
WRML
WRML
WRML

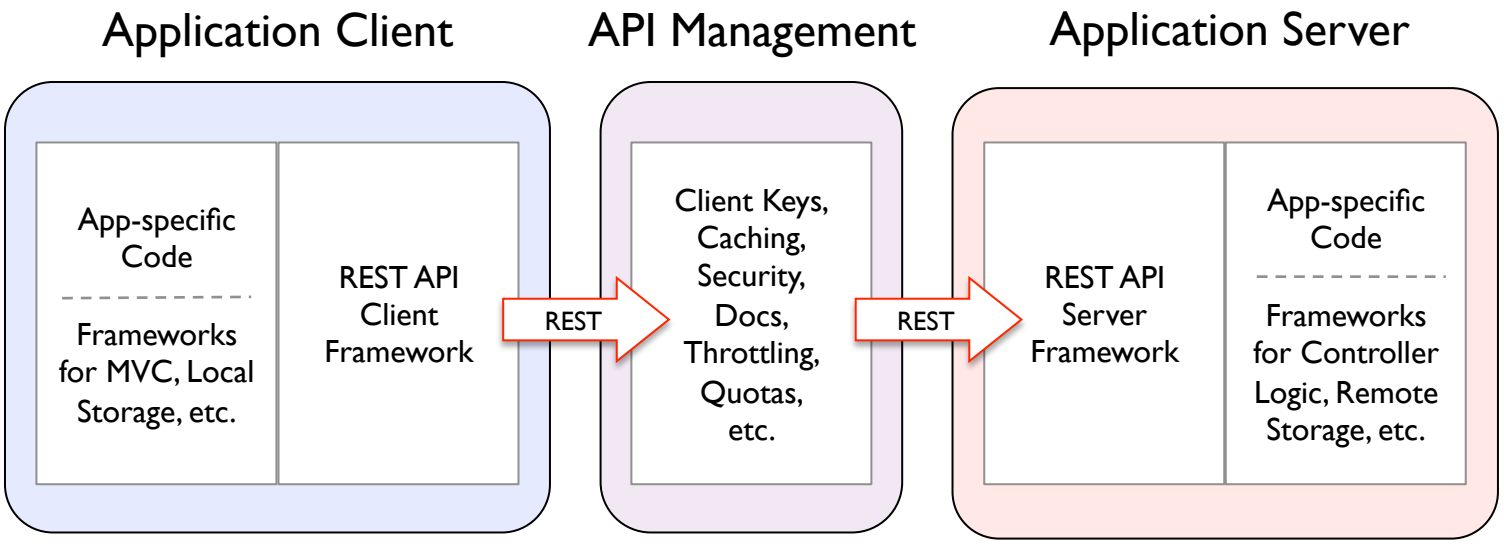
WRML

Web Resource Modeling Language

Design Notes

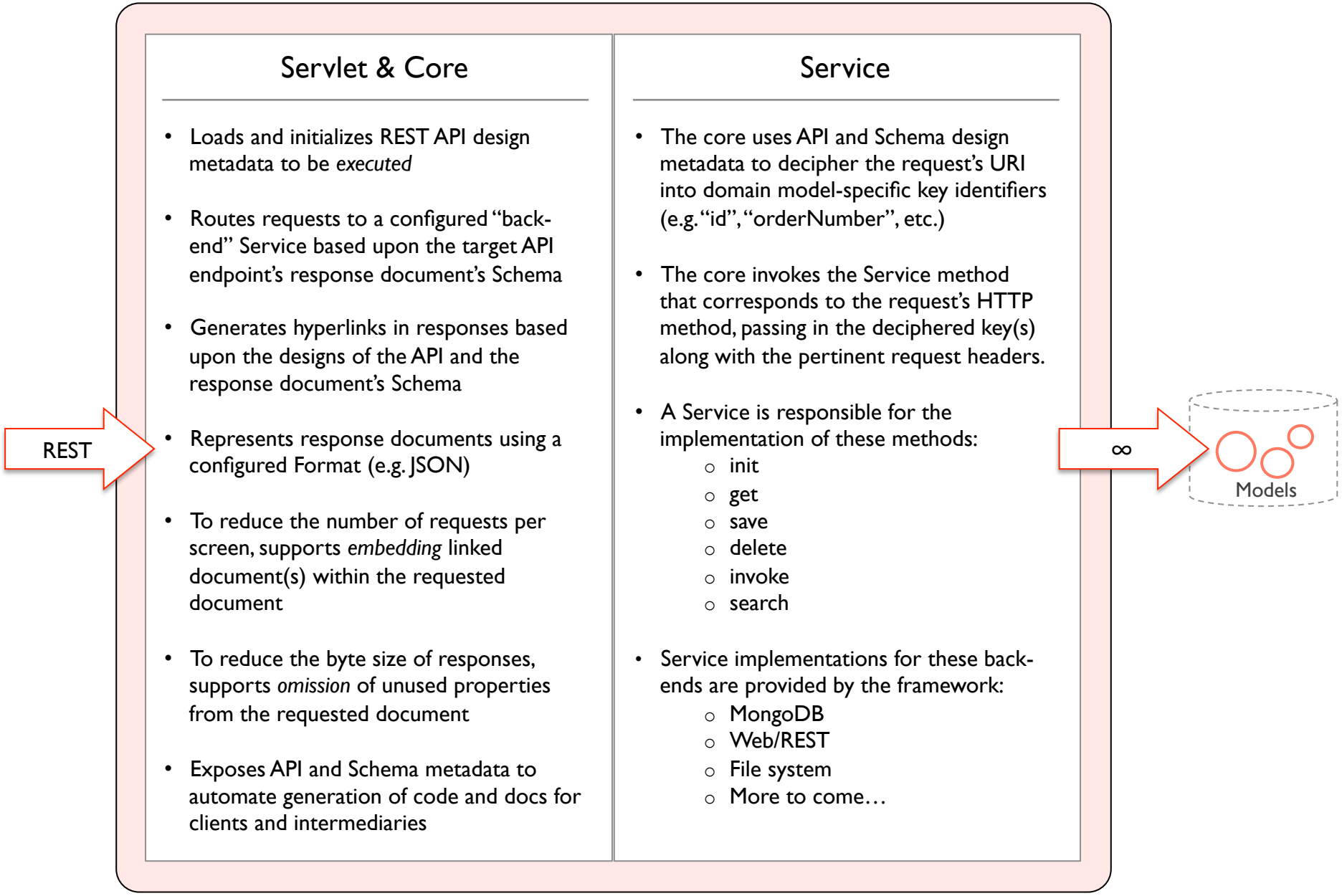
All Slides - Copyright © Mark Masse (OSS project WRML.org)

