MarkLogic サーバー

Reference Application Architecture Guide

MarkLogic 8
2015 年 2 月

最終更新：8.0-1、2015 年 2 月

目次

Reference Application Architecture Guide

1.0 リファレンスアーキテクチャについて

MarkLogic リファレンスアプリケーションアーキテクチャは、MarkLogic サーバーを使用するアプリケーションの設計、開発、導入を行うアーキテクト、開発者、および管理者を対象とし、3 層（ティア）アプリケーションのテンプレートと一連のベストプラクティスを提供します。

このガイドでは、以下の内容を取り上げます。

1.1 目的

MarkLogic は柔軟で強力なプラットフォームであり、さまざまなアプリケーションソリューションで利用できます。MarkLogic リファレンスアプリケーションアーキテクチャは、MarkLogic サーバーを使用するアプリケーションの設計、開発、導入を行うアーキテクト、開発者、および管理者を対象とし、3 層（ティア）アプリケーションのテンプレートと一連のベストプラクティスを提供します。

MarkLogic リファレンスアプリケーションアーキテクチャは、以下のもので構成されています。

- アプリケーションアーキテクチャ、各ティアの役割、ティア間の関係性の説明。
- MarkLogic サーバー上に構築される信頼性の高い大規模なアプリケーション開発に関するベストプラクティス。
- オープンソースのサンプルアプリケーションであるサンプルスタック。これにより、推奨される 3 層モデルを、ベストプラクティスに基づいて最新のツールおよびフレームワークに実装する。

1.2 3 層アーキテクチャを選択した理由

従来は、エンドユーザーが目にするほとんどの web アプリケーションは、バックエンドサーバーに支えられています。つまりバックエンドのサーバーが、アプリケーションとセッション状態を管理し、ブラウザ用に HTML を作成しています。現在では、ブラウザとブラウザフレームワークの能力が向上したため、ブラウザアプリケーションが各自のビュー、セッション状態、および一部のビジネスロジックを管理する割合が高くなりました。また、アプリケーションが、統合が容易な REST API などで業務サービスを提供することも一般的になりました。ここではアプリケーションの一部を簡単に他のものと置き換えられます。

MarkLogic アプリケーションは、従来の 2 層モデルを含むさまざまな方法で構築できます。例えば、MarkLogic サーバーにはアプリケーションサーバーが含まれているため、Xquery や JavaScript を使って、MarkLogic サーバー内部でブラウザ用 HTML を生成する 2 層の MarkLogic アプリケーションを簡単に作成できます。

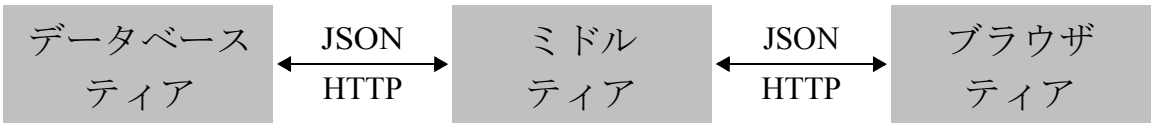
しかしながら、私たちはリファレンスアーキテクチャとしては、JSON、REST over HTTP、Java/JavaScript による 3 層モデルを選択しました。MarkLogic が初めての人にとって、このほうが短期間かつ容易に習得できるからです。MarkLogic リファレンスアプリケーションアーキテクチャの利点は以下のとおりです。

- 業界標準のフレームワークとプログラミング言語（Java、Spring、JavaScript、AngularJS、JSON）だけを使用して作業できる。
- 作業開始にあたって MarkLogic の内部構造をほとんど知らなくてもよい。
- データサービスレイヤーからビジネスロジックが分離されているため、MarkLogic を既存のエンタープライズインフラストラクチャに簡単に統合できる。

3 層アーキテクチャには、階層を個別に拡張および最適化したり、セキュリティ問題を分離できるという利点もあります。

1.3 リファレンスアーキテクチャの概要

MarkLogic のリファレンスアプリケーションアーキテクチャは、データベースティア、ミドルティア、ブラウザティアで構成される 3 層モデルです。次の図に示すように、すべての階層（ティア）間のやり取りは JSON over RESTful HTTP で行われます。



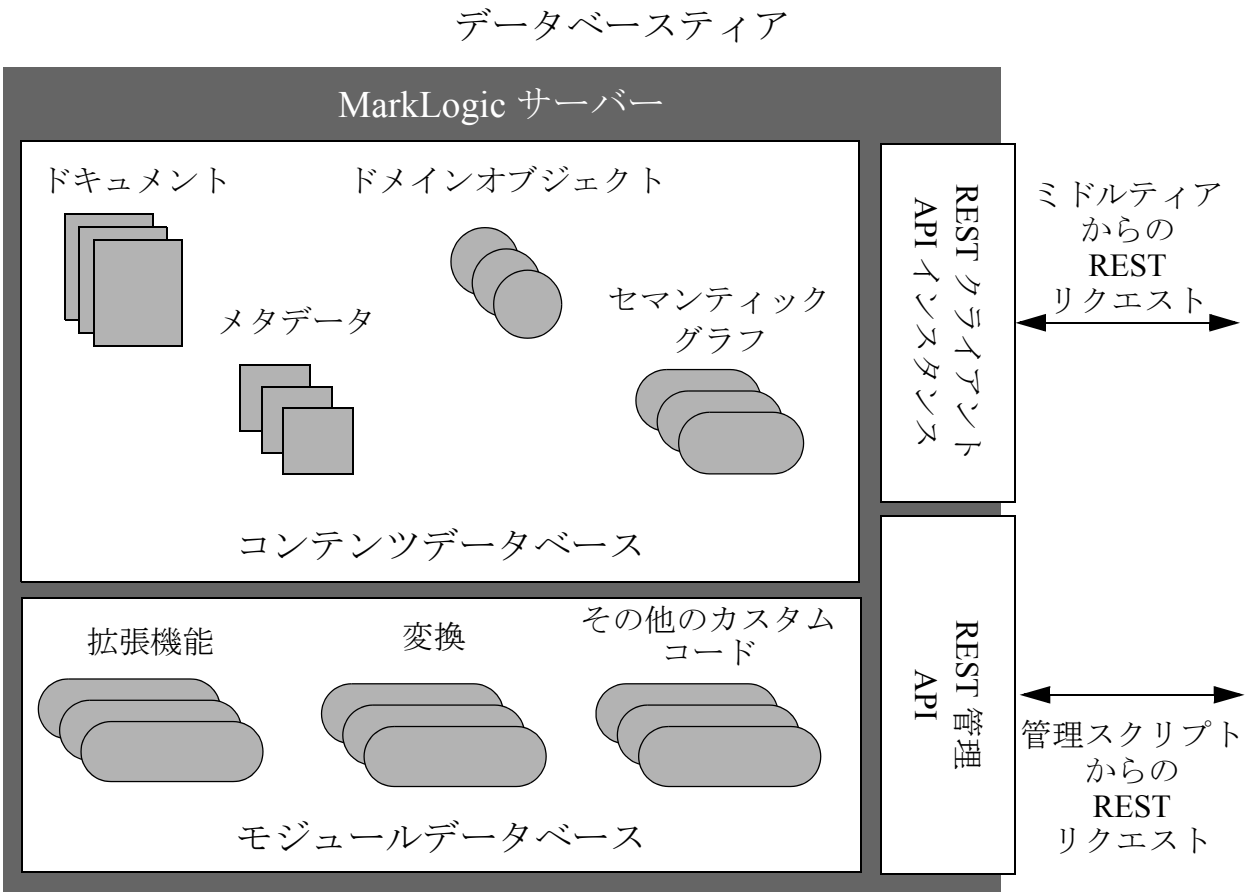
データベースティアは、長期間実行される高可用性データサービスをミドルティアに提供します。アプリケーションが必要とするすべてのデータは、MarkLogic サーバーがデータベースティアで管理します。また、スケーラビリティと簡潔性のため、永続的なアプリケーションの状態はここで管理されます。このティアには、パフォーマンス上の理由により、またはデータモデルを強制するためにデータの近くで実行する必要があるコードを含めることができます。MarkLogic では、拡張できる強力な MarkLogic クライアント API を通じてミドルティアにサービスとデータを提供します。詳細については、「データベースティア」（5 ページ）を参照してください。

