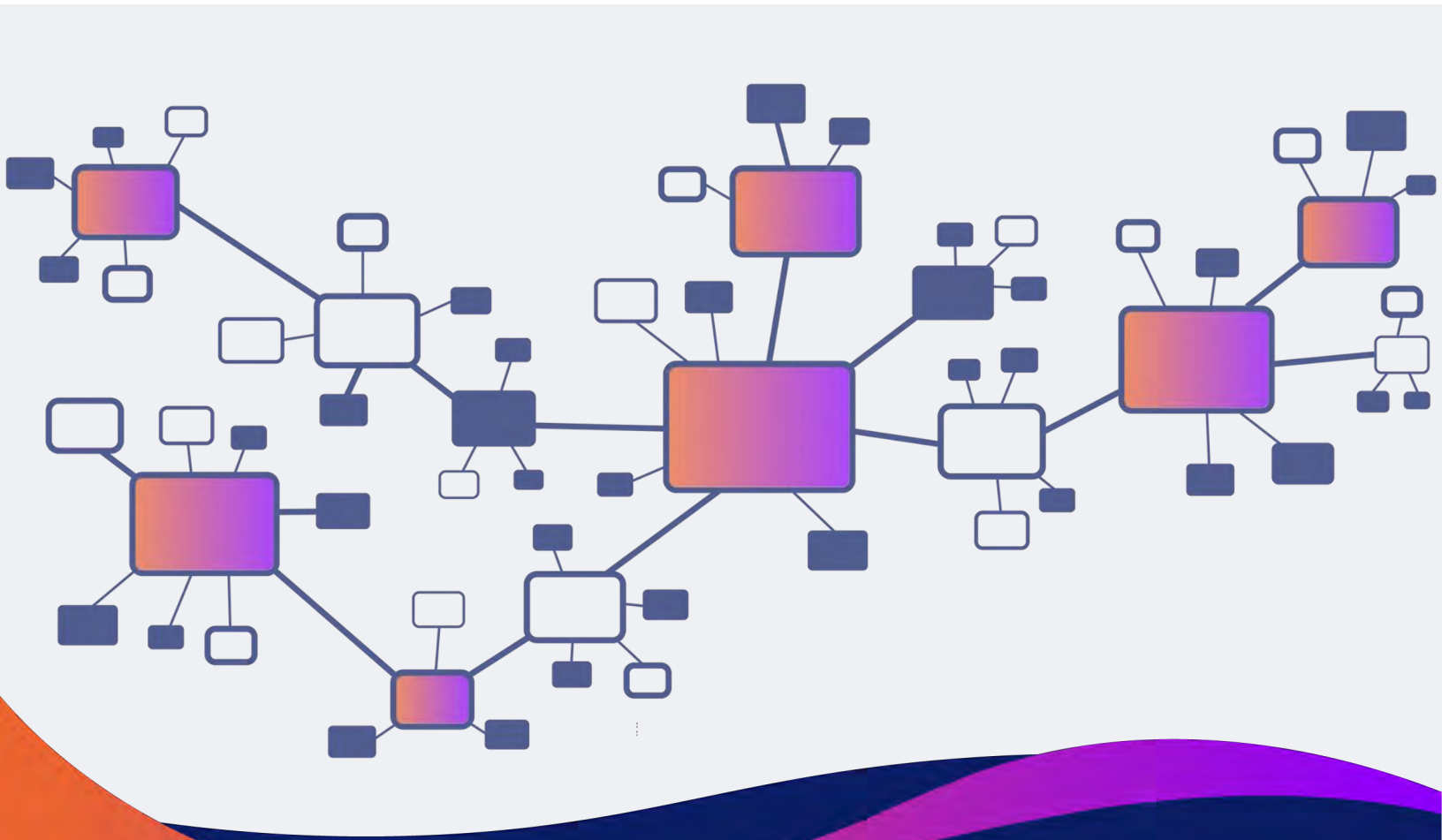


5 Answers for the Developer



The Challenge

New application updates, the move to microservices, the push to eliminate technical debt in old code – Dev teams are updating application code every day. All these changes make it difficult to understand how all the pieces of an application fit together. To make matters worse, application development teams rely on outdated or un-documented knowledge to help them decipher what's in the code and how it works. This lack of visibility doesn't provide dev teams with what's needed to get the job done right.