REST-Oriented Application Architecture



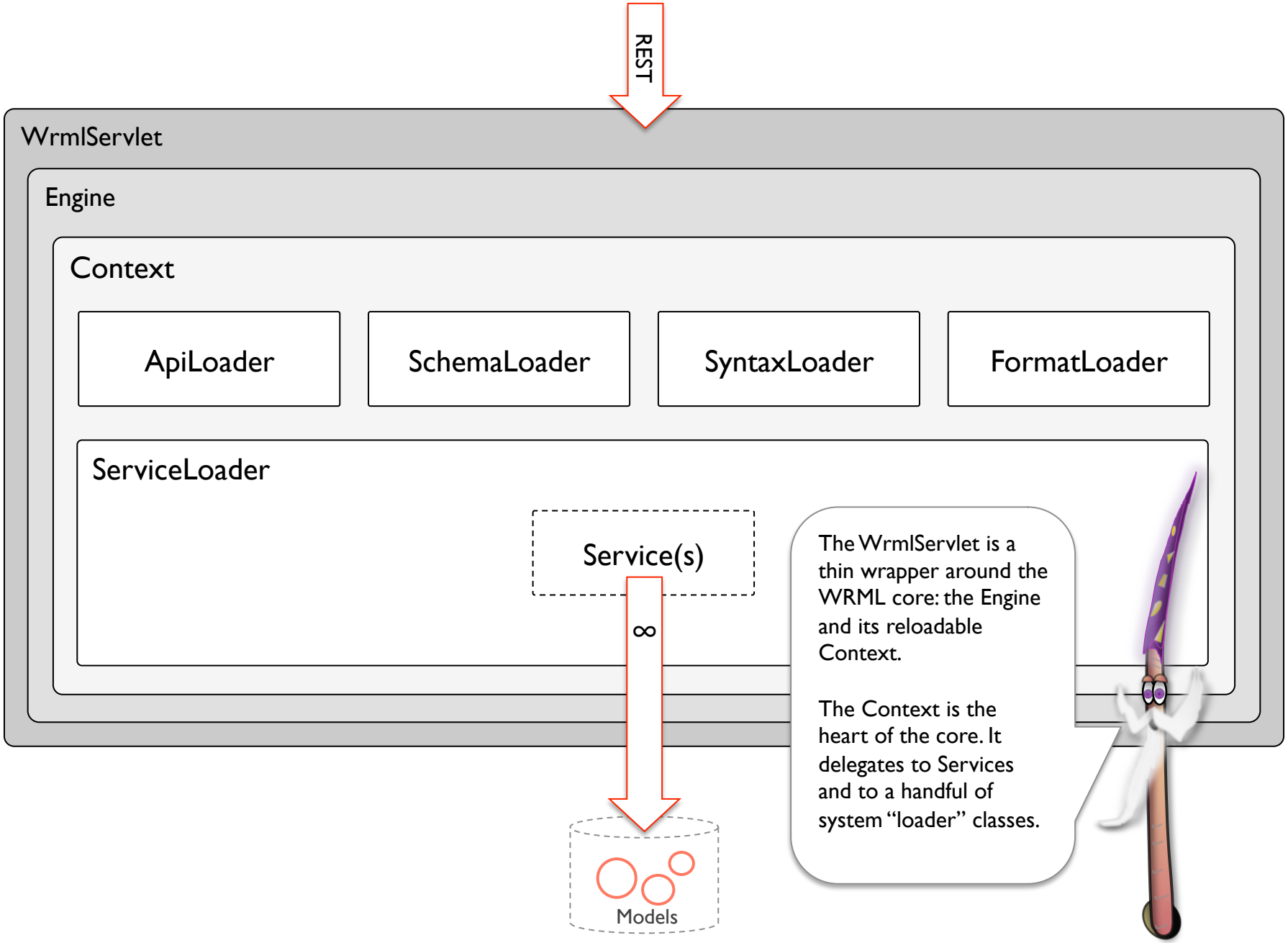
All Slides - Copyright © Mark Masse (OSS project WRML.org)

WRML Application Server Architecture



All Slides - Copyright © Mark Masse (OSS project WRML.org)

WRML Application Server Internals



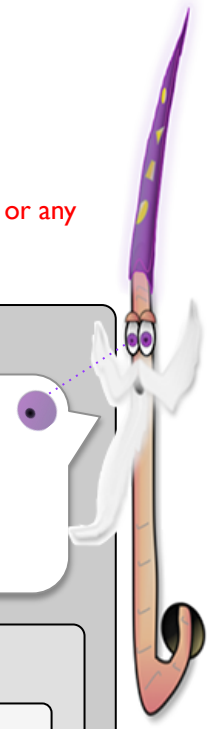
All Slides - Copyright © Mark Masse (OSS project WRML.org)

REST Interactions with WRML Application Server

Client App

REST

Accept either: *application/json*, *application/wrml*, *application/html*, or any other server-loaded Format's associated media type.



WrmlServlet

The WrmlServlet handles the direct communication with clients. It is designed to be a minimalistic pass-through layer that translates HTTP-based communication to and from the WRML Context and its core components. Once a REST request has been validated and the target Resource endpoint has been identified, a Service method is invoked to handle the service-specific business logic. The WRML core is designed to maintain a clear separation of concerns between a web service's client-facing interface and its implementation details (business logic).

WRML

Engine

Context

Request routing and HATEOAS automation

ApiLoader

Generic model implementation support

SchemaLoader

Formatting and parsing text into handy "wrapper" objects

SyntaxLoader

Pluggable serialization in a variety of media types

FormatLoader

ServiceLoader

CRUD, *Search*, and *Invoke* implementation (flexible backend)

