Unit 1

MarkLogic in a Three Tiered Architecture

MarkLogic

Learning Objectives

- Install MarkLogic and describe lab environment dependencies.
- Describe the goals, uses, and functionality of Samplestack.
- Describe the reference architecture for Samplestack.
- Run Samplestack on the middle tier and browser tier.
- Explore Samplestack as an end user.
- Explore helpful resources for the Node.js Developer.

MarkLogic

Samplestack

- An end-to-end three-tiered application with implementations in Java and Node.js
- Encapsulates best practices and introduces key MarkLogic concepts
- Use sample code as a model for building applications more efficiently
- Modern technology stack demonstrates how MarkLogic can fit in your existing Enterprise environment

MarkLogic

Samplestack unlocks MarkLogic's power

- This unit will explore access methods to MarkLogic data via the fluent Node.js API.
- This unit will emphasize the advantages of tiered services, particularly the role of the middle tier for authentication and business services, and will highlight the interactions of the middle tier with the front-end and the database.
- This unit will show you how to make data modeling easier using relationships and native document formats (JSON, POJOs and RDF in addition to XML).
- This unit will demonstrate MarkLogic's built-in search and indexing capabilities at work in a realistic enterprise application.

MarkLogic

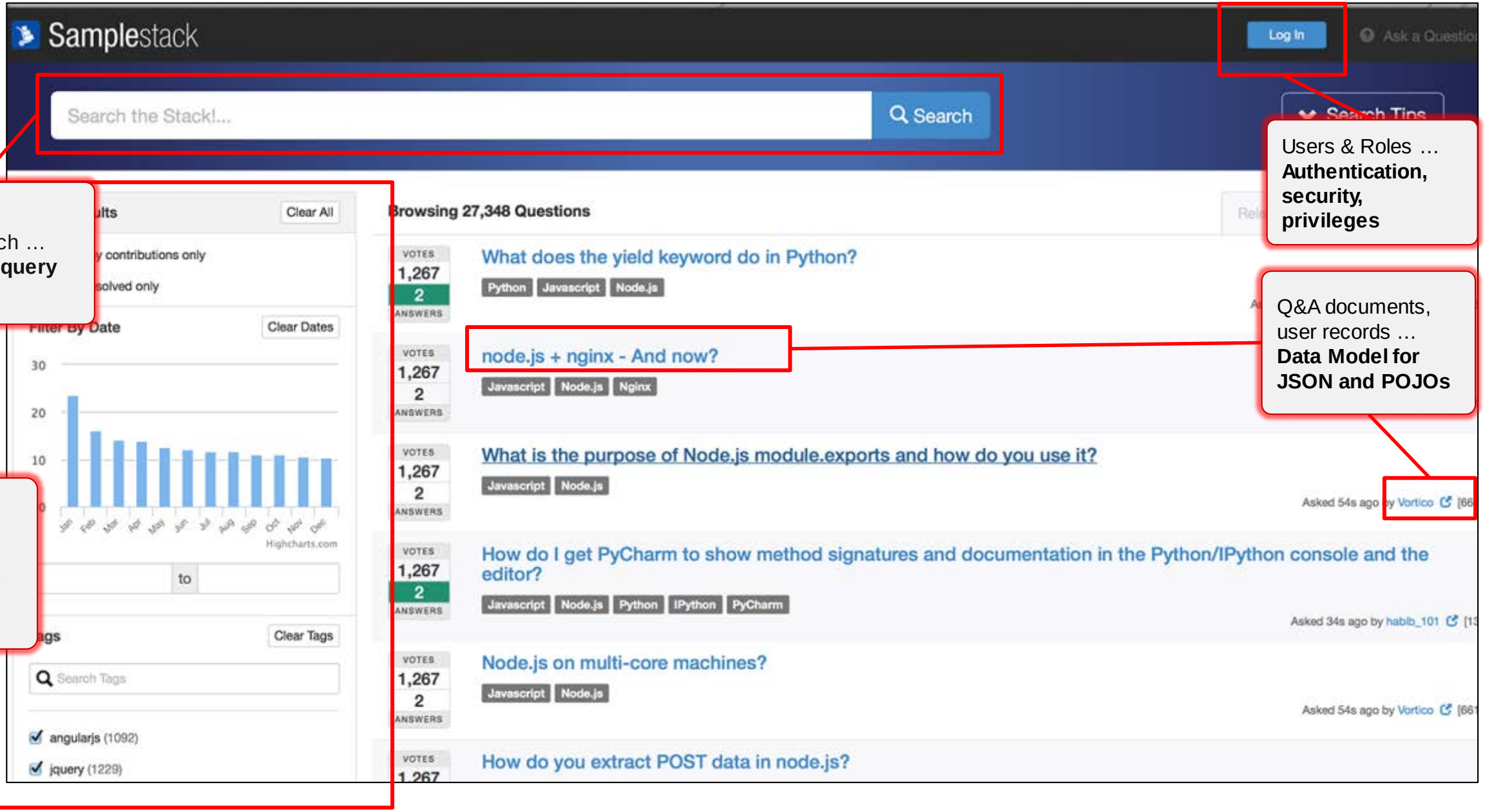
Every feature teaches a MarkLogic concept