ミドルティアは、ブラウザティアにデータを提供し、またセッション状態をブラウザティアと共有します。MarkLogic アプリケーションでは、ミドルティアが MarkLogic クライアント API を使用して、ブラウザティアとデータベースティア間のやり取りを行います。ミドルティアはビジネスロジックの検証を行います（レート制限や MarkLogic 以外の外部サービスとの統合など）。詳細については、「ミドルティア」（7 ページ）を参照してください。

ブラウザティアには、エンドユーザーに表示される web アプリケーションのフロントエンドがあります。このアプリケーションには、ブラウザで実行されるコード、マークアップ、スタイル（ユーザーエクスペリエンスのカスタマイズ用）が用意されています。このリファレンスアプリケーションアーキテクチャでは、ブラウザアプリケーションはリッチクライアントです。つまりブラウザティアが、ビューの編成や遷移など、UI のレンダリング処理をすべて行います。ブラウザティアは、ミドルティアとともにビジネスロジックを認識します。詳細については、「ブラウザティア」（9 ページ）を参照してください。

1.4 データベースティア

データベースティアは、データサービスとアプリケーションサービスをミドルティアに提供します。次の図は、データベースティアの主なコンポーネントを示したものです。



MarkLogic サーバーは、アプリケーションが必要とするデータとコードを管理します。次のようなものがあります。

- JSON、XML、バイナリ、テキストドキュメント
- ドキュメントに関するメタデータ（権限、クオリティ、コレクション、アプリケーション固有のドキュメントプロパティなど）
- 永続的なアプリケーション状態（ミドルティアで Java または JavaScript オブジェクトとして実現されたドメインオブジェクトなど）
- セマンティックグラフ
- スキーマ
- 拡張機能、変換、その他のアプリケーション固有のコード（パフォーマンス、カプセル化、またはデータルール強制を目的としてデータベースティアで実行する必要があるもの）

これらのアセットをデータベースに格納することで、アプリケーションにおいてトランザクションの一貫性が実現されます。MarkLogic のトランザクションモデルとして、マルチステートメントトランザクションなどがあります。これにより、アプリケーションはデータベースティア内のトランザクション処理をミドルティアおよびブラウザティア内のビジネスロジックとインターリーブできます。

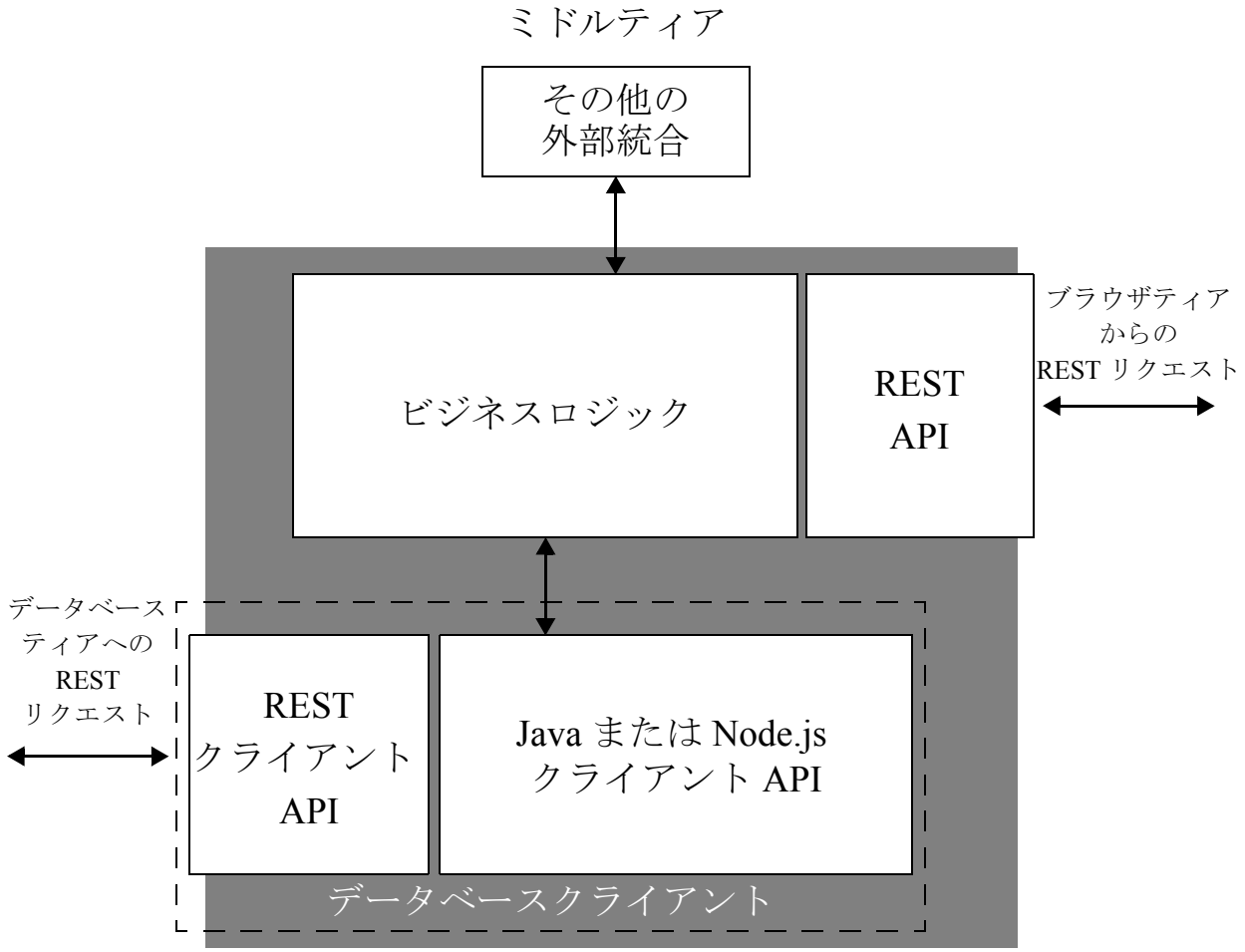
ミドルティアは、REST クライアント API および REST 管理 API を通じて MarkLogic サーバーと通信します。入ってくる REST リクエストは、MarkLogic サーバーに組み込まれている HTTP アプリケーションサーバーによって処理されます。

REST クライアント API は、一連の MarkLogic クライアント API の基盤です。アプリケーションはこれによって、データベースコンテンツの作成、読み取り、更新、削除、検索ができます。Java クライアント API と Node.js クライアント API は、この基盤を使用しています。すべての MarkLogic クライアント API は拡張可能であり、コンテンツ変換、検索のカスタマイズ、REST リソースサービス拡張などのデータベースティアライブラリモジュールを簡単にインストールして使用できます。そのようなデータベースティアコードは、モジュールデータベースに格納され、MarkLogic クライアント API で管理できます。

REST 管理 API を使用すると、REST リクエストを通じて MarkLogic サーバーの構成およびステータスをリモートから管理、監視、レビューできます。MarkLogic サーバーはインタラクティブな管理および監視用のユーザーインターフェイスを備えています。REST 管理 API を使うと、そのようなタスクのスクリプトを簡単に記述できます。

1.5 ミドルティア

ミドルティアは、ブラウザティア、データベースティア、その他の外部サービス間で、サーバー間通信を仲介します。次の図は、MarkLogic アプリケーションのミドルティアの主なコンポーネントを示したものです。



シッククライアントモデル（ビジネスロジックの大半がブラウザティアにあり、ブラウザティアとデータベースティア間をデータがほとんどまたはまったく変更されずに流れる）の場合でも、ミドルティアは次のような重要なサービスを提供します。

- ブラウザティアのサーバー間通信（ブラウザティアとデータベースティア間、またはブラウザティアとその他の外部サービス間など）を調整する。
- セキュリティサービス。ミドルティアはユーザー認証サービスを提供できます。つまり、安全ではないブラウザティアからのデータがビジネスルールに従っているかどうかを検証できます。

- ブラウザティアのネットワークトラフィックの最適化。ブラウザティア、ミドルティア、データベースティア間の通信は、比較的長いホップになるのが一般的です。このため、ブラウザからのリクエストを最小限に抑えることでパフォーマンスを向上できます。ブラウザティアは、ミドルティアとデータベースティア間での複数のリクエストを要求する 1 つのリクエストをミドルティアに対して行うことができます。
- トランザクションの一貫性。ミドルティアは、単一のトランザクションの一部である論理的には別個のアプリケーション処理を調整できます（預金の引き出しや預け入れと同時に口座残高を更新するなど）。

通常、ビジネスロジックを実装するのはリッチブラウザクライアントですが、上記のサービスをサポートするために必要に応じてミドルティアがビジネスロジックを注入（インジェクト）できます。