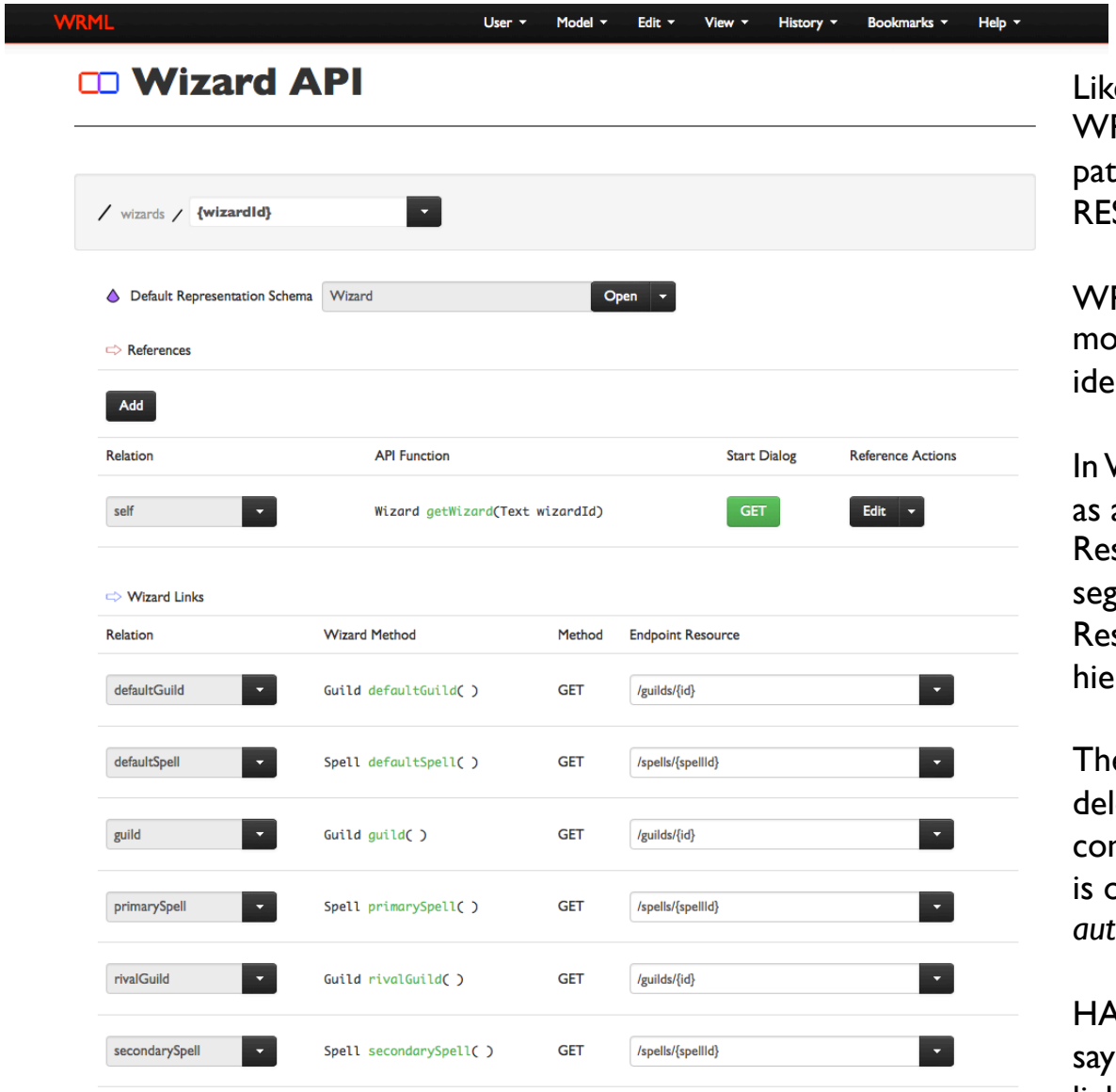
Service(s)

∞

Models

All Slides - Copyright © Mark Masse (OSS project WRML.org)

WRML REST API Model



Like I/O Docs, API Blueprint, and Swagger, WRML uses metadata to represent the URI pattern and allowed interactions for each REST API resource endpoint.

WRML expresses REST API metadata as a model described by the WRML Schema identified by: org/wrml/model/rest/Api.

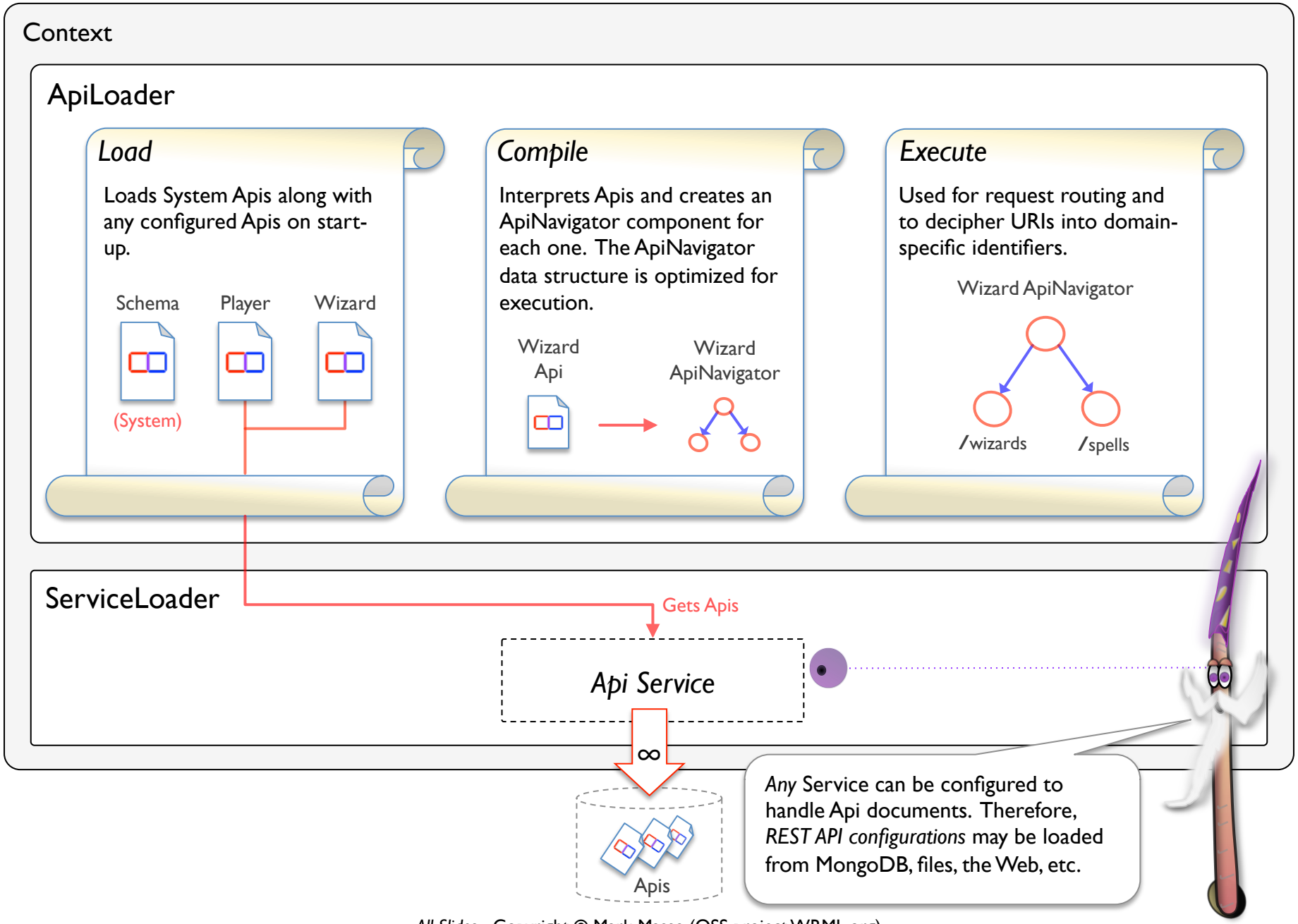
In WRML, an *Api* model defines the endpoints as a tree of *ResourceTemplates*. Each ResourceTemplate model has a URI path segment and an optional list of child ResourceTemplates to represent the hierarchical nature of URIs.

