Chapter 9

RESTful Web Services Development With a Model-Driven Engineering Approach

Rafael Corveira da Cruz Gonçalves
Polytechnic Institute of Porto, Portugal

Isabel Azevedo
Polytechnic Institute of Porto, Portugal

ABSTRACT

A RESTful web service implementation requires following the constrains inherent to REST architectural style, which, being a non-trivial task, often leads to solutions that do not fulfill those requirements properly. Model-driven techniques have been proposed to improve the development of complex applications. In model-driven software development, software is not implemented manually based on informal descriptions but partially or completely generated from formal models derived from metamodels. A model-driven approach, materialized in a domain specific language that integrates the OpenAPI specification, an emerging standard for describing REST services, allows developers to use a design first approach in the web service development process, focusing in the definition of resources and their relationships, leaving the repetitive code production process to the automation provided by model-driven engineering techniques. The code generation process covers the entire web-service flow from the description and exposure of the endpoints to the definition of database tables.

DOI: 10.4018/978-1-5225-7455-2.ch009
INTRODUCTION

During the last few years, offering software in the form of web services has gained popularity due to the evolution of cloud architectures and the impact of the Internet of Things (IoT) on the number of connections that can take place.

Ever since its introduction by Fielding (Fielding, 2000), the REpresentational State Transfer (REST) architectural style has been increasingly preferred by developers for its simplicity and scalability (Haupt, Leymann, Scherer, & Vukojevic-Haupt, 2017). REST comprises a set of rules and practices and allows simple comprehensible Application Programming Interfaces (APIs), clear representations, and scalable services (Dimitrieski et al., 2017).

In an attempt at defining a set of standards to describe APIs, and to aid developers in the creation of the interfaces, the OpenAPI Initiative (OAI) was formed by “a consortium of forward-looking industry experts who recognize the immense value of standardizing on how REST APIs are described. As an open governance structure under the Linux Foundation, the OAI is focused on creating, evolving and promoting a vendor neutral description format” (The Linux Foundation, 2017a).

The OpenAPI Specification requires the description of the capabilities of each service without mandating a specific development process, such as design-first or code-first, but does facilitate either technique by establishing clear interactions with a REST API (The Linux Foundation, 2017b).

Meanwhile, from an automated software engineering perspective, Model Driven Engineering (MDE) is gaining popularity (Zolotas, Diamantopoulos, Chatzidimitriou, & Symeonidis, 2017), and it can be useful for the development of REST APIs. A great amount of time is required because developers must address several issues, such as software architecture, error handling, security, access rules, among others, to ensure a suitable design and the desired usability (Schreibmann & Braun, 2014). To implement this solution according to all the best practices is time consuming and it is an expensive part of every software project.

The specification of the structural and behavioral aspects of an application using model-driven principles can be used for code generation (Schreibmann & Braun, 2015).

With the standardization of REST APIs description, provided by the OAI, an MDE approach for the development of a domain specific language (DSL) to generate full API applications can be a helpful and powerful resource for the developers (Scheidgen, Efttinge, & Marticke, 2016). Although there have been several initiatives devoted to the modeling and formal description of REST applications, most of them do not support or even force RESTful compliance (Ed-Douibi, Izquierdo, Gómez,
Tools for the Learning of Programming Languages and Paradigms: Integration of a Code Validator and Exercises Module Into the Moodle eLearning Platform

A Survey on Computer Programming Learning Environments