ミドルティアは、MarkLogic クライアント API（Java クライアント API や Node.js クライアント API など）を使用し、データベースクライアントを通じて MarkLogic サーバーと通信します。これらの API は各アプリケーションに適したスムーズなインターフェイスを提供し、HTTP 経由の REST リクエストで MarkLogic にアクセスします。つまり、これらの API ではプログラミング言語やクエリ言語を新しく習得する必要はありません。またコード文字列の安全ではない評価によるインジェクションのリスクを排除できます。

MarkLogic クライアント API では、次のことを実行できます。

- ドキュメント、メタデータ、セマンティックトリプル、ドメインオブジェクトを単独またはバッチで挿入、読み取り、更新、および削除する。
- MarkLogic の強力な検索機能を使用して、データベース内のドキュメントなどのデータにクエリを実行する。ニーズに最適なものを複数のクエリスタイルの中から選択できます。
- 複数の拡張機能ポイントを利用してビルトインのサービスを拡張する。拡張機能は、MarkLogic クライアント API で使用および管理できます。

クライアント API を通じて MarkLogic から返されるデータ構造は、JSON または XML として直接ブラウザティアに渡せます。同様に、データベースに格納されているドメインオブジェクトは、ネイティブの Java および JavaScript オブジェクトとして取得できます。

ミドルティア内のビジネスロジックは、ブラウザティアが提供するデータを検証し、他のシステムとの統合を処理します。ミドルティアは、アプリケーションに応じてブラウザティアと通信する REST インターフェイスをパブリッシュできます。この API には、さまざまなフロントエンド実装をサポートできる柔軟性が必要です。通常、ミドルティアとブラウザティアの間のデータのやり取りは、JSON 形式です。

1.6 ブラウザティア

MarkLogic は、特定の web アプリケーションアーキテクチャに依存しません。このリファレンスアーキテクチャでは、ブラウザティアがすべてのビューのレンダリングおよび一時的なアプリケーション状態を処理するシッククライアント（シングルページアプリケーション（SPA）など）となります。ただし、他のブラウザティアソリューションをアーキテクチャにプラグインすることもできます。

ブラウザティアは、アプリケーション固有の REST API が提供する JSON サービスを通じ、JavaScript 形式でミドルティアとやり取りします。ティア間のやり取りは、この REST API が提供するサービスにより、アプリケーションに適した方法でモデル化されます。この API があるため、データ変換は必須ではありません。

シッククライアントである SPA ソリューションでは、ブラウザが、MVC（モデル / ビュー / コントローラ）アプリケーションとなります。ここにはビジネスロジックがあり、またデータベースティアが公開するデータモデルに従っています。これによりブラウザティアは、（ミドルティア経由で）データベースティアからのデータを最小限の変換で利用できます。

ブラウザティアにはビジネスロジックのほとんどが実装されていますが、ブラウザは本質的にインジェクション攻撃や DoS 攻撃などのリスクに対して脆弱です。ミドルティアおよびデータベースティアは、セキュリティ、ビジネスルールおよびデータルールの検証、レート制限などの保護を提供します。

このモデルでは、JSON スキーマに対して JavaScript オブジェクトの検証を実行できます。また、ビューは HTML および CSS ベースです。

1.7 次のステップ

次の表は、MarkLogic サーバーのアプリケーション開発において推奨される文献およびチュートリアルです。

目的	参照先
リファレンスアーキテクチャの詳細について学ぶ	「サンプルスタック：リファレンスアーキテクチャのインスタンス化」（13 ページ）
リファレンスアーキテクチャに基づくフル機能の MarkLogic アプリケーションについて調べる	サンプルスタックアプリケーション（http://github.com/marklogic/marklogic-samplestack）
ドキュメントの操作、検索、セマンティックなど、MarkLogic の特定の機能について学ぶ	MarkLogic 開発者コミュニティサイトのチュートリアル（http://developer.marklogic.com/learn）
MarkLogic クライアント API の詳細について学ぶ	『*Java Application Developer's Guide*』 『*Node.js Application Developer's Guide*』 『*REST Application Developer's Guide*』
MarkLogic サーバーの内部構造について学ぶ	MarkLogic 開発者コミュニティサイトの『Inside MarkLogic Server』 http://developer.marklogic.com/inside-marklogic

2.0 推奨されるベストプラクティス

MarkLogic アプリケーションを設計する際には、少なくとも以下のベストプラクティスを開発プロセスに取り入れてください。

- 論理コンポーネントを基にプロジェクトを編成する
- コード、テスト、自動化ドライバにソースコントロールを使用する
- プロジェクトの開発と導入を構成しやすくする
- 開発、テスト、導入の各タスクを自動化する
- すべての開発フェーズに自動化されたテストを組み込む

2.1 論理コンポーネントを基にプロジェクトを編成する

プロジェクトは、アプリケーションのコンポーネントが論理的に分離されるように編成する必要があります。これにより、各ティア担当のチーム間でプロジェクトを共有しやすくなるとともに、ソースファイルなどのアセットを見つけやすくなります。

2.2 コード、テスト、自動化ドライバにソースコントロールを使用する

ソースコード、テスト、自動化スクリプトなどのアプリケーションアセットは、すべてソースコントロールで扱うようにします。ソースコントロールはアプリケーションへの変更履歴を記録するため、複数スタッフが同一プロジェクトで同時に作業できます。また、ソースコントロールですべてのアプリケーションの開発および導入コンポーネントを扱うことで、製品の各リリースの内容を簡単に把握できます。小規模なチームでもソースコントロールは役に立ちます。

2.3 プロジェクトの開発と導入を構成しやすくする

アプリケーションのビルド、テスト、および導入は、構成（設定）を中心に考えなければなりません。

例えば、テスト用インフラストラクチャを新しいホストに移動する場合、すべてのテストではなく、構成ファイル 1 つを更新するだけで済むようにする必要があります。同様に、製品ビルドの依存関係の変更は、構成ファイルの最小限の変更で実現する必要があります。

構成ファイルでそのような変更に対応できるようにしておくと、変更の際に容易にミスをトラッキングしてこれを減らせます。

2.4 開発、テスト、導入の各タスクを自動化する

プロジェクト開発には多数の反復タスクが伴います。例えば、1 日に何回も編集 / テスト / デバッグサイクルを繰り返すことがあります。そのため、一般的なタスクを自動化することで生産性および再現性が向上します。

同様に、自動化によりテストを単純化することで、開発者が頻繁にテストを行うのが楽になります。問題の診断と修正は、その原因となる要素が追加された直後に行うのが最も簡単であるため、頻繁にテストしたほうが製品の安定性が向上します。また、テストを自動化することで、テスト結果の信頼性が向上します。

アプリケーション開発および導入環境のセットアップを自動化することで、新しい開発者の追加や依存関係の管理が容易になり、安定した開発、テスト、およびリリースプラットフォームを実現できます。

2.5 すべての開発フェーズに自動化されたテストを組み込む

製品テストは、複数のレベルで行うことが重要です。例えば以下のような製品テストを用意します。

- 単体テストでは、コードの特定部分（通常は関数 / クラス / インターフェイスレベル）の機能を検証します。単体テストは、通常、開発者が作成します。単体テストは、開発者による編集 / テスト / デバッグサイクル中および各チェックイン前に簡単に実行できます。
- 結合テストでは、ブラウザティアとミドルティア間など、コンポーネント間のインターフェイスと結合ポイントを検証します。
- 回帰（リグレッション）テストでは、以前は動作していた機能が、コードの変更により意図せずに動作しなくなったことを特定します。
- システムテストでは、アプリケーションのエンドツーエンドの動作を検証します。
- スモークテスト（サニティテスト）は、最小限の運用レベルをチェックする小規模なテストセットです。例えば、スモークテストでは、多数の開発者と共有しているコードを変更する前に、プロジェクトの結合箇所をテストできます。

3.0 サンプルスタック：リファレンスアーキテクチャのインスタンス化

この章では、サンプルスタックアプリケーションの概要について説明します。サンプルスタックは、MarkLogic リファレンスアプリケーションアーキテクチャに基づいたデモアプリケーションです。

ここでは、以下の内容を取り上げます。

- サンプルスタックとは
- サンプルスタックの実装の概要
- サンプルスタックによって示されるベストプラクティス
- サンプルスタックで使用されている技術
- サンプルスタックの詳細について