The WRML Application Server's runtime delegates to the *ApiLoader* component to compile each Api model into a structure that is optimized for request routing and *HATEOAS automation*.

HATEOAS automation is just a fancy way of saying that WRML uses metadata to generate link href values in response representations.

All Slides - Copyright © Mark Masse (OSS project WRML.org)

ApiLoader



All Slides - Copyright © Mark Masse (OSS project WRML.org)

REST API Loading Details

ApiLoader

Load API

At present, REST API metadata may be loaded in the WRML JSON format.

The plan is to also support these formats are for API representation:

- I/O Docs
- API Blueprint
- Swagger

Generate Api Model

Once read, the Api model is cached by the ApiLoader and mapped to an *ApiNavigator* data structure that aims to optimize runtime execution of API metadata related concerns.

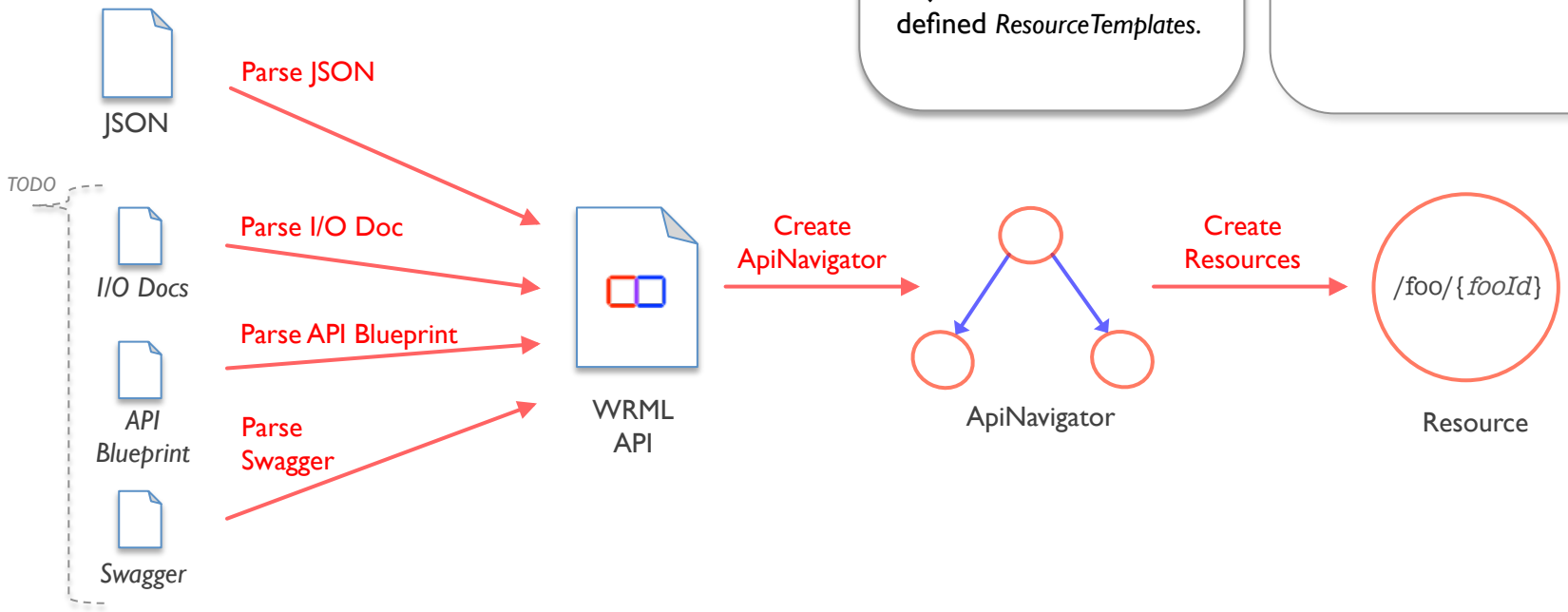
Create ApiNavigator

Within the ApiLoader, every *loaded* REST API (Api model) is wrapped by an ApiNavigator instance.

The ApiNavigator object processes an Api model and creates a *Resource* object for each of the defined *ResourceTemplates*.

Create Resources

The Resource objects, together with their parent ApiNavigator enable the WRML runtime to facilitate the routing of inbound REST requests and the generation of hyperlinks (*href* values) in response documents.