Application Feature	MarkLogic Concept
Full-text Search	→ Indexes, query styles
Facets	→ Search constraints, analytics
Users & Roles	→ Authentication, security, privileges
Restricted Content	→ Document permissions
Ask, Answer, Comment	→ Document updates
Voting	→ Updates impact search relevance
Accepted Answers & Reputation	→ Transactional model, data integrity
Related tags	→ Semantics
User records and Q&A documents	→ Data Model for JSON and POJOs

MarkLogic

GitHub Development

- https://github.com/marklogic/marklogic-samplestack
- Open Development on GitHub
  - Transparency!
- What does this mean for external involvement?
  - Responsive!
  - We are set up to receive contributions (CLA) – internal and external
- Changes Engineering/Product relationship with outside participants



Full-text Search …
Indexes and query

Facets…
Search constraints, analytics

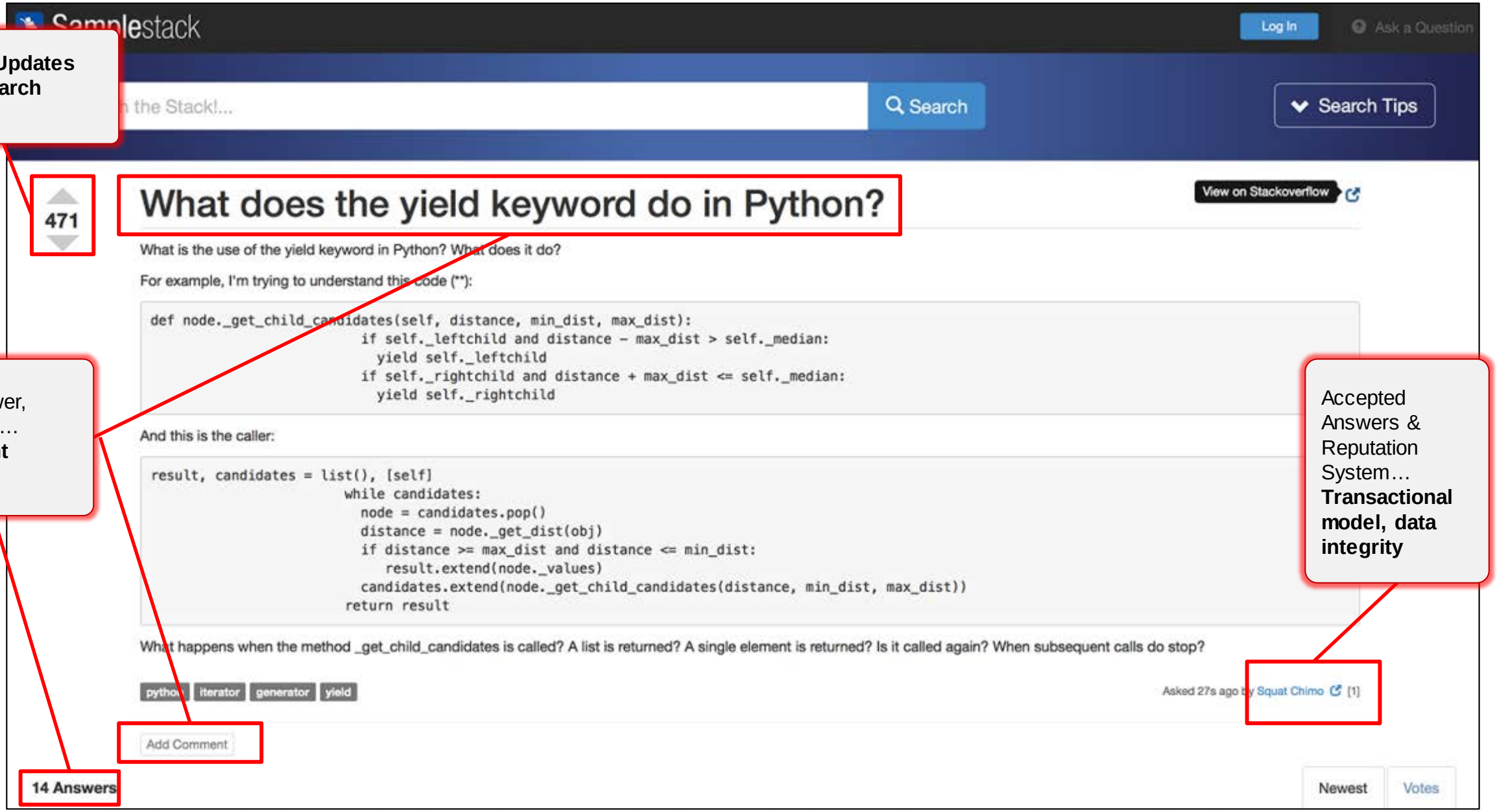
Users & Roles …
Authentication, security, privileges