3.1 サンプルスタックとは

サンプルスタックは、MarkLogic リファレンスアプリケーションアーキテクチャと、このガイドで説明しているベストプラクティスを実装したフル機能のサンプル web アプリケーションです。サンプルスタックは、学習用ツールとして開発環境に導入できます。これは、オープンソースプロジェクトとして維持されています。

サンプルスタックで使用する技術は、今後エンタープライズ web アプリケーションを開発することを前提として選択されています。ただし、アーキテクチャ、アプリケーション、またはベストプラクティスで、これら特定の技術を必ず使用しなければならない訳ではありません。

サンプルスタックでは、stackoverflow.com の無償で利用可能なデータを使用して、「Question and Answer」サイトをモデル化しています。サンプルスタックユーザーは、次のタスクを実行できます。

- 質問と回答を参照または検索する
- 質問を投稿する
- 回答を投稿する
- 質問または回答に投票する
- 回答を承認する（投稿者のみ）

質問と回答の参照および検索は、匿名でも実行できます。ただし、質問、回答、またはコメントの投稿、回答の承認、投票を実行する場合、ユーザーはサイトにログインする必要があります。投票は検索結果の順序に影響を与えます。つまり、投票が多い質問が少ないものより上に表示されます。また、質問に対する「ベスト」アンサーに選択（承認）された投稿者の評判が変わります。

MarkLogic サーバーでは、サンプルスタックのデータベースティアが提供されます。ミドルティアは、Spring Boot などの標準的な Java ライブラリをベースにした Java スタックです。ブラウザティアは、AngularJS 上に実装された SPA（シングルページアプリケーション）です。サンプルスタックで使用されているライブラリおよびフレームワークの詳細については、「サンプルスタックで使用されている技術」（27 ページ）を参照してください。

サンプルスタックプロジェクトには、「推奨されるベストプラクティス」（11 ページ）で説明しているベストプラクティスを実証する堅牢なプロジェクトインフラストラクチャがあります。詳細については、「サンプルスタックによって示されるベストプラクティス」（24 ページ）を参照してください。

サンプルスタックは、GitHub で管理されるオープンソースプロジェクトです。このプロジェクトは、http://github.com/marklogic/marklogic-samplestack でレビュー、ダウンロード、および投稿できます。

3.2　サンプルスタックの実装の概要

このセクションでは、サンプルスタックのいくつかの主な機能が MarkLogic サーバーでどのように実現されているのかを説明します。サンプルスタックでは、ここでは説明していない MarkLogic の機能も使用しています。詳細については、「サンプルスタックの詳細について」（28 ページ）を参照してください。

ここでは、以下の内容を取り上げます。

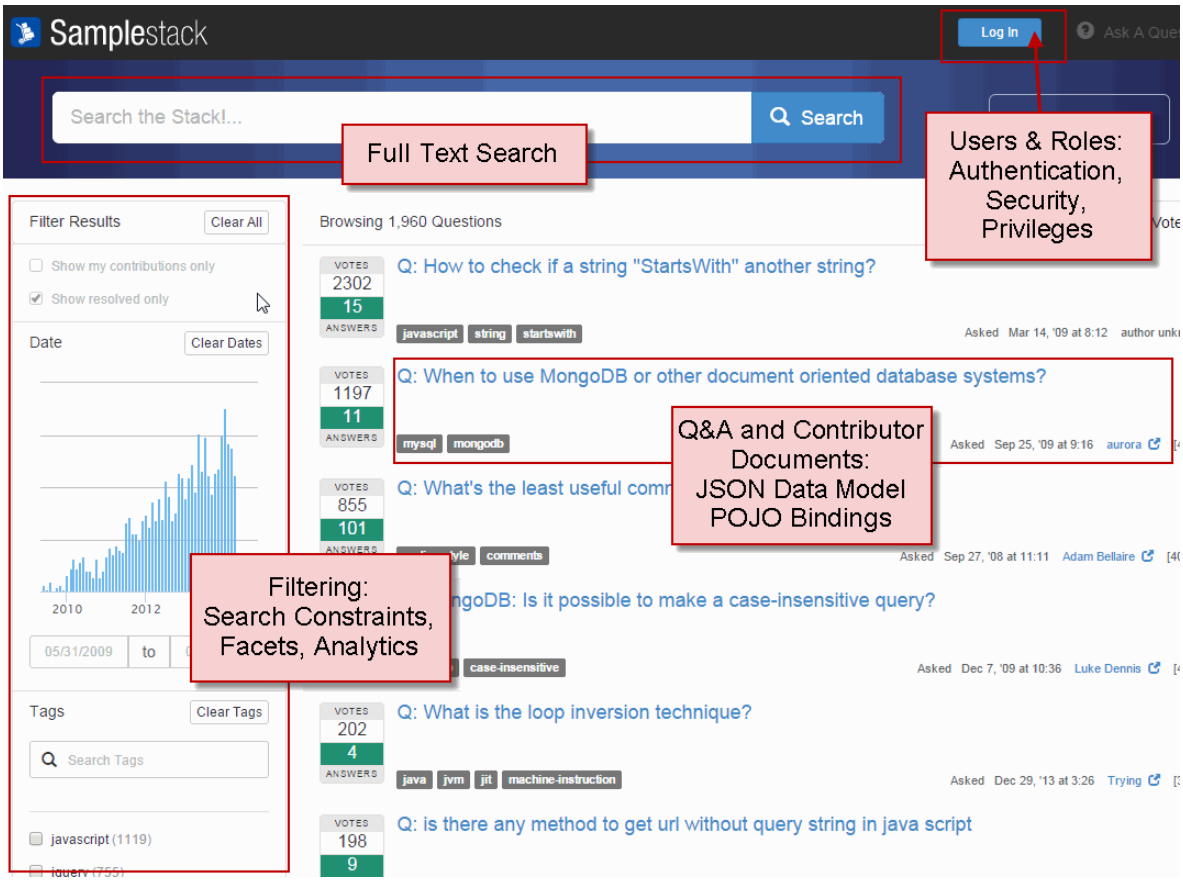
- MarkLogic 機能の UI の概要
- 全文検索
- 検索結果のフィルタリング
- ユーザーおよびロール
- ドキュメントモデル
- ドキュメントの挿入と更新
- トランザクションとデータ整合性

3.2.1 MarkLogic 機能の UI の概要

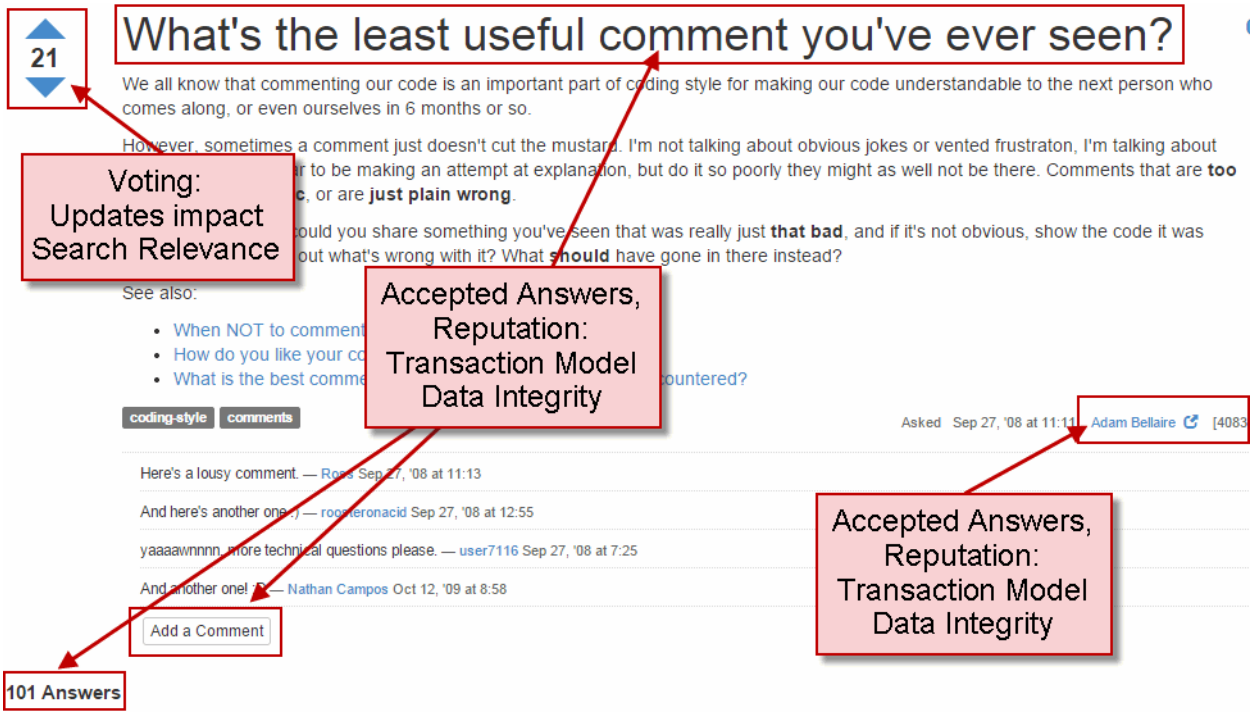
次の表は、サンプルスタックの要素と MarkLogic 機能の対応と、それを簡潔に説明している場所を示したものです。またこの表の後の図は、それぞれの機能に対応するサンプルスタック UI を示しています。