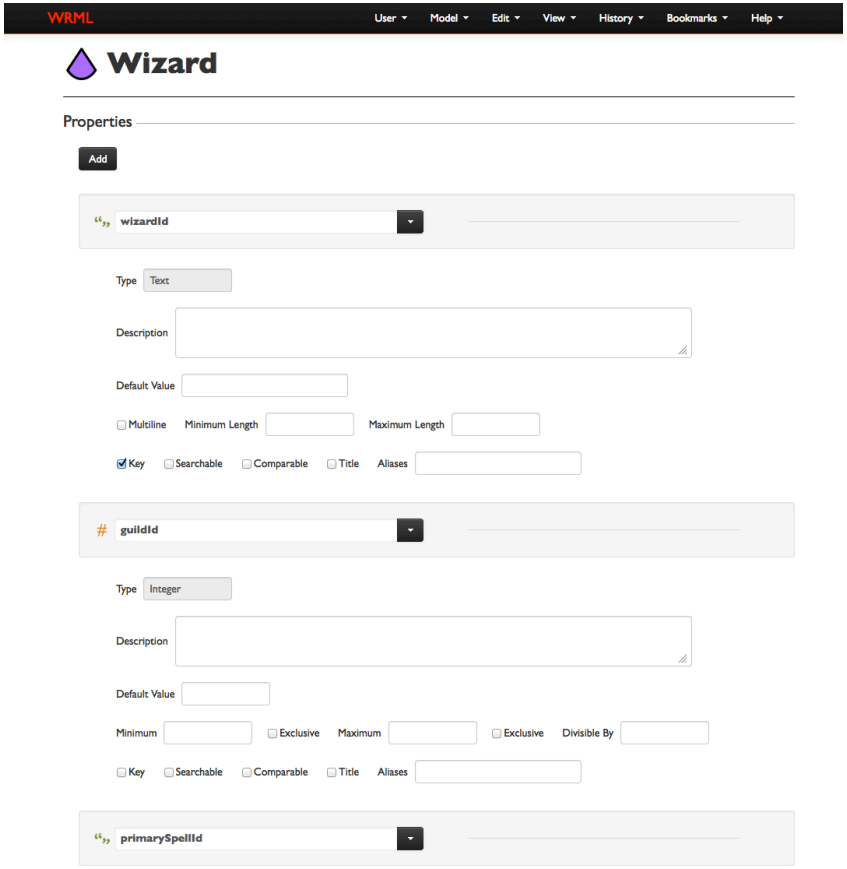


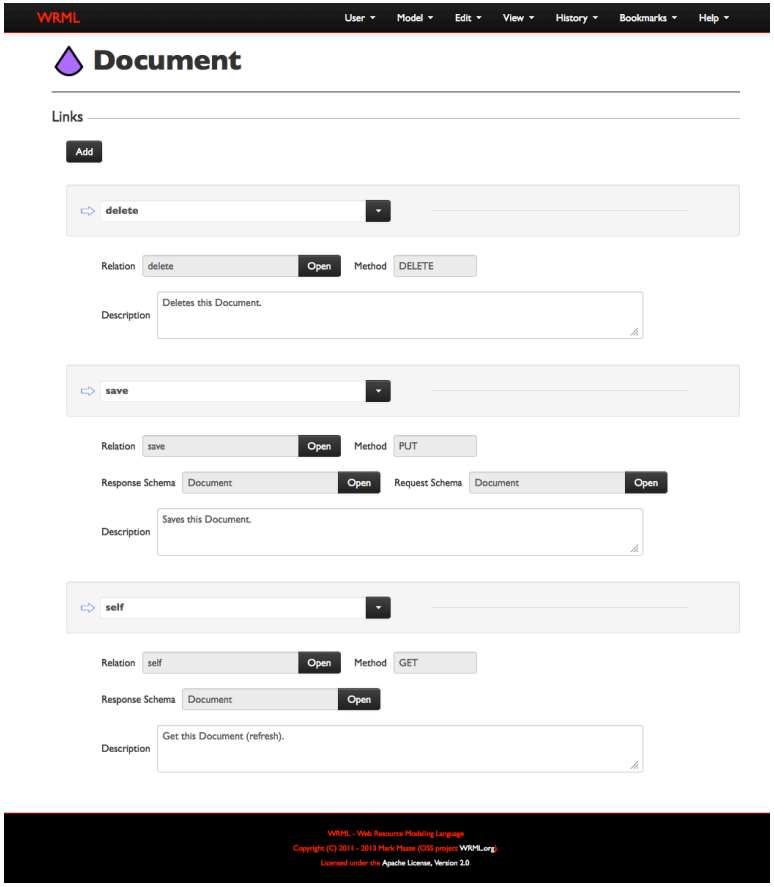
All Slides - Copyright © Mark Masse (OSS project WRML.org)

WRML Schema Model

Schemas describe the structure for a "class" of Models. In this way, Schema are like O-O interfaces. Schema models are also a bit like: SQL table structures, XML DTDS, JSON or XML schemas, AVRO schemas, XLS templates, CMS content types, etc.

A Schema may declare any number of name-value *slots*, with the slot value type being one of the following primitives:





All Slides - Copyright © Mark Masse (OSS project WRML.org)

Schemas and the WRML Type System

A Schema may declare any number of name-value *slots*, with the slot value type being one of the following primitives:

Value Type	Description	JSON Type	Java Type
Boolean	Must be true, false, or null.	true or false	java.lang.**Boolean**
Date	A date (with an optional time component).	string	java.util.**Date**
Double	A double-precision 64-bit IEEE 754 floating point number.	number	java.lang.**Double**
Integer	A 32-bit integer.	number	java.lang.**Integer**
Link	A simple structure containing URI values for ***href*** and ***rel*** slots.	object	org.wrml.model.rest.**Link**
List	A list of values of any single (non-List) type (Text, Integer, etc.).	array	java.util.**List**
Long	A 64-bit integer.	number	java.lang.**Long**
Map*	A mapping of Text keys to any single (non-Map) typed value.	object	java.util.**Map**
Model	An instance of a Schema with a map containing its slot values.	object	org.wrml.model.**Model**
Single Select	A special Text-based value that is selected from a fixed menu of possible choices.	string	java.lang.**Enum**
Text	A sequence of zero or more Unicode characters.	string	java.lang.**String**

* See Issue: WRML-300

All Slides - Copyright © Mark Masse (OSS project WRML.org)

SchemaLoader

Context

SchemaLoader

Load

Loads System Schemas along with any configured Schemas on start-up.

Format Player Wizard

(System)

Compile

Interprets Schemas and creates a Prototype component for each one. The Prototype data structure is optimized for execution.

Wizard Schema → Wizard Prototype

Execute

Used for type identification, structural description, and model instance validation.

Wizard Prototype

Name	Type
wizardKey	Text
guildId	Integer
guild	Link

ServiceLoader

Gets Schemas

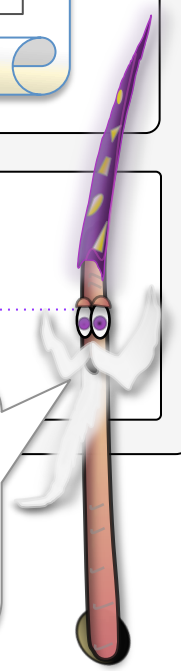
Schema Service

∞

Schemas

A Schema describes the structure (properties and links) of a *class* of models.

Any Service can be configured to handle Schema documents.



