



WRML

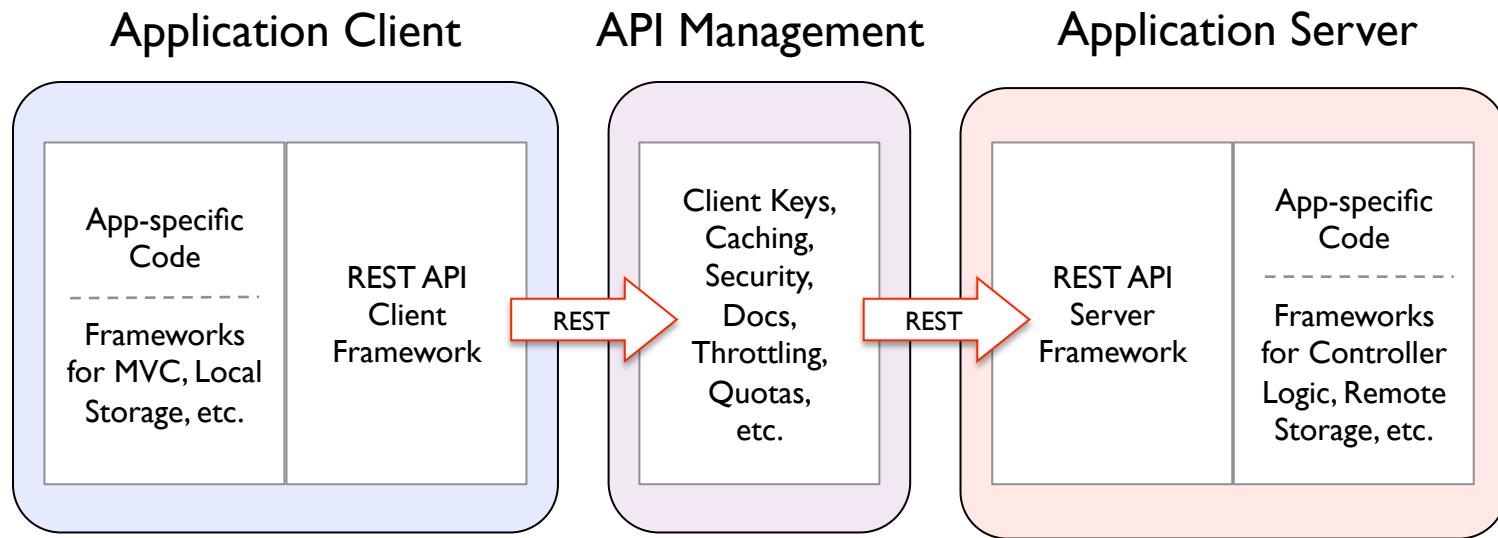
WRML

Web Resource Modeling Language

Design Notes



REST-Oriented Application Architecture



WRML Application Server Architecture

Servlet & Core

- Loads and initializes REST API design metadata to be executed
- Routes requests to a configured “back-end” Service based upon the target API endpoint’s response document’s Schema
- Generates hyperlinks in responses based upon the designs of the API and the response document’s Schema
- Represents response documents using a configured Format (e.g. JSON)
- To reduce the number of requests per screen, supports *embedding* linked document(s) within the requested document
- To reduce the byte size of responses, supports *omission* of unused properties from the requested document
- Exposes API and Schema metadata to automate generation of code and docs for clients and intermediaries

