Writing REST APIs with OpenAPI and Swagger Ada

Stéphane Carrez  FOSDEM 2018
OpenAPI and Swagger Ada

- Introduction to OpenAPI and Swagger
- Writing a REST Ada client
- Writing a REST Ada server
- Handling security with OAuth2
- Demo
30 years of RPC

- Sun RPC (RFC 1057) in 1988
- CORBA IDL in 1991
- Ada95 Distributed Annex E in 1995
- Java RMI in 2000
- WSDL and SOAP in 2000
- Google gRPC with Protocol Buffers since 2001
30 years but same goals

- Simplify the developer’s job
- Describe the protocol between client & server
- Generate client stubs and server skeleton
- Handle and hide communication details
- Document the client & server interaction

https://github.com/stcarrez/swagger-ada
Why REST and OpenAPI?

- REST as an alternative to SOAP since 2000 (Roy Thomas Fielding)
- Easier to use, write, implement, debug
- Can easily be used from browsers
- Increasing usage of REST with mobile applications
- Need for description, documentation
- Need for client language bindings
OpenAPI Specification

- Started in 2010 to describe REST APIs
- OpenAPI Initiative created in Nov 5, 2015 (Google, Microsoft, IBM, Paypal, ...)
- OpenAPI 3.0 released July 26, 2017
- https://github.com/OAI/OpenAPI-Specification

https://github.com/stcarrez/swagger-ada
# OpenAPI 2.0 Document Structure

- **info**
- **host**
- **basePath**
- **schemes**
- **produces**
- **consumes**
- **paths**
- **tags**
- **externalDocs**
- **definitions**
- **parameters**
- **responses**

## YAML or JSON file with well defined keywords

YAML or JSON file with well defined keywords.

## Describes security aspects

Describes security aspects.

## What the API accepts as input, what it produces

What the API accepts as input, what it produces.

## Describes REST APIs paths, operations, can reference definitions, parameters, responses

Describes REST APIs paths, operations, can reference definitions, parameters, responses.

## Describes data types, parameters, responses

Describes data types, parameters, responses.
OpenAPI benefits

Documentation
Online Documentation
Client Binding
Server Skeleton
Server Configuration
API Validation

https://github.com/stcarrez/swagger-ada
Swagger: Tools for OpenAPI

- Online Editor: Swagger Editor
- Generator: Swagger Codegen
- Documentation: Swagger UI
- Sources: https://github.com/swagger-api

https://github.com/stcarrez/swagger-ada
Swagger Editor: https://editor.swagger.io
Swagger Codegen

OpenAPI Document

{ YAML JSON }

Swagger Codegen

...}

API Doc (HTML)

Ada REST Client

Ada REST Server

Java REST Client

Java REST Server

Python REST Client

Python Flask Server

25+ Programming Languages

https://github.com/stcarrez/swagger-ada
Writing a REST Ada client

- Get the OpenAPI/Swagger description file
  - From SwaggerHub: https://swaggerhub.com/
  - From APIs.guru: https://apis.guru/openapi-directory/
- Generate the Ada client code
- Use the generated code to make API calls

https://github.com/stcarrez/swagger-ada
OpenAPI: Info description (1/3)

- YAML or JSON file
- General purpose description of the API
- Describe the service entry point

```yaml
swagger: "2.0"
info:
  version: "1.0"
  title: "Todo API"
  contact:
    email: Stephane.Carrez@gmail.com
  license:
    name: Apache 2.0
    url: 'http://www.apache.org/licenses/LICENSE-2.0.html'
host: localhost:8080
basePath: /v1
tags:
  - name: tasks
    description: Operations to manage tasks
schemes:
  - https
  - http
```

https://github.com/stcarrez/swagger-ada
OpenAPI: REST operation (2/3)

- **Describe the REST operations**

paths:
  /todos:
  get:
    tags:
      - tasks
    summary: List the available tasks
    description: List the available tasks
    operationId: listTodos
    produces:
      - application/json
    parameters:
      - name: status
        in: query
        description: Filters the task by their status
        required: false
        type: string
        enum:
          - done
          - waiting
          - working
          - all
    responses:
      '200':
        description: successful operation
        schema:
          type: array
          items:
          $ref: '#/definitions/Todo'
      '400':
        description: Invalid status value

https://github.com/stcarrez/swagger-ada
OpenAPI: Model definitions (3/3)

```
definitions:
  Todo:
    type: object
    properties:
      id:
        type: integer
        format: int64
        description: The todo identifier
      title:
        type: string
        description: The todo title
      create_date:
        type: string
        format: date-time
        description: The todo creation date
      done_date:
        type: string
        format: date-time
        description: The todo resolution date
      status:
        type: string
        description: The todo state
        enum:
          - waiting
          - done
      required:
        - id
        - title
        - status
        - create_date
```
Client: let’s generate the code!