All Slides - Copyright © Mark Masse (OSS project WRML.org)

Schema Loading Details

SchemaLoader

Load Schema

Schemas may be loaded in a variety of *formats*.

These formats are currently supported for Schemas:

- application/wrml (JSON)
- JSON Schema
- Java Interface

Generate Schema Model

If a Schema is not *native*, meaning it did not originate as a Java interface (with WRML annotations), then it will be read (desterilized) via a Format-specific Formatter component.

Once read, the Schema model is cached by the SchemaLoader.

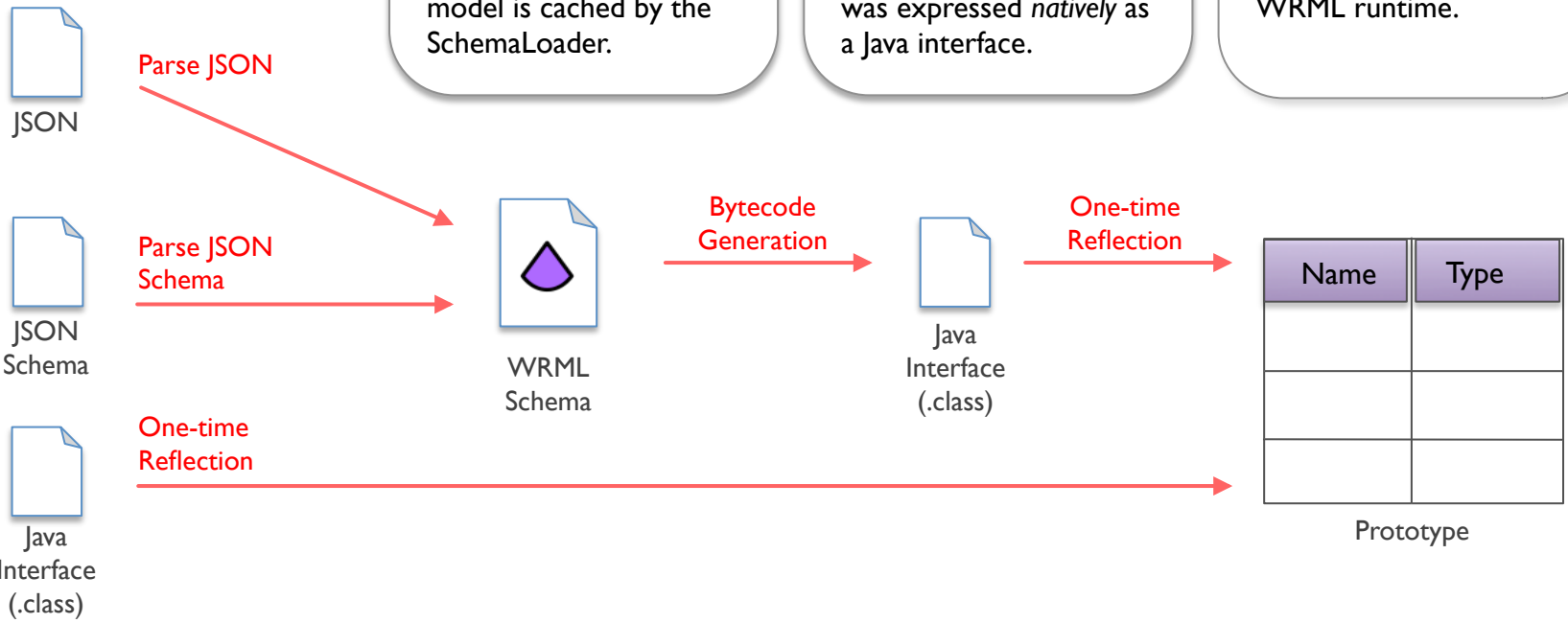
Generate Schema Interface

The SchemaLoader uses a *SchemaGenerator* component to create the Java bytecode (.class) representation of the WRML Schema.

This brings the loading process to the same point as if the Schema was expressed *natively* as a Java interface.