アプリケーションの機能	MarkLogic の機能	詳細情報
全文検索	全文検索、さまざまなクエリスタイル、インデックス付け	「全文検索」（17 ページ）
ファセット	検索の制約、分析	「検索結果のフィルタリング」（18 ページ）
ユーザーおよびロール	認証、セキュリティ、権限	「ユーザーおよびロール」（19 ページ）
ユーザーレコードと Q&A ドキュメント	JSON データモデル、POJO データバインド	「ドキュメントモデル」（20 ページ）
質問、回答、コメントの投稿	ドキュメントの挿入と更新	「ドキュメントの挿入と更新」（22 ページ）
回答の承認、評判	トランザクションモデル、データの整合性	「トランザクションとデータ整合性」（23 ページ）

次の図は、サンプルスタックの検索ビューの主な要素を実現する MarkLogic の機能を示したものです。このビューから、ユーザーは検索したり（フィルタあり / なし）、検索結果をレビューしたり、ログインしたり、質問したりできます（ログインしている場合）。



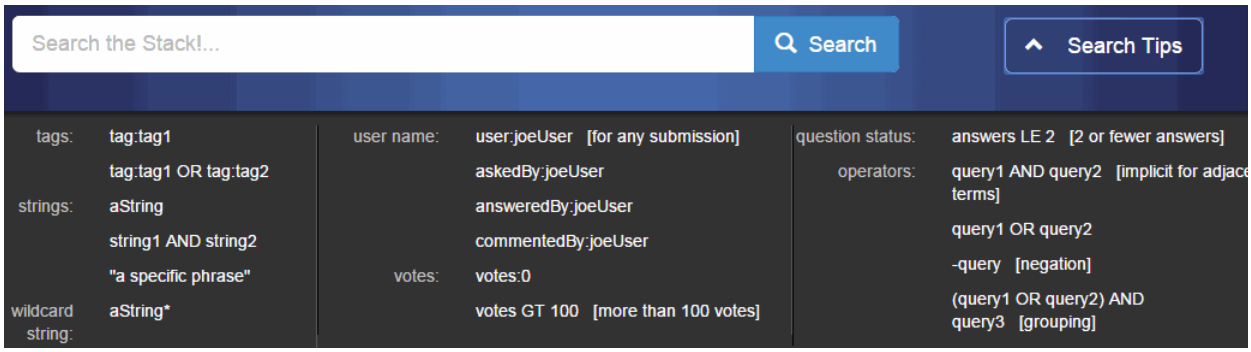
次の図は、サンプルスタックの Q&A ビューの主な要素を実現する MarkLogic 機能を示しています。このビューで、ユーザーは質問、回答、およびコメントをレビューできます。ログインしているユーザーは、回答やコメントの投稿や、質問や回答への投票ができます。また、質問の投稿者は、ベストアンサーを承認できます。



3.2.2 全文検索

各ページの上部にある全文検索ボックスでは、MarkLogic にビルトインされている文字列検索文法を使用します。この文法は、検索語修飾子と特定のデータプロパティ（ユーザー名やタグなど）とのバインドをカスタム定義することで強化されています。

［Search Tips］をクリックすると、サンプルスタックでサポートされている検索文法の要約が表示されます。

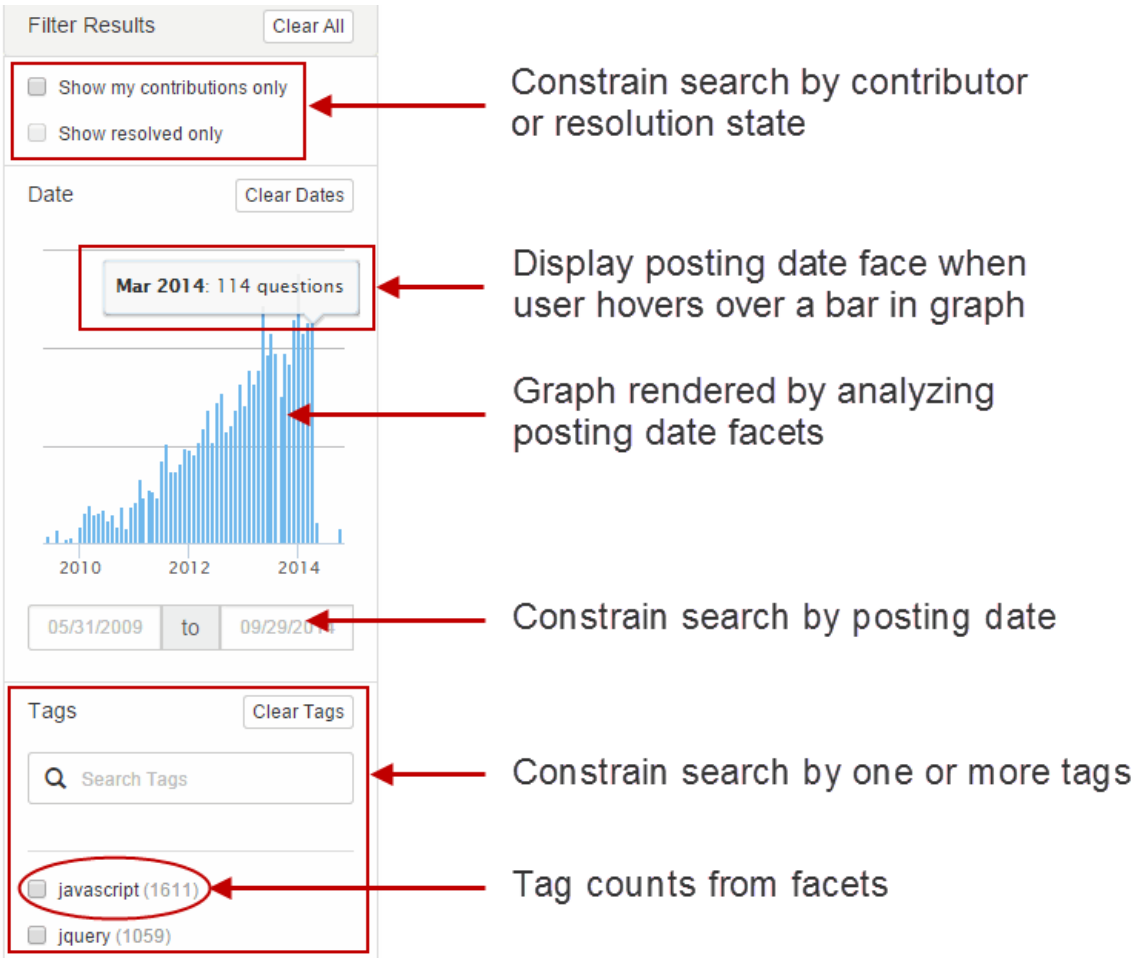


この要約にある表現はすべて、MarkLogic のデフォルトの文字列検索文法に組み込まれています。論理演算子、関係演算子、グループ化、修飾ターム（「tag:value」）などがあります。

修飾ターム（`tag`、`user`、`askedBy`、`answeredBy`、`commentedBy`、`votes`）のアプリケーション固有のプレフィックスは、名前（「answeredBy」）と、レンジインデックスが利用されているデータベースコンテンツのスライス間のバインドを表しています。MarkLogic クライアント API では、そのようなバインドを簡単に定義できます。

3.2.3 検索結果のフィルタリング

ページ左側の［Filter Results］ウィジェットを使用すると、検索結果を複数の方法で絞り込むことができます。フィルタリングのタイプは、次の図に示すとおりです。これらのフィルタが利用する検索の制約とファセット化の機能は、MarkLogic に組み込まれています。アプリケーションの方ではこの詳細を設定するだけです（制約やファセット生成の対象となるコンテンツプロパティなど）。