REST

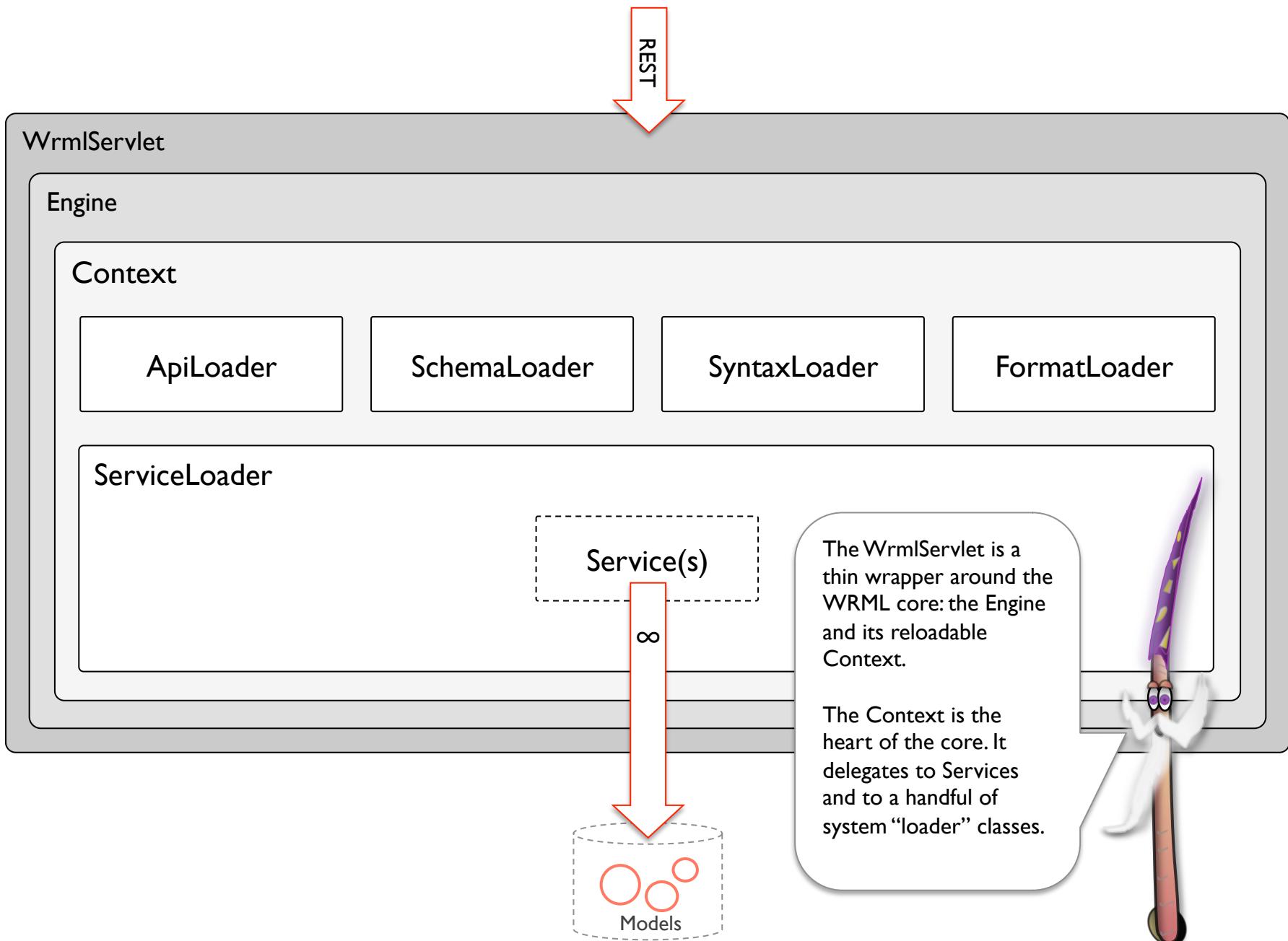
Service

- The core uses API and Schema design metadata to decipher the request’s URI into domain model-specific key identifiers (e.g. “id”, “orderNumber”, etc.)
- The core invokes the Service method that corresponds to the request’s HTTP method, passing in the deciphered key(s) along with the pertinent request headers.
- A Service is responsible for the implementation of these methods:
 - init
 - get
 - save
 - delete
 - invoke
 - search
- Service implementations for these back-ends are provided by the framework:
 - MongoDB
 - Web/REST
 - File system
 - More to come...

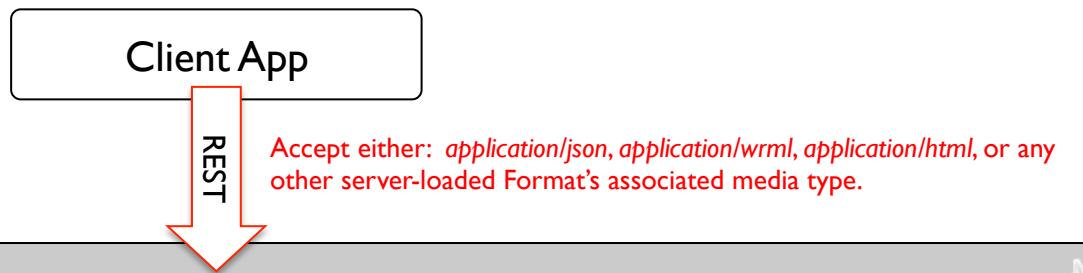
∞

Models

WRML Application Server Internals



REST Interactions with WRML Application Server



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).