Q&A documents, user records …
Data Model for JSON and POJOs

Voting … **Updates impact search relevance**

Ask, Answer, Comment… **Document updates**

Accepted Answers & Reputation System… **Transactional model, data integrity**



MarkLogic

User Interface

Data views, user workflow

Middleware

Business rules, domain model, integration

Database

Persistent state, stored procedures

Business Services

- Resources (Customer, Approval, etc.)
- JSON over HTTP

Data Services

- Documents, collections, elements
- JSON/XML over HTTP

MarkLogic

Service Layer

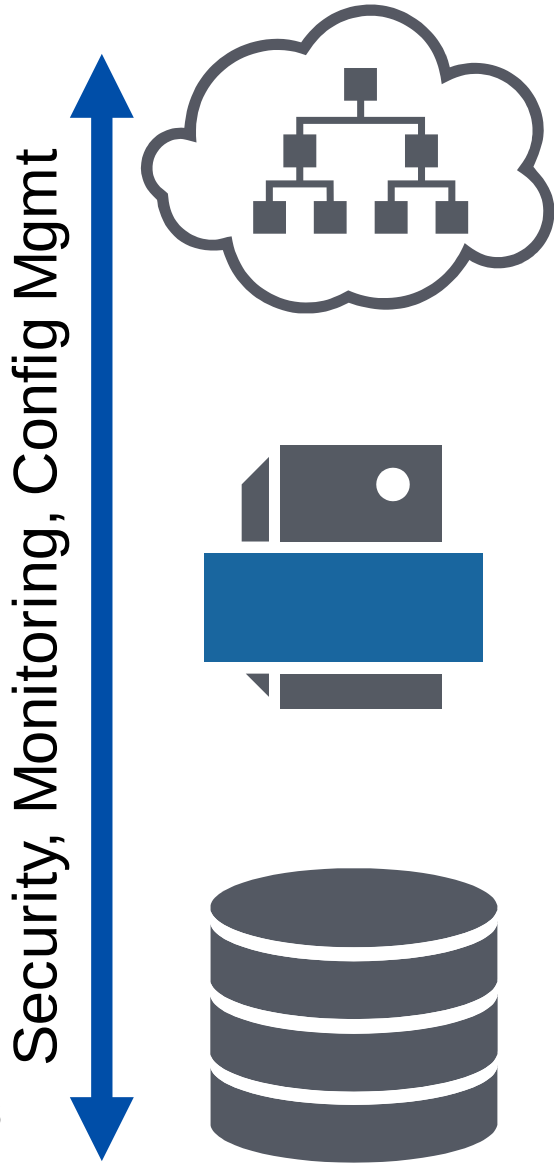
Orchestration, messaging

Middleware

Business rules, domain model, integration

Database

Persistent state, stored procedures



Business Services

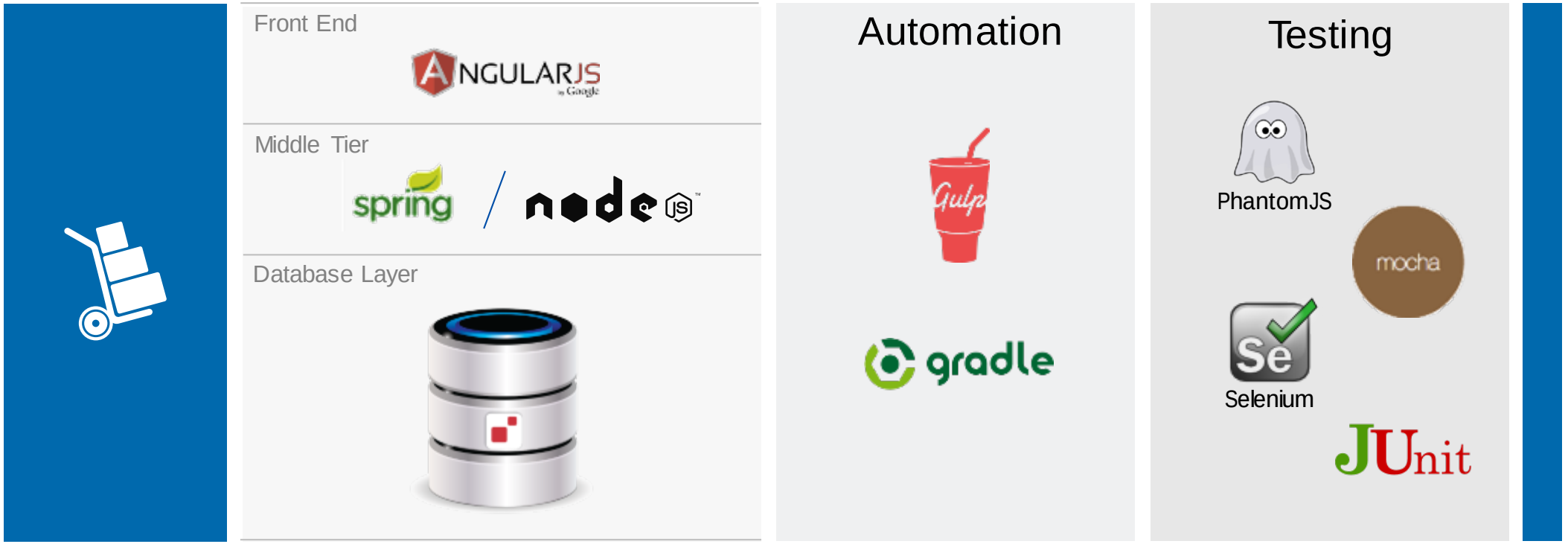
- Resources (Customer, Approval, etc.)
- JSON over HTTP