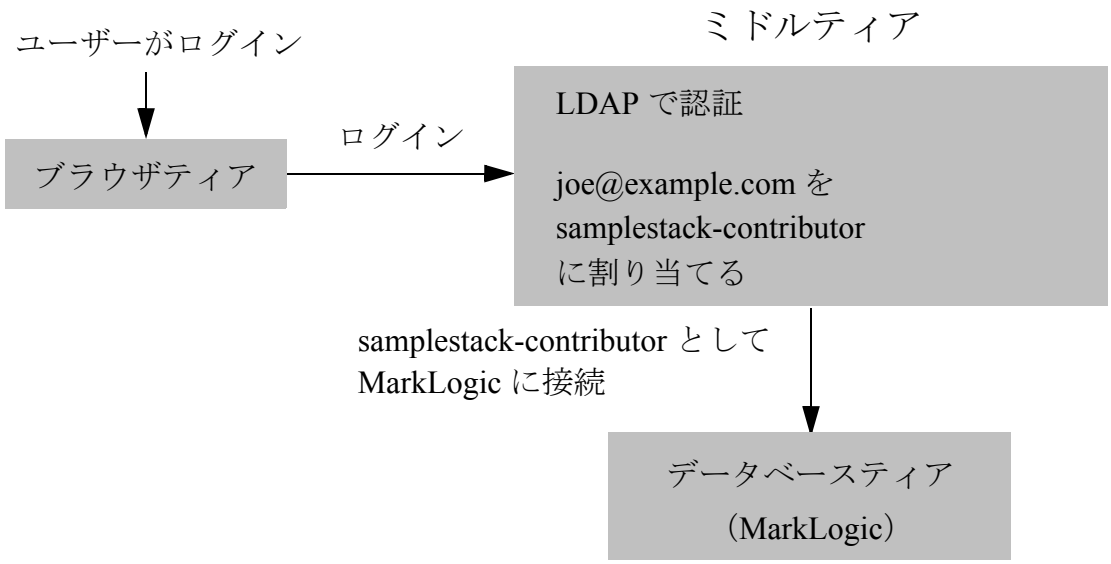
3.2.4 ユーザーおよびロール

サンプルスタックのユーザーには、次のようなロールがあります。

- Guest（ゲスト）：ログインしていないユーザー。質問と回答を見ることはできますが、投票したり、質問、コメント、または回答を投稿したりできません。また、ゲストユーザーは、まだ回答が承認されていない質問を見ることはできません。
- Contributor（投稿者）：ログインしているユーザー。投票したり、質問、コメント、および回答を投稿したりできます。また、認証された（authenticated）投稿者は、各自が投稿した質問に対するベストアンサーを承認できます。
- Administrator（管理者）：管理権限を持つユーザー。投稿者の持つ権限をすべて持っているほか、構成を変更したりアプリケーションを管理したりできます。これらの機能には、サンプルスタック UI からはアクセスできません。ただしプロジェクトインフラストラクチャでは、このロールを使用して MarkLogic サーバーを構成し、アプリケーションを導入します。

サンプルスタックをセットアップする場合、1 人の投稿者ユーザー「joe@example.com」と 1 人の管理者ユーザー「mary@example.com」が事前定義されています。資格情報の詳細については、GitHub の `marklogic-samplestack` プロジェクトを参照してください。

ユーザーがサンプルスタックにログインすると、ミドルティアは LDAP で資格情報を認証し、そのユーザーを適切な権限のある共有 MarkLogic ユーザーに割り当てます。次の図は、投稿者権限を持つユーザー「joe@example.com」について、このフローを示したものです。



サンプルスタックのセットアップの過程において、サンプルスタックのユーザーモデルをサポートする MarkLogic ユーザーとセキュリティロールが作成されます。例えば、セットアップ時に次のロールが作成されます。

Role	Compartment	Description	Roles
samplestack-guest		Role for access to limited set of questions and answers, read-only	rest-extension-user
samplestack-writer		role for contributing to samplestack corpus	rest-writer, samplestack-guest

`samplestack-writer` ロール（あるいは同等の権限）のユーザーだけが、MarkLogic クライアント API の書き込み操作を実行できます。これは `rest-writer` ロールが含まれているのは、このロールだけだからです。MarkLogic では、`rest-writer`、`rest-reader`、`rest-admin`、および `rest-extension-user` の各ロールを使用して、MarkLogic クライアント API を通じたアクセスを管理します。

また、サンプルスタックのセットアップでは、次の MarkLogic ユーザーが作成され、各ユーザータイプのアクセス権に応じて適切なロールが割り当てられます。`samplestack-guest` ユーザーには `samplestack-guest` ロールだけが割り当てられます。`samplestack-contributor` ユーザーには `samplestack-guest` ロールと `samplestack-writer` ロールの両方が割り当てられます。

User	Description	Roles
samplestack-admin	user that can administer REST server	rest-reader, rest-extension-user, manage-user, rest-admin, ...
samplestack-contributor	user for write unrestricted access to samplestack data	rest-reader, rest-extension-user, samplestack-guest, samplestack-writer, ...
samplestack-guest	user for read-only, restricted access to samplestack data	rest-extension-user, samplestack-guest

`samplestack-admin` ユーザーには、REST 管理 API を使用できるロールと、MarkLogic クライアント API の管理エンドポイントを使用できる `rest-admin` ロールが割り当てられます。そのため `samplestack-admin` ユーザーは、データベースの構成や MarkLogic 内部で実行されるあらゆるアプリケーションコードのインストールなどの操作を実行できます。

3.2.5 ドキュメントモデル

このサンプルスタックのデータは、次の 2 種類の JSON ドキュメントで構成されています。

- Q&A ドキュメント。各ドキュメントは、質問とその回答、およびコメントで構成されています。
- 投稿者ドキュメント。アプリケーションに登録されている各ユーザーには対応する投稿者ドキュメントがあります。これには表示名、投票、および評判などの情報が記録されています。

投稿者ドキュメントは、サンプルスタックのドメインオブジェクトを表現しています。つまり、ミドルティアでは、MarkLogic Java クライアント API の POJO バインド機能を使用して、Java クラス `com.marklogic.samplestack.domain.Contributor` として直接投稿者データを操作します。

Q&A ドキュメントは、投稿者ドキュメントよりもサイズが大きく複雑です。このため、Q&A ドキュメントに対する変更は、MarkLogic Java クライアント API のドキュメントパッチ機能を通じて行います。詳細については、「ドキュメントの挿入と更新」（22 ページ）を参照してください。

各投稿者の評判は、投稿者ドキュメントに格納されます。ただし、質問をすべての回答およびコメントとともに表示する場合、投稿者の評判がユーザー情報の一部として表示されます。例えば、次の図のように投稿者情報が回答とともに表示されます。588 がこのユーザーの評判です。

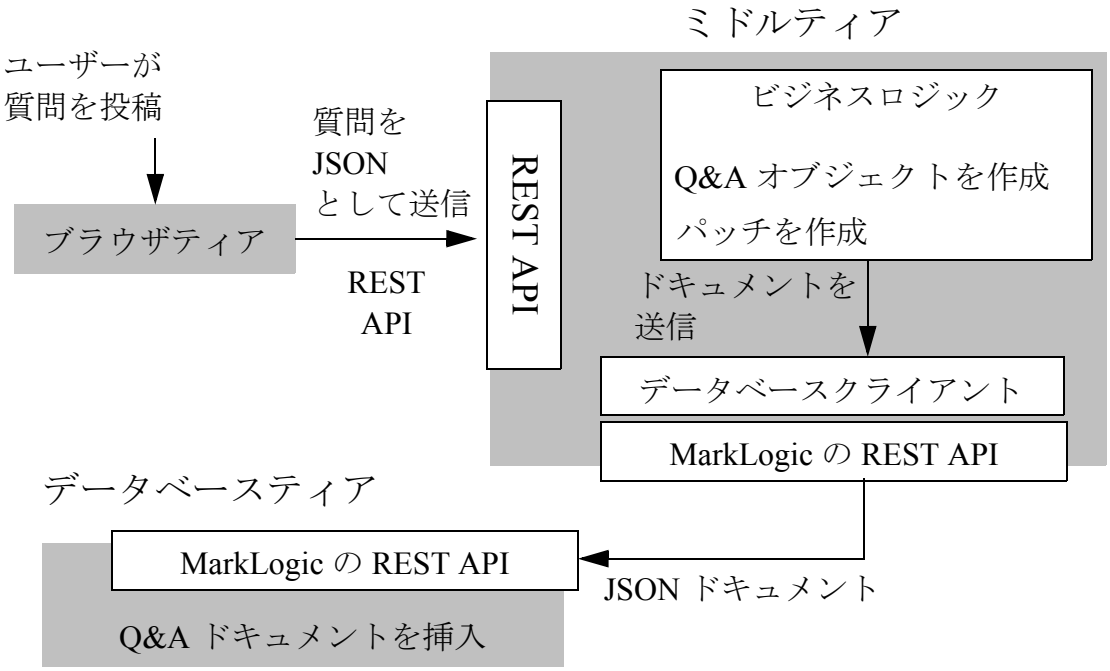
結合はデータの近くで実行する必要があるため、MarkLogic にインストールされているアプリケーション固有の変換は、特定の Q&A ドキュメントのリクエストを受け取ったときに投稿者ドキュメントの評判データを Q&A データに結合します。同様の結合は、検索結果の取得時に実行されます。