Engine

Context

Request routing and HATEOAS automation

Generic model implementation support

Formatting and parsing text into handy “wrapper” objects

Pluggable serialization in a variety of media types

ApiLoader

SchemaLoader

SyntaxLoader

FormatLoader

ServiceLoader

CRUD, Search, and Invoke
implementation (flexible backend)

Service(s)

∞



WRML REST API Model

WRML

User ▾ Model ▾ Edit ▾ View ▾ History ▾ Bookmarks ▾ Help ▾

Wizard API

The screenshot shows the WRML interface with the title "Wizard API". The top navigation bar includes links for User, Model, Edit, View, History, Bookmarks, and Help. Below the title, there's a breadcrumb trail: / wizards / {wizardId}. A toolbar with "Default Representation Schema" (dropdown), "Wizard" (button), "Open" (button), and "Edit" (button) is visible. A "References" section and an "Add" button are also present. The main content area has tabs for Relation, API Function, Start Dialog, and Reference Actions. Under the Relation tab, a row shows "self" as the relation, "Wizard getWizard(Text wizardId)" as the API Function, "GET" as the Method, and "/guilds/{id}" as the Endpoint Resource. An "Edit" button is next to the method. Below this, a "Wizard Links" section lists several entries:

Relation	Wizard Method	Method	Endpoint Resource
defaultGuild	Guild defaultGuild()	GET	/guilds/{id}
defaultSpell	Spell defaultSpell()	GET	/spells/{spellId}
guild	Guild guild()	GET	/guilds/{id}
primarySpell	Spell primarySpell()	GET	/spells/{spellId}
rivalGuild	Guild rivalGuild()	GET	/guilds/{id}
secondarySpell	Spell secondarySpell()	GET	/spells/{spellId}

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.

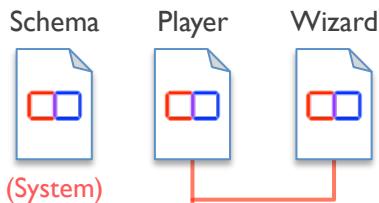
ApiLoader

Context

ApiLoader

Load

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



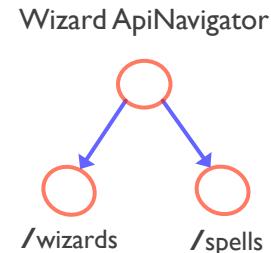
Compile

Interprets Apis and creates an ApiNavigator component for each one. The ApiNavigator data structure is optimized for execution.



Execute

Used for request routing and to decipher URLs into domain-specific identifiers.