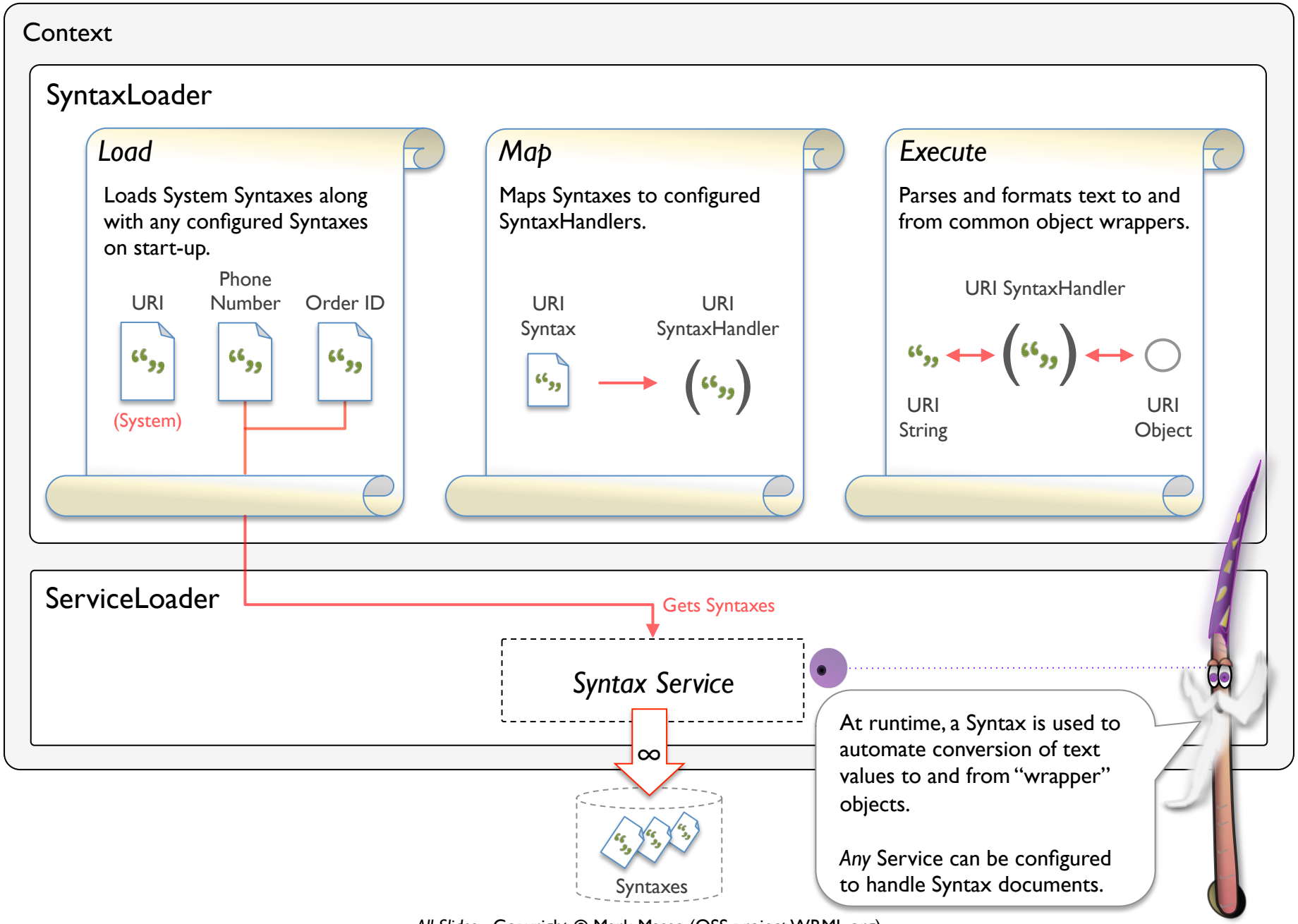
Generate Schema Prototype

For both native and generated Schemas, the runtime performs a one-time reflection on the Java interface to extract the annotations and method signature information needed to optimize the many uses for Schemas within the WRML runtime.



All Slides - Copyright © Mark Masse (OSS project WRML.org)

SyntaxLoader



All Slides - Copyright © Mark Masse (OSS project WRML.org)

FormatLoader

Context

FormatLoader

Load

Loads System Formats along with any configured Formats on start-up.

JSON YAML Avro

(System)

Map

Maps Formats to configured Formatters.

JSON Format → JSON Formatter

Execute

Writes and reads models to and from formatted output and input streams.

JSON Formatter

I/O Stream — Model

ServiceLoader

Gets Formats

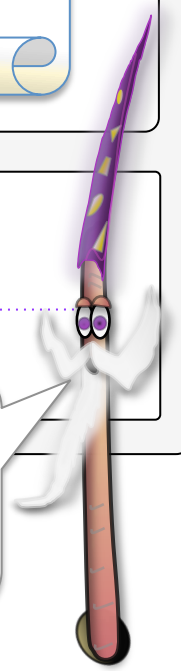
Format Service

∞

Formats

A Format describes a model serialization technique.

Any Service can be configured to handle Format documents.



All Slides - Copyright © Mark Masse (OSS project WRML.org)

Service Interface

Context

ServiceLoader

Service

The *Keys* structure contains the requested URI, along with any deciphered, domain-specific identifiers.

The *Dimensions* structure contains the requested Schema URI, to identify the response type, along with any pertinent request header metadata (e.g. Locale).

Init

Perform one-time initialization of the Service.

Get

Get a model based on the specified *Keys* and *Dimensions*.

Search

Get a set of models based upon the specified SearchCritera.

Save

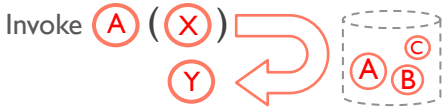
Save (PUT) the specified model.

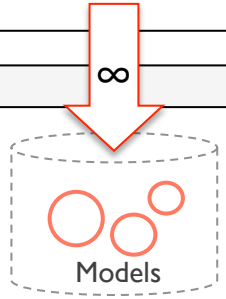
Delete

Delete a model based on the specified Keys and Dimensions.

Invoke

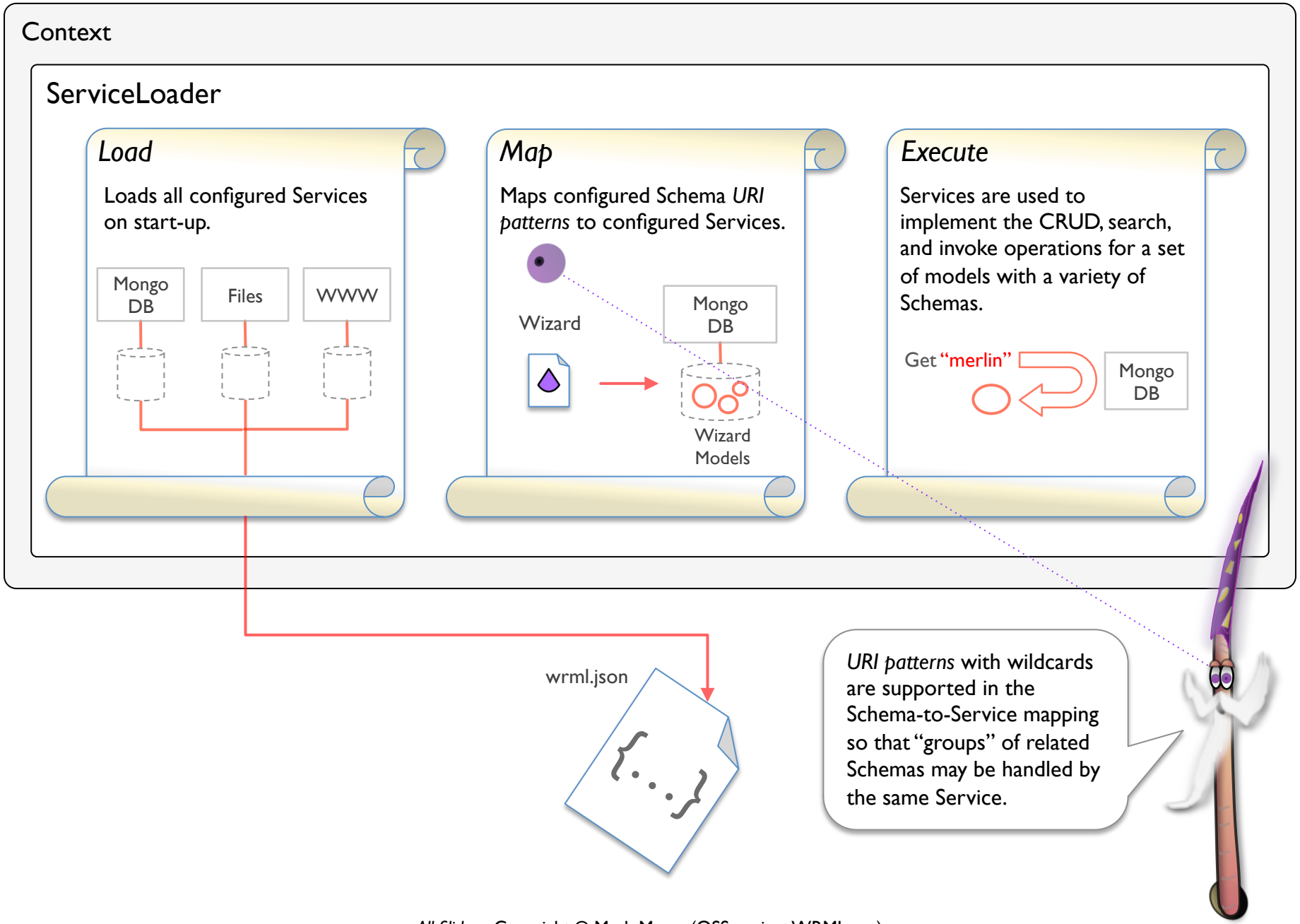
Invokes a *functional* model with an (optional) parameter model.