ユーザーが質問、回答、およびコメントを投稿したときにブラウザティアから受け取る JSON は、ミドルティアによって最小限変更されています。この変更は、その後ブラウザティアでのフェッチで利用できるようにするためです。これにより非常に効率的にデータベースからデータをフェッチしてブラウザティアに戻すことができます。

3.2.6 ドキュメントの挿入と更新

投稿者が質問を投稿すると、Q&A ドキュメントがデータベースに挿入されます。ブラウザティアは、アプリケーション固有の REST API を使用して、投稿者と質問を JSON としてミドルティアに送信します。ミドルティアは情報を追加し、ドキュメントのデータベース URI を生成します。また完成した Q&A 構造を、MarkLogic クライアント API によってデータベースティアに送信します。

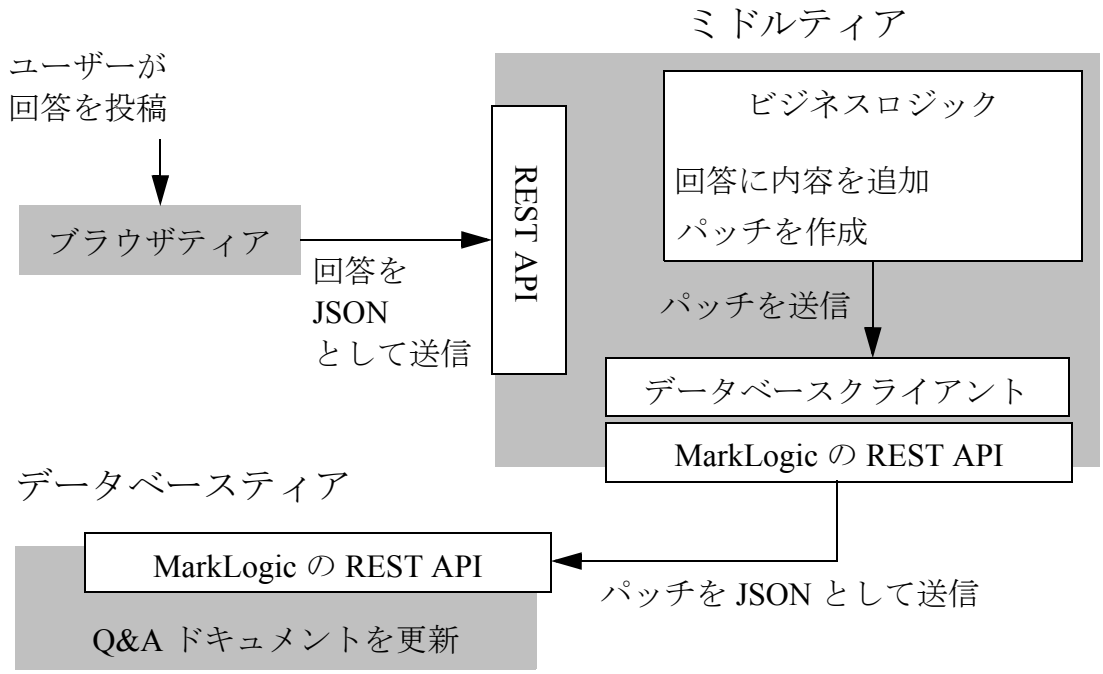
次の図は、質問の投稿フローを示したものです。



ユーザーが質問に回答すると、データベース内の対象 Q&A ドキュメントに回答が追加されます。質問、その回答、および関連するコメントは、単一のドキュメントに格納されることを思い出してください。

このフローでは、ブラウザティアはアプリケーション固有の REST API を使用して、回答者、質問 ID、および回答を JSON 形式でミドルティアに送信します。ミドルティアは回答に情報を追加し、その回答を MarkLogic クライアント API のパッチ機能を使用して Q&A ドキュメントに挿入します。この更新において、Q&A ドキュメントをデータベースティアからフェッチする必要はありません。コメントの投稿も同じようなフローになります。

次の図は、回答の投稿フローを示したものです。



3.2.7　トランザクションとデータ整合性

回答を承認する際には、Q&A ドキュメントと投稿者ドキュメントの両方を更新する必要があります。投稿者が各自の質問に対する回答を「ベスト」アンサーとして承認すると、これを反映して Q&A ドキュメントが更新されます。同時に、回答を投稿したユーザーの評判の値が 1 つ増えます。

Q&A ドキュメントと投稿者の評判の更新は、アトミック（不可分）な操作でなければなりません。評判が更新されることなく回答が承認されてはいけません。また、回答が承認されることなく評判が加算されてもいけません。

データの整合性は、MarkLogic のマルチステートメントトランザクションによって保持されます。ミドルティアは、回答の承認をブラウザティアから受け取ると、データベースクライアントに対して MarkLogic サーバー上にマルチステートメントトランザクションを作成するように要求します。データベースクライアントはトランザクション ID を返します。このトランザクション ID を質問および投稿者の更新操作に含めることで、ミドルティアは両方の操作が同一トランザクションの一部であることを保証します。

両方の更新操作が正常に完了すると、ミドルティアはトランザクションをコミットします。ただし、いずれかの操作が失敗した場合、ミドルティアはこのトランザクションをロールバックします。

3.3 サンプルスタックによって示されるベストプラクティス

サンプルスタックのプロジェクトインフラストラクチャは、MarkLogic リファレンスアプリケーションアーキテクチャによって推奨される次のベストプラクティスを実証するものです。

- プロジェクトの編成
- ソースコントロールと問題のトラッキング
- 構成と依存関係の管理
- タスクの自動化
- テスト

3.3.1 プロジェクトの編成

github で `marklogic-samplestack` プロジェクトの編成を確認すると、プロジェクトの最上位レベルが各ティアのフォルダと共有アセットのフォルダに分割されていることがわかります。

```
marklogic-samplestack/
  appserver/
  browser/
  database/
  shared/
```

各ティアのフォルダには、そのティアをビルド、テスト、および導入するのに必要なソースコード、テスト、構成ファイル、自動化ドライバが含まれています。プロジェクトの編成の詳細を確認するには、次の URL を参照してください。

http://github.com/marklogic/marklogic-samplestack

3.3.2 ソースコントロールと問題のトラッキング

サンプルスタックプロジェクトでは、ソースコントロールの管理に GitHub と `git` を使用しています。大規模なサンプルデータセットを除き、アプリケーションをビルド、テスト、および導入するのに必要なすべてのアセットはソースコントロール下にあります。

GitHub には問題のトラッキングシステムが用意されており、サンプルスタックでも、そのシステムを使用して不具合、タスク、および未解決の設計上の問題をトラッキングします。

3.3.3　構成と依存関係の管理

サンプルスタックプロジェクトでは、ビルド、テスト、および導入を行うのに構成プロパティファイルを多用します。

例えば、`marklogic-samplestack/appserver/java-spring/gradle.properties` には、MarkLogic ユーザー名や、MarkLogic 接続に使用するホストおよびポート情報など、Java ミドルティアが MarkLogic サーバーに接続するのに必要な構成プロパティが記述されています。これらと同じプロパティは、データベースティアとテストのセットアップにも使用されます。このため、この 1 つのファイルを変更してセットアップスクリプトを再実行するだけで、別の MarkLogic インスタンスを使用したり、別のユーザーとして接続したりできます。

GitHub 上のこのファイルを表示するには、次の URL にアクセスします。

> http://github.com/marklogic/marklogic-samplestack/blob/master/appserver/java-spring/gradle.properties