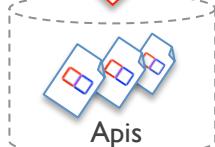


ServiceLoader

Gets Apis

Api Service

∞

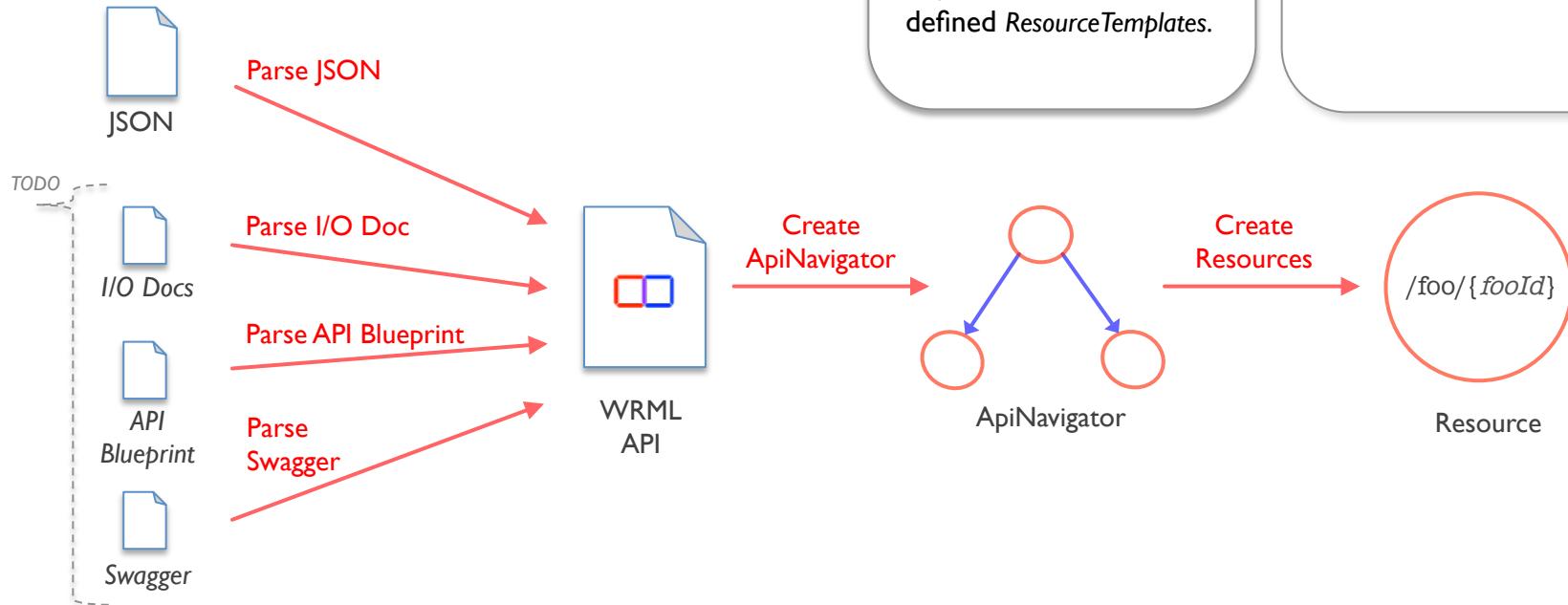
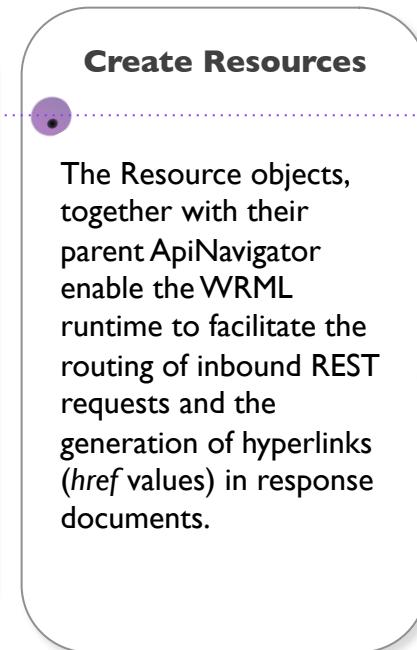
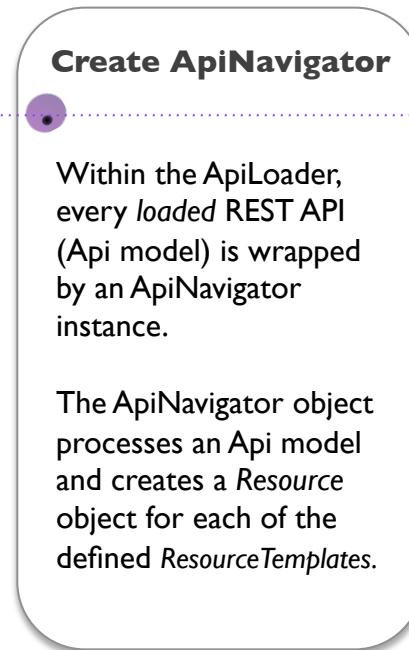
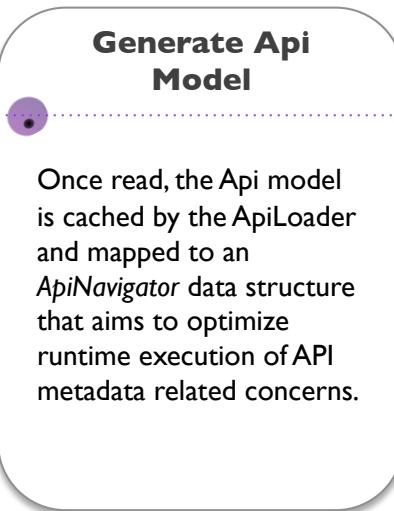
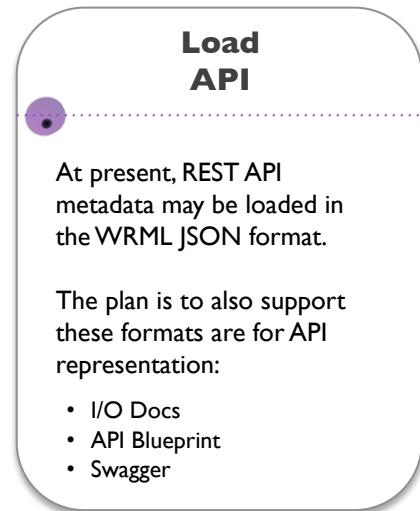