All Slides - Copyright © Mark Masse (OSS project WRML.org)

ServiceLoader

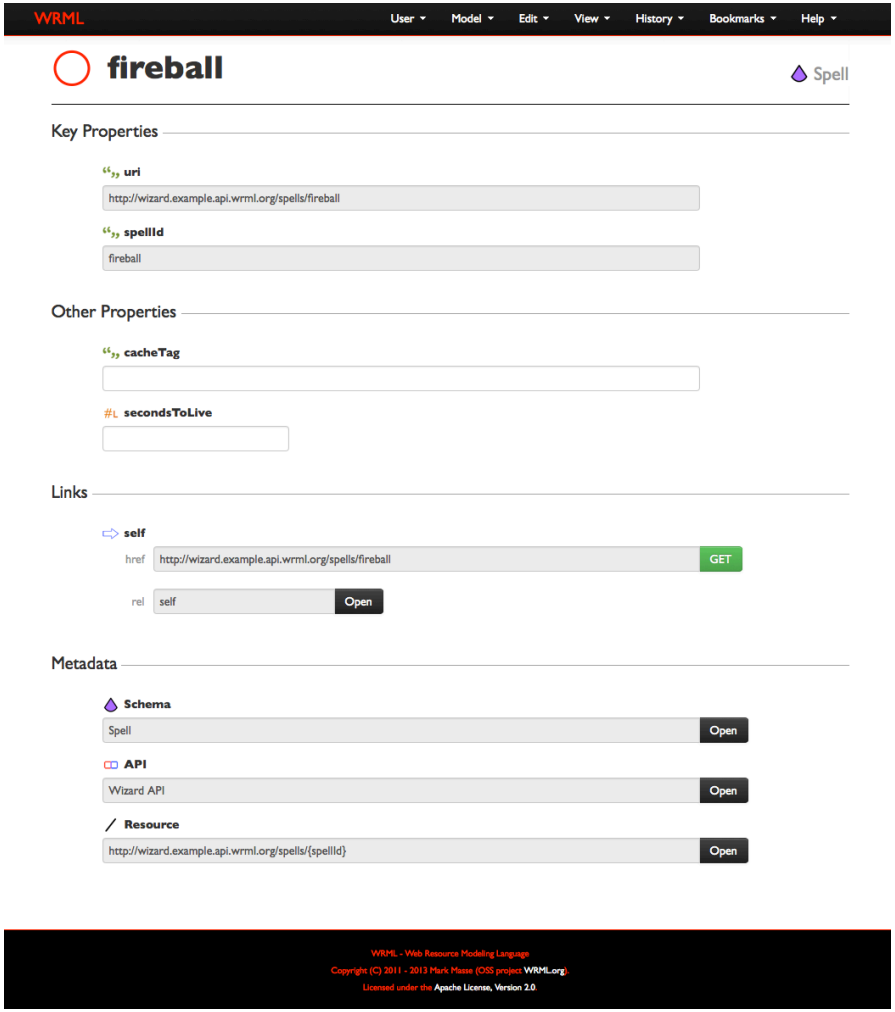


All Slides - Copyright © Mark Masse (OSS project WRML.org)

WRML REST API Design Tools

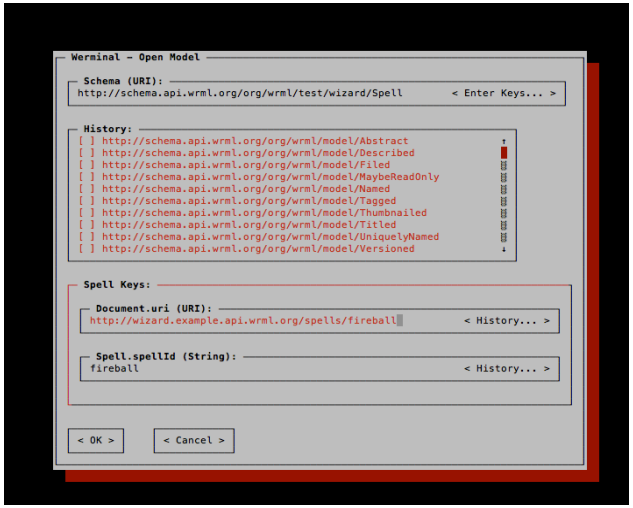
Wrmldoc

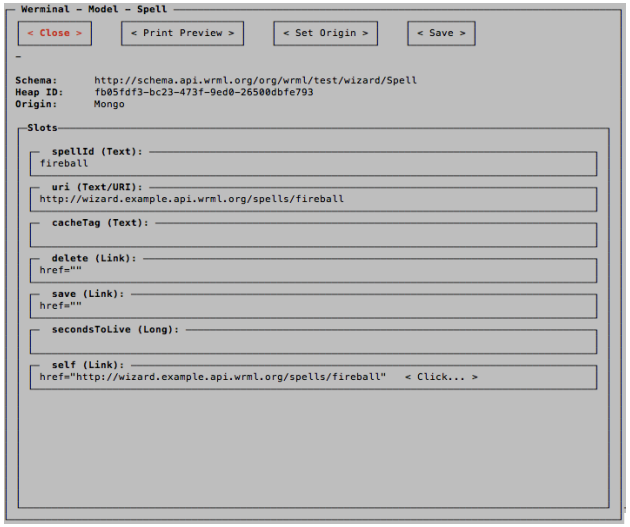
Web-based GUI



Werminal

Command-line GUI





All Slides - Copyright © Mark Masse (OSS project WRML.org)