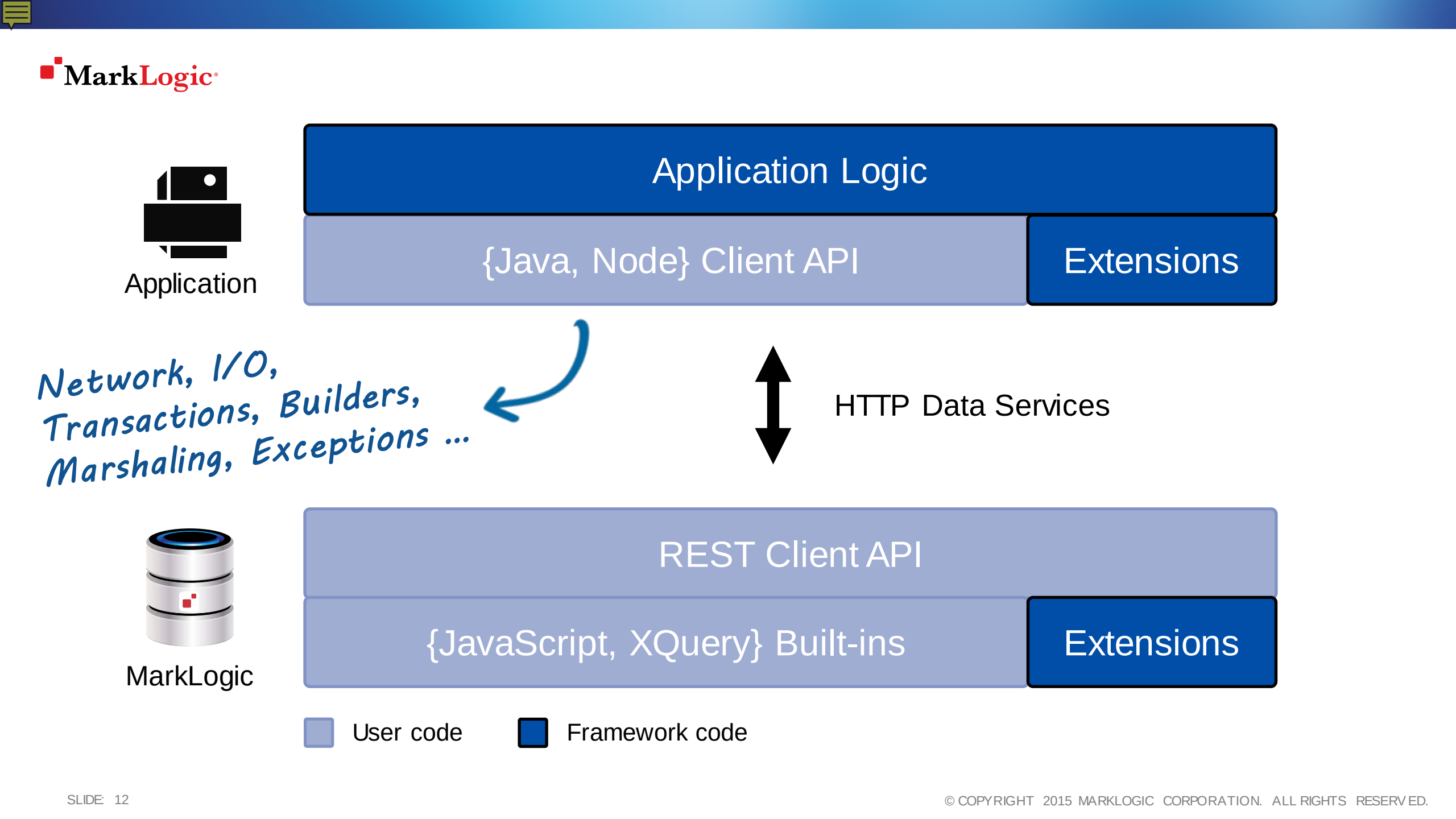
Data Services

- Documents, collections, elements
- JSON/XML over HTTP

MarkLogic

Technology Stack





MarkLogic

Node.js API Key Capabilities

- Bulk write and read
- Patch updates
- Extensions
  - Resource
  - Transformation
  - Eval / Invoke
- Semantics
- Search
  - Query
    - By example
    - Structured
    - String
  - Projection
    - Snippets
    - Sorts
    - Pagination
    - Highlights

MarkLogic

Deploy in every environment

- Increase flexibility by reusing existing skills, tools
- Minimize integration costs with a pure Node.js interface
- Maximize performance by bringing code to the data
- Scale up (or down) without modifying application code
- Build, test, instrument, debug with standard tools

MarkLogic

Node.js API Documentation

- MarkLogic product documentation:
  - docs.marklogic.com
- MarkLogic Node.js Developer Guide:
  - http://docs.marklogic.com/guide/node-dev
- MarkLogic JSdoc:
  - https://www.npmjs.com/package/marklogic
- General Node.js API Documentation
  - https://nodejs.org/api/

MarkLogic

The Lab Environment: What's Been Setup For You

- MarkLogic installed, service started (developer.marklogic.com)
- Atom editor installed (free, great editor for writing code)
- Node.js installed (nodejs.org, v0.10.33)
- NPM installed (npmjs.org, v2.1.11)
- Gulp installed (npm install gulp -g)
- Samplestack project has been cloned & setup as documented here:
  - https://github.com/marklogic/marklogic-samplestack

Demo: Samplestack Introduction

Labs: Unit 1

Exercise 1: Launch Samplestack
Exercise 2: Explore Samplestack
Exercise 3: Explore Node.js Client API Resources
Exercise 4: Create a Simple Node.js App
Exercise 5: Create a Simple Front-End

Unit Review Question 1:

Samplestack is a reference application that shows how to use MarkLogic and the Node.js client API to perform:

1. Full text and semantic search
2. Document updates
3. Document security
4. Automated configuration and deployment
5. All of the above

Unit Review Question 1:

Samplestack is a reference application that shows how to use MarkLogic and the Node.js client API to perform:

1. Full text and semantic search
2. Document updates
3. Document security
4. Automated configuration and deployment
5. **All of the above**

Unit Review Question 2:

AngularJS must be used as the front-end technology to build a 3 tier application using MarkLogic and Node.js:

1. True
2. False

Unit Review Question 2:

AngularJS must be used as the front-end technology to build a 3 tier application using MarkLogic and Node.js:

1. True
2. **False**

Unit Review Question 3:

In order to communicate between a Node.js application and the MarkLogic database you must have:

1. A MarkLogic REST instance
2. The Atom editor installed
3. Performed an “npm install marklogic” for your project
4. Both #1 and #3
5. All of the above

Unit Review Question 3:

In order to communicate between a Node.js application and the MarkLogic database you must have:

1. A MarkLogic REST instance
2. The Atom editor installed
3. Performed an “npm install marklogic” for your project
4. **Both #1 and #3**
5. All of the above

Unit Review Question 4:

In a 3 tiered architecture with MarkLogic as the database, you can write server side code and extensions using:

1. PHP
2. Java
3. JavaScript
4. Python

Unit Review Question 4:

In a 3 tiered architecture with MarkLogic as the database, you can write server side code and extensions using:

1. PHP
2. Java
3. **JavaScript**
4. Python