Any Service can be configured to handle Api documents. Therefore, REST API configurations may be loaded from MongoDB, files, the Web, etc.



REST API Loading Details

ApiLoader



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:

WRML Wizard

Properties

Add

wizardId

Type: Text

Description

Default Value

Multiline Minimum Length Maximum Length

Key Searchable Comparable Title Aliases

guildId

Type: Integer

Description

Default Value

Minimum Exclusive Maximum Exclusive Divisible By

Key Searchable Comparable Title Aliases

primarySpellId

WRML Document

Links

Add

delete

Relation: delete Open Method: DELETE

Description: Deletes this Document.

save

Relation: save Open Method: PUT

Response Schema: Document Open Request Schema: Document Open

Description: Saves this Document.

self

Relation: self Open Method: GET

Response Schema: Document Open

Description: Get this Document (refresh).

WRML - Web Resource Modeling Language
Copyright (C) 2011 - 2013 Mark Masse (OSS project WRML.org)
Licensed under the Apache License, Version 2.0

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](#)

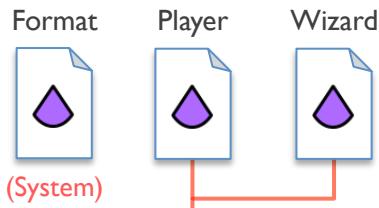
SchemaLoader

Context

SchemaLoader

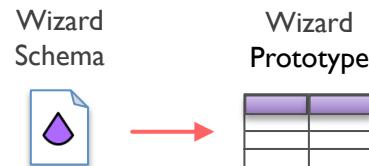
Load

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



Compile

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



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



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

Any Service can be configured to handle Schema documents.

Schema Loading Details

SchemaLoader

Load Schema

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

These formats are currently supported for Schemas:

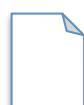
- application/wrml (JSON)
- JSON Schema
- Java Interface



Parse JSON



Parse JSON Schema



Java Interface
(.class)

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.



WRML Schema

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.



Java Interface
(.class)