More than ever, IT executives and their teams need a clear understanding of their application connections and dependencies to be more productive, better understand code complexity, and make informed application strategy decisions.

Our Solution

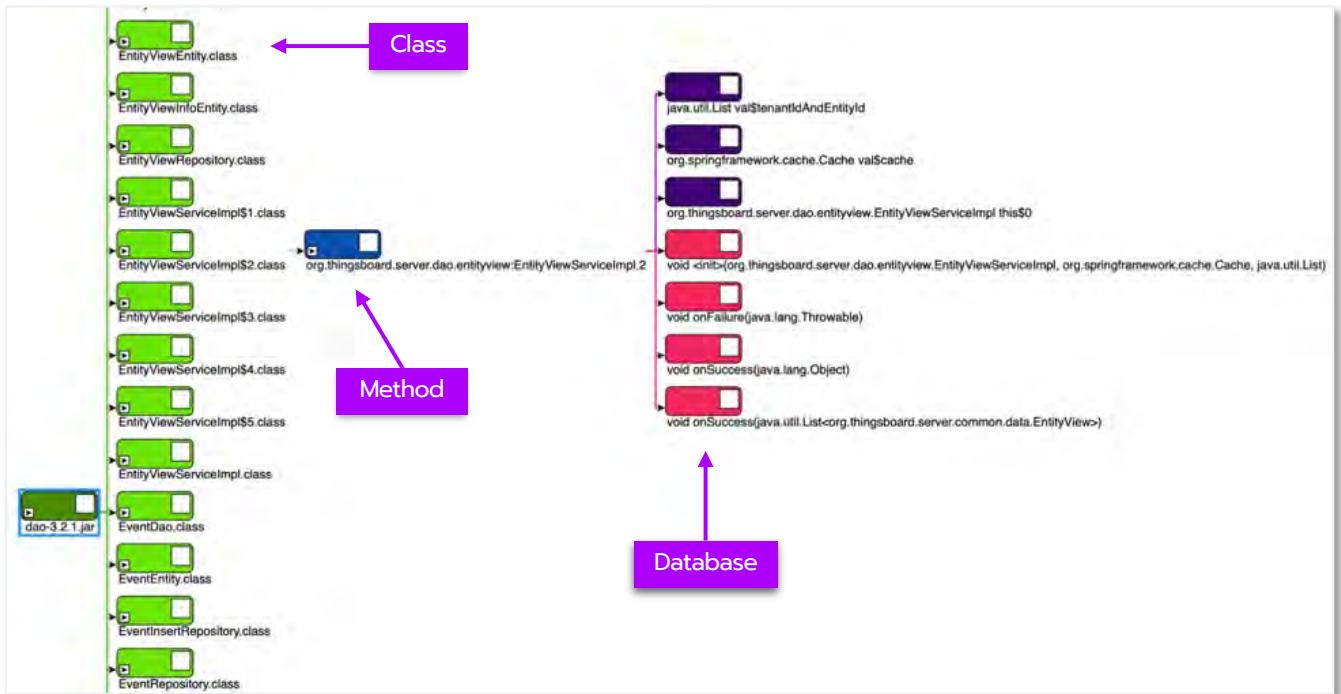
CodeLogic delivers the reliable insights that application teams need to accurately navigate code changes – from small product enhancements to full-stack architectural updates. CodeLogic's self-updating application code map enables developers, architects, SRE's and testing teams to quickly see all the unknown or hidden connections and dependencies within and across applications.

Our Approach

CodeLogic's application intelligence goes beyond source and starts where other tools – such as APM, ITSM, and IDEs – stop. CodeLogic provides the industry's most comprehensive, real-time application dependency mapping and change impact analysis tool. Teams using CodeLogic can map, document, visualize, and understand their application connections and dependencies from API to method to database column – on schedule, on-demand, or after every build.

Question 1

What's in my application?

