the point at which they’re introduced. Test automation also helps improve the reliability of test results.

Automating the setup of your application development and deployment environment makes it easier to add new developers, manage dependencies, and provide a stable development, testing, and release platform.

2.5 Incorporate automated testing into all phases of development

It is important to have several levels of tests for your product. For example:

- Unit tests verify the functionality of specific sections of code, usually at the function/class/interface level. Unit tests are usually created by the developers. Unit tests can easily be run during a developer’s edit-test-debug cycle and before each checkin.
- Integration tests verify the interfaces and integration points between components, such as between the browser tier and the middle tier.
- Regression tests detect when previously working features break as an unintended consequence of a code change.
- System tests verify the end-to-end behavior of your application.
- Smoke tests, sometimes called sanity tests, are a small test set that check for some minimal level of operability. For example, you can use smoke tests to test a project integration branch before merging changes into a code line shared with a larger set of developers.

3.0 Technical Support

MarkLogic provides technical support according to the terms detailed in your Software License Agreement or End User License Agreement.

We invite you to visit our support website at http://help.marklogic.com to access information on known and fixed issues, knowledge base articles, and more. For licensed customers with an active maintenance contract, see the Support Handbook for instructions on registering support contacts and on working with the MarkLogic Technical Support team.

Complete product documentation, the latest product release downloads, and other useful information is available for all developers at http://developer.marklogic.com. For technical questions, we encourage you to ask your question on Stack Overflow.

4.0 Copyright