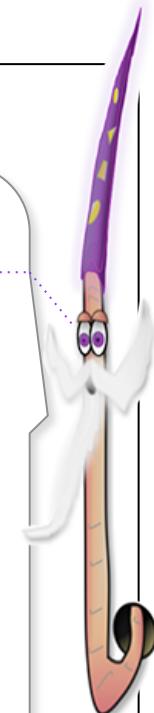
Bytecode Generation

One-time Reflection

Name	Type

Prototype

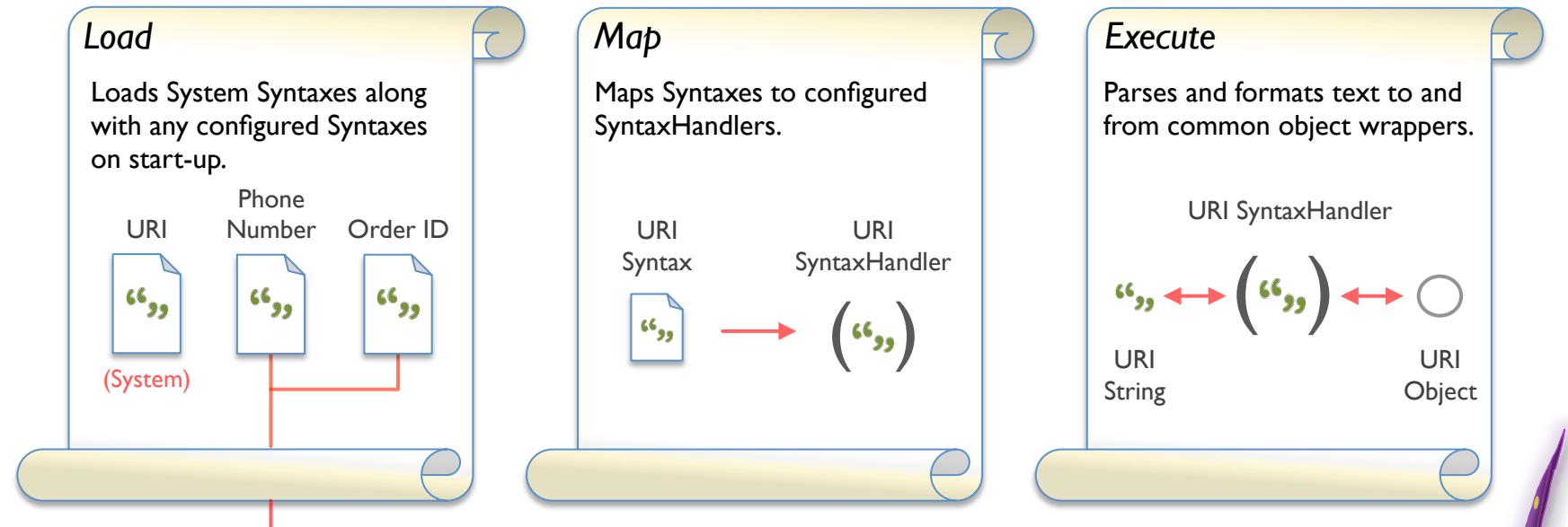
One-time Reflection



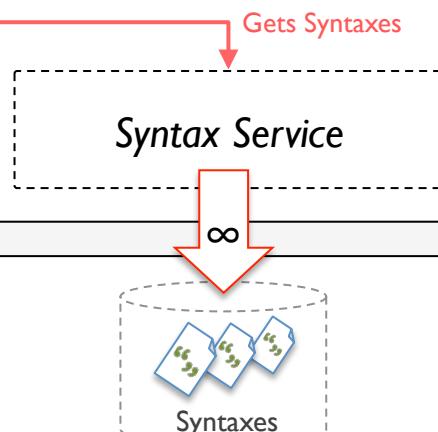
SyntaxLoader

Context

SyntaxLoader



ServiceLoader



At runtime, a Syntax is used to automate conversion of text values to and from “wrapper” objects.

Any Service can be configured to handle Syntax documents.

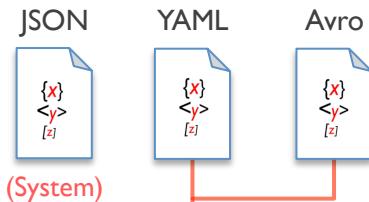
FormatLoader

Context

FormatLoader

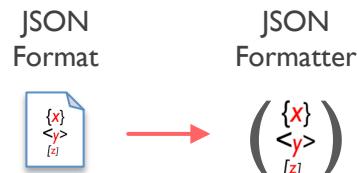
Load

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



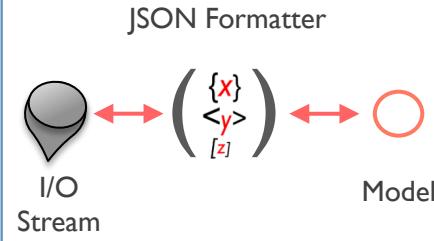
Map

Maps Formats to configured Formatters.



Execute

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



ServiceLoader

Gets Formats

Format Service

A Format describes a model serialization technique.

Any Service can be configured to handle Format documents.

Service Interface

Context

ServiceLoader

Service

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 SearchCriteria.



Save

Save (PUT) the specified model.



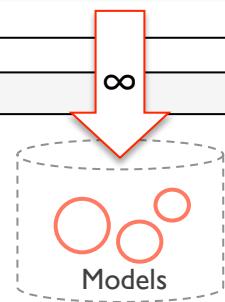
Delete

Delete a model based on the specified Keys and Dimensions.