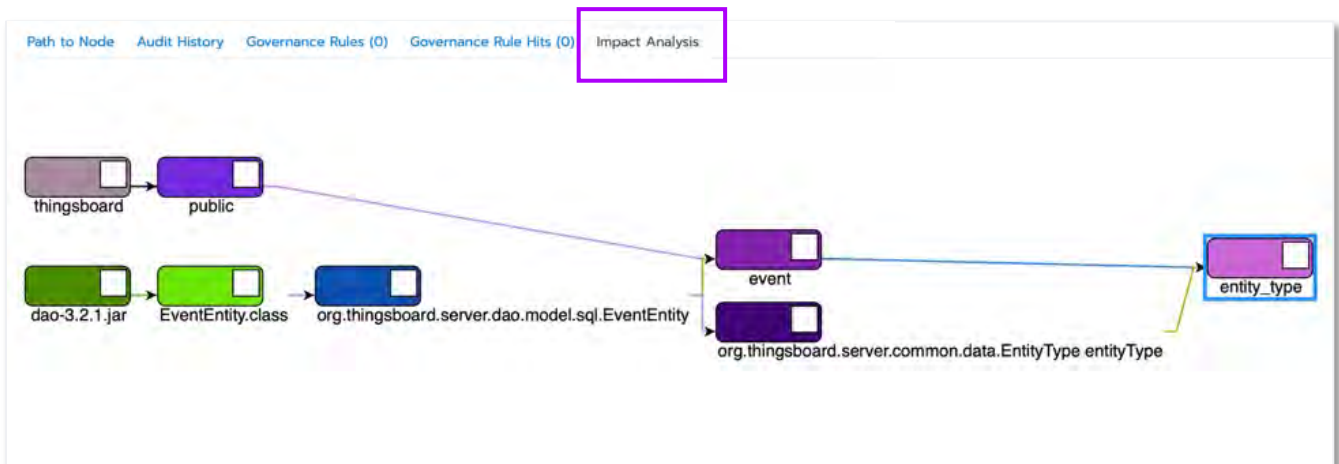


CodeLogic Answer

Visualize a map of your code from class to method to database and see how they're connected.

Question 2

What's the impact if I am going to make a change?



CodeLogic Answer

See all the items impacted by a proposed change.

Question 3

What's changed since my last update?

The screenshot shows the 'Node Details' page for a node named 'event'. The page is divided into three main sections: 'Inbound Relationships', 'Selected Node', and 'Outbound Relationships'. The 'Selected Node' section displays the following properties:

- id: 591ab9ef-7fb6-35f6-904f-638e99f44947
- identity: jdbc:postgresql://localhost:5432/thingsboard|public|event
- firstObserved: 2021-08-10T19:36:55.784Z
- dataSourceId: sqlCape
- primaryLabel: SqlJdbcTable
- lastObserved: 2021-08-10T19:36:55.784Z
- nodeOrRelationship: node
- isScanRoot: false

Below the node details, there is a 'Current node link' and a navigation bar with options: 'Path to Node', 'Audit History', 'Governance Rules (0)', 'Governance Rule Hits (0)', and 'Impact Analysis'. The 'Audit History' option is highlighted with a purple box. Below the navigation bar, the 'Changes on 2021-08-10' section shows a list of changes for 'Version 1' with purple arrows pointing to the 'New Value' field:

- Added Key: firstObserved
New Value: 1628624215784
- Added Key: primaryLabel
New Value: SqlJdbcTable
- Added Key: identity
New Value: jdbc:postgresql://localhost:5432/thingsboard|public|event
- Added Key: nodeOrRelationship
New Value: node
- Added Key: name
New Value: event
- Added Key: id
New Value: 591ab9ef-7fb6-35f6-904f-638e99f44947

CodeLogic Answer

An audit history logs all changes made to an application.

Question 4

Where is the technical debt in my code?

The screenshot displays the CodeLogic interface. On the left, a sidebar shows a 'Nodes' list with columns for Type, Color, Visible, and Count. A 'Macro View' callout points to a large network diagram of nodes and relationships. A 'Node Details' window is open, showing 'Inbound Relationships' (with 'ts_kv_2021_04' listed), 'Selected Node' (with details for 'json_v'), and 'Outbound Relationships'. A callout 'Item-Level View' points to the 'Node Details' window. At the bottom, a path diagram shows 'thingsboard' -> 'public' -> 'ts_kv_2021_04' -> 'json_v', with an 'Impact Analysis' button highlighted.

CodeLogic Answer

In this example, a macro view of item relationships highlights where there are orphaned tables in your database. An item-level view shows the details.

