Creating a REST API to Distribute Student Data for Departmental Applications

July 11, 2016
Student Records Database (SRDB)

• Started in 1997 to Support Campus Demand for Student Data
• Source of Near Real-Time Student Data (5 minute Latency)
• Source of Blended Data
• Over 1500 Database Objects (Stored Procs, Views etc.)
• Major Component of Campus Infrastructure
  • Academic Support
  • Administrative Support
  • Operation/Production Support
Current State of Data Distribution

Point to Point Solutions via SRDB and SOAP based Web Services
What is an API?

API (application program interface) is a set of routines, protocols, and tools for building software applications

• Provides building blocks for graphical user interface

• Allows implementation in different ways without compromising the interface
What is a REST API?

Representational State Transfer (REST) is a software architectural style.

http://www.restapitutorial.com/lessons/whatisrest.html#

- base URI

- standard HTTP methods (e.g., GET, PUT, POST, or DELETE)

- Internet media type for data (e.g., JSON, XML)

- hypertext links
Service-Oriented Architecture (SOA)

• Service-Oriented Architecture:
  • an architectural style that supports service-orientation

• Service-Orientation:
  • A way of thinking in terms of services (units of functionality)

• Service:
  • a logical representation of a repeatable business activity that has a specified outcome (e.g. Courses, Students, Rosters etc.)
Service-Oriented Architecture

Services are composed of:

- Service contract layer
- Service logic layer
- Data layer
Service-Oriented Architecture

• Utility Services:
  • Based on non-business-centric functional contexts
  • Example: Authentication, Authorization

• Entity Services:
  • Functional context derived from business entities
  • Re-used in support of business processes
  • Example: Student Ethnicity
Service-Oriented Architecture

- Service Loose Coupling: minimize dependencies
- Service Abstraction: minimize the availability of meta information
- Service Composability: maximize composability
- Implement a standardized contract
  - Standardized Service Contract
  - Service Reusability: implement generic and reusable logic and contract
  - Service Autonomy: implement independent functional boundary and runtime environment
  - Service Statelessness: implement adaptive and state management-free logic
  - Service Discoverability: implement communicative meta information
RESTful API Demo