同様に、`marklogic-samplestack/appserver/java-spring/build.gradle` には、サンプルスタックのミドルティアが必要とするすべての外部依存関係が記述されています。また、ブラウザティアは同様の依存関係のリストを `marklogic-samplestack/browser/bowser.json` に保持します。

データベースティアでは、MarkLogic データベース構成の詳細は `marklogic-samplestack/database/database-properties.json` に保持されています。これは、MarkLogic REST 管理 API で MarkLogic に直接渡すのに適した形式になっています。

3.3.4　タスクの自動化

サンプルスタックのセットアップ、ビルド、導入、およびテストは完全に自動化されています。ティアの要件はそれぞれ異なるため、使用する自動化ツールは異なります。

データベースティアとミドルティアの Java 実装では、ビルド自動化ツール `gradle` を使用します。サンプルスタックの `gradle` 構成では、ミドルティアの構築とテストに必要な次のような各種タスクを定義します。

- サンプルスタックが必要とする MarkLogic ロール、ユーザー、REST API インスタンス（アプリケーションサーバーおよびデータベース）を作成する。
- データベースインデックスなどのプロパティを構成する。
- MarkLogic サーバー上のモジュールデータベースにデータベースティアのサンプルスタックアセットをロードする。このようなアセットとしては、リソースサービス拡張、コンテンツ変換、永続的なクエリオプションなどが挙げられます。
- データベースにシードデータをロードする。

- ミドルティアの Java 実装をビルドする。
- ミドルティアを起動する。
- ミドルティアをテストする。
- MarkLogic の終了 - REST API インスタンスおよびデータベースを除去する。
- これらのすべてのステップを 1 ステップにまとめ、クリーンインストールされた MarkLogic をスモークテスト済みの実行可能なサンプルスタックミドルティアにする包括的タスク。

詳細については、次のページを参照してください。

http://github.com/marklogic/marklogic-samplestack/tree/master/appserver/java-spring

ブラウザティアは JavaScript および Node.js に実装されているため、同様の目的であっても `npm`、`bower`、および `gulp` などの異なるツールを使用します。詳細については、次のページを参照してください。

http://github.com/marklogic/marklogic-samplestack/tree/master/browser

3.3.5 テスト

ブラウザティアとミドルティアには、互いに独立して実行できる一連の単体テストがそれぞれ用意されています。これらのテストには、隣接するティアのモックアップを作成するスキャフォールディングが含まれます。

例えば、ミドルティアには、ミドルティアのスタートアップタスク（`appserver` および `assemble` gradle タスク）の一部として実行される Java 実装用の JUnit 駆動単体テストが用意されています。これにより、変更を導入するたびにサニティチェックが提供されます。

ミドルティアには、ティアどうしのやり取りを個別にテストする結合テストも用意されています。つまり、ミドルティアとデータベースティア間のやり取り、およびミドルティアとブラウザティア間のやり取りをテストします。

ブラウザティアのテストには、アプリケーションスタック全体についてテストを実施するエンドツーエンドのテストが用意されています。このようなテストでは、ブラウザを駆動するのに Selenium を使用し、`gherkin`、`cucumber`、`protractor` などのテストツールを使用して動作と想定される結果を定義します。詳細については、次のページのテストに関するセクションを参照してください。

http://github.com/marklogic/marklogic-samplestack/tree/master/browser

サンプルスタックでのテストについてさらに詳細な情報を確認するには、このプロジェクトの次の部分を参照してください。

- `marklogic-samplestack/appserver/java-spring/src/test`
- `marklogic-samplestack/browser/test`

3.4 サンプルスタックで使用されている技術

このセクションでは、サンプルスタックの実装、ビルド、テスト、および導入に使用されるいくつかの技術について説明します。業界標準の技術を使用して、推奨されるベストプラクティスの実装や、MarkLogic の開発プロセスへの統合が簡単にできます。

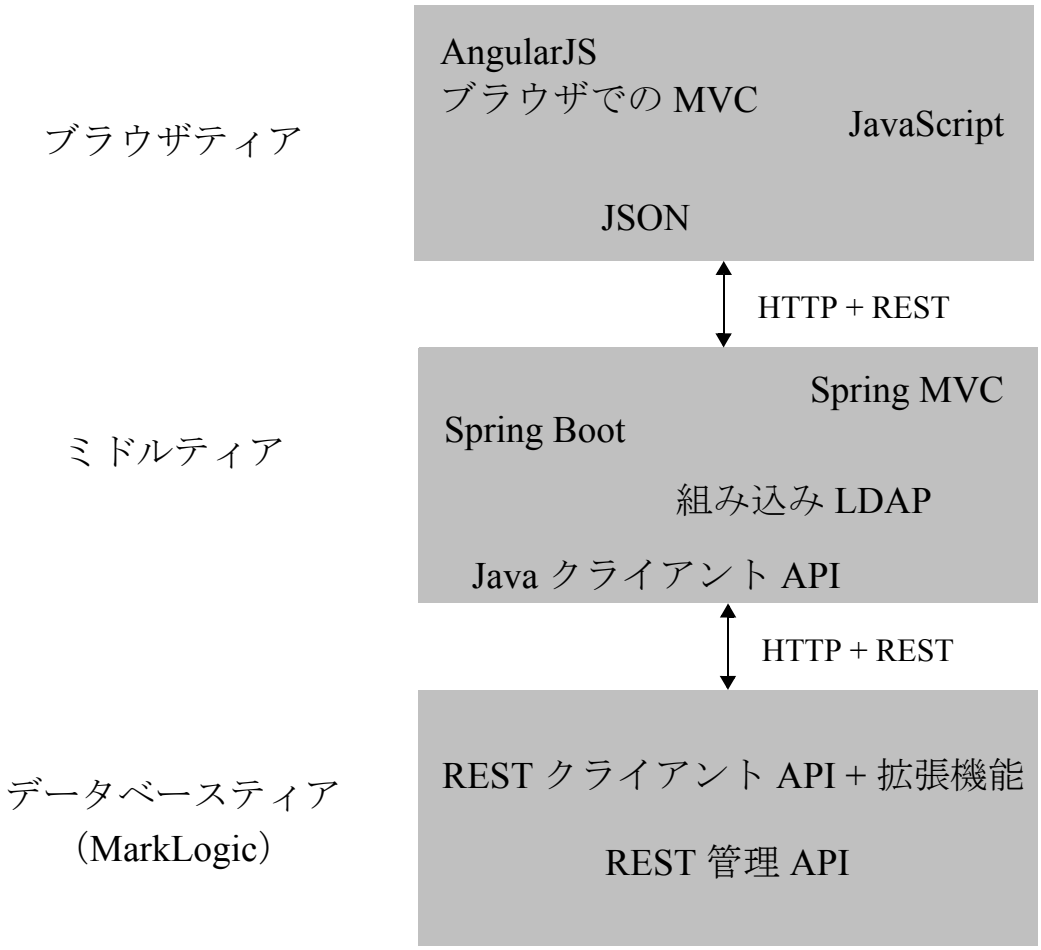
アプリケーションで MarkLogic を使用するのに、これと同じツールとフレームワークを使用する必要はありません。

このセクションでは、以下の内容を取り上げます。

- サンプルスタックの実装
- サンプルスタックのビルド、テスト、導入の自動化

3.4.1 サンプルスタックの実装

サンプルスタックの実装には、MarkLogic の主要な機能に加え、次の図に示すような技術を使用します。



3.4.2 サンプルスタックのビルド、テスト、導入の自動化

サンプルスタックのプロジェクトインフラストラクチャでは、ソースコード管理、自動化されたビルド、自動化された単体、結合、およびシステムテストなどの開発のベストプラクティスと構成主導のアプリケーション開発を示します。

サンプルスタックの構成、ビルド、テスト、および導入には、次のツールと技術を使用します。

- ソースコントロールと問題のトラッキング：GitHub
- ビルド、テスト、および導入プロセスの自動化：gradle、gulp、bower、npm
- テスト：JUnit、Selenium、mocha、PhantomJS、cucumber、protractor、gherkin
- アプリケーションの構成と導入：MarkLogic REST 管理 API、mlcp（MarkLogic content pump）

3.5 サンプルスタックの詳細について

サンプルスタックを試したり、実装の詳細を調べたりするには、GitHub で `marklogic-samplestack` プロジェクトを参照してください。

http://github.com/marklogic/marklogic-samplestack

プロジェクトページには、アプリケーションのビルドとセットアップの手順も示されています。またプロジェクトの wiki には、実装、テスト、およびツールに関する詳細情報が掲載されています。