Invoke

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



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

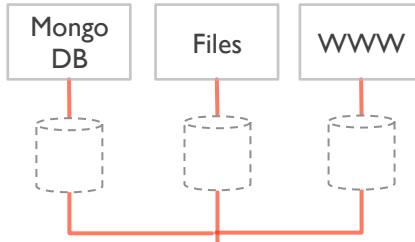
ServiceLoader

Context

ServiceLoader

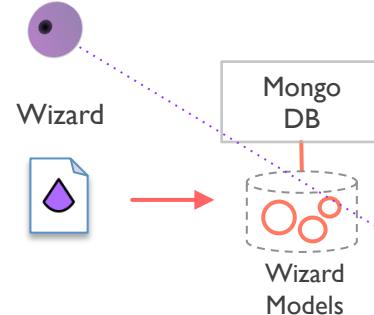
Load

Loads all configured Services on start-up.



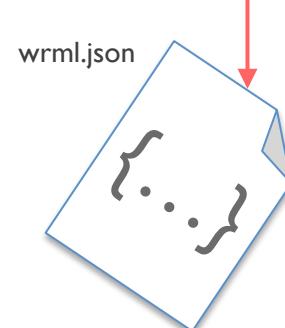
Map

Maps configured Schema *URI patterns* to configured Services.

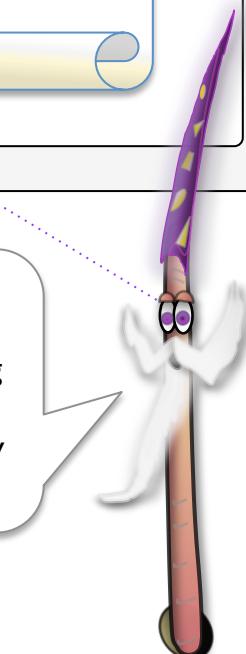


Execute

Services are used to implement the CRUD, search, and invoke operations for a set of models with a variety of Schemas.



URI patterns with wildcards are supported in the Schema-to-Service mapping so that "groups" of related Schemas may be handled by the same Service.



WRML REST API Design Tools

Wrmldoc Web-based GUI

WRML

User ▾ Model ▾ Edit ▾ View ▾ History ▾ Bookmarks ▾ Help ▾

fireball

Spell

Key Properties

“`url`
http://wizard.example.api.wrml.org/spells/fireball

“`spellId`
fireball

Other Properties

“`cacheTag`

#`secondsToLive`

Links

↳ `self`
href: http://wizard.example.api.wrml.org/spells/fireball GET
rel: self Open

Metadata

Schema
Spell Open

API
Wizard API Open

Resource
http://wizard.example.api.wrml.org/spells/{spellId} Open

WRML - Web Resource Modeling Language
Copyright (C) 2011 - 2013 Mark Masse (OSS project WRML.org)
Licensed under the Apache License, Version 2.0.

Werminal Command-line GUI

Werminal - Open Model

Schema (URI): http://schema.api.wrml.org/org/wrml/test/wizard/Spell < Enter Keys... >

History: [] http://schema.api.wrml.org/org/wrml/model/Abstract
[] http://schema.api.wrml.org/org/wrml/model/Described
[] http://schema.api.wrml.org/org/wrml/model/Filed
[] http://schema.api.wrml.org/org/wrml/model/MayBeReadOnly
[] http://schema.api.wrml.org/org/wrml/model/Nameable
[] http://schema.api.wrml.org/org/wrml/model/Tagged
[] http://schema.api.wrml.org/org/wrml/model/Titled
[] http://schema.api.wrml.org/org/wrml/model/UniquelyNamed
[] http://schema.api.wrml.org/org/wrml/model/Versioned

Spell Keys:

Document (URI): http://wizard.example.api.wrml.org/spells/fireball < History... >

Spell.spellId (String): fireball < History... >

< OK > < Cancel >

Werminal - Model - Spell

< Close > < Print Preview > < Set Origin > < Save >

Schema: http://schema.api.wrml.org/org/wrml/test/wizard/Spell
Heap ID: fb5fdf3-bc23-473f-9ed0-26580dbfe793
Origin: Mongo

Slots

spellId (Text): fireball
uri (Text/URL): http://wizard.example.api.wrml.org/spells/fireball
cacheTag (Text):
delete (Link): href=""
save (Link): href=""
secondsToLive (Long):
self (Link): href="http://wizard.example.api.wrml.org/spells/fireball" < Click... >