- Generate the client code with Swagger Codegen

```
$ java -jar swagger-codegen-cli.jar generate -l ada -i todo.yaml \\
    -DprojectName=Todos --model-package Todos
```

```
Client API: package Todos.Clients

Model API: package Todos.Models

Sample: procedure Todos.Client

GNAT project
```

[Image of project structure]

3 directories, 9 files

https://github.com/stcarrez/swagger-ada
Ada REST Client

Client Application

Generated code

Swagger runtime

Brings security with OAuth2 support

Brings JSON/XML serialization deserialization and more

Choose between libcurl or AWS

https://github.com/stcarrez/swagger-ada
Client and Server Data Model

- Data types described in the Models package
- Same Models Ada package for client and server
- Operations to serialize and deserialize (JSON/XML)

package Todos.Models is
  type Todo_Type is record
    Id            : Swagger.Long;
    Title         : Swagger.UString;
    Create_Date   : Swagger.Datetime;
    Done_Date     : Swagger.Nullable_Date;
    Status        : Swagger.UString;
  end record;
package Todo_Type_Vectors is
  new Ada.Containers.Vectors
    (Positive, Todo_Type);
end Todos.Models;

Todo:
  type: object
  properties:
    id:
      type: integer
      format: int64
      description: The todo identifier
    title:
      type: string
      description: The todo title
    create_date:
      type: string
      format: date-time
      description: The todo creation date
    done_date:
      type: string
      format: date-time
      description: The todo resolution date
    status:
      type: string
      description: The todo state
      enum:
        - waiting
        - done

https://github.com/stcarrez/swagger-ada
Client API

- Represented by the Client_Type tagged record
- Provides operations described by the OpenAPI
- Allows to control the API call (headers, security)

```ada
package Todos.Clients is
    type Client_Type is
        new Swagger.Clients.Client_Type with null record;

    procedure Create_Todo (Client : in out Client_Type;
                            Title : in Swagger.Ustring;
                            Result : out Todos.Models.Todo_Type);

    procedure List_Todos (Client : in out Client_Type;
                          Status : in out Swagger.Nullable_UString;
                          Result : out Todos.Models.Todo_Vector);
end Todos.Clients;
```

https://github.com/stcarrez/swagger-ada
Calling REST in Ada

- Declare a `Client_Type` instance
- Configure it (server URL, credentials)
- Call the operation with its parameters

```ada
with Todos.Clients;
with Todos.Models;
...
Client : Todos.Clients.Client_Type;
List   : Todos.Models.Todo_Type_Vectors.Vector;
Empty  : Swagger.Nullable_String := (Is_Null => True, Value => <>);
...
Client.Set_Server ("http://localhost:8080/v1");
Client.List_Todos (Empty, List);
```
Writing a REST Ada server

- Write the OpenAPI/Swagger description file
- Generate the Ada server code
- Implement the server operations
- Share the OpenAPI description on SwaggerHub!
**Server: let’s generate the code!**

- **Generate the server code with Swagger Codegen**

```
$ java -jar swagger-codegen-cli.jar generate -l ada-server -i todo.yaml \
-DprojectName=Todos --model-package Todos
```

![Directory structure](https://github.com/stcarrez/swagger-ada)

- **Model API**: `package Todos.Models`
- **Server skeleton**: `package Todos.Skeletons`
- **Server**: `procedure Todos.Server`
- **Server code**: `package Todos.Servers`
- **GNAT project, server configuration file**

[https://github.com/stcarrez/swagger-ada](https://github.com/stcarrez/swagger-ada)
Ada REST Server

Server Application

Server Skeleton & Model

Swagger Ada

Ada Security
Ada Servlet
Ada Utility Library
XML/Ada
AWS

Your server code and application
Generated code
Swagger runtime
Brings REST server support with security and OAuth2 support on server side

https://github.com/stcarrez/swagger-ada
Server Skeleton

- Declares the `Server_Type` limited interface to describe the operations
- Additional `Context_Type` object gives access to request, response
- Two generic packages for server skeleton provide two server models:
  - Instance per request
  - Global shared instance within a protected object

```ada
package Todos.Skeletons is
  type Server_Type is limited interface;

  procedure Create_Todo
  (Server   : in out Server_Type;
   Title    : in  Swagger.Ustring;
   Result   : out Todos.Models.Todo_Type;
   Context  : in out Swagger.Servers.Context_Type) is abstract;
...
end Todos.Skeletons;
```

https://github.com/stcarrez/swagger-ada
Server Implementation (1/2)

- Implement the `Server_Type` interface with its operations
- Populate `Result` or use the `Context` to send an error
- Serialization/Deserialization handled by the skeleton

```ada
package Todos.Servers is
  type Server_Type is limited new Todos.Skeletons.Server_Type ...

  overriding procedure Create_Todo
    (Server : in out Server_Type;
     Title  : in   Swagger.Ustring;
     Result : out Todos.Models.Todo_Type;
     Context: in out Swagger.Servers.Context_Type);
...
end Todos.Servers;
```

https://github.com/stcarrez/swagger-ada
Server Implementation (2/2)

- Instantiate one of the two server skeletons (per-request model or shared model)

```ada
package Todos.Servers is
    ...
    package Server_Impl is
        new Todos.Skeletons.Shared_Instance (Server_Type);
    end Server_Impl;
end Todos.Servers;
```

- Register the OpenAPI to the application

```ada
procedure Todos.Server is
begin
    ... Todos.Servers.Server_Impl.Register (App);
end Todos.Server;
```

https://github.com/stcarrez/swagger-ada
OpenAPI: Describing security

- Describe security endpoints
- Describe security scopes
- Assign required security scopes to operations

Security:
- todo_auth: []

securityDefinitions:
todo_auth:
  type: oauth2
  flow: password
  tokenUrl: /v1/oauth/token
  scopes:
    'write:todo': Write a todo
    'read:todo': Read a todo

paths:
/todos:
  get:
    ...
    parameters:
    ...
    responses:
    ...
    security:
      - todo_auth:
        - 'read:todo'
Client Security with OAuth2 (1/2)

- Create and initialize a credentials object
- Obtain the access token and optional refresh token
- Configure the client to use the credentials

```ada
with Swagger.Credentials.OAuth;

Cred : aliased Swagger.Credentials.OAuth.OAuth2_Credential_Type;
...
  Cred.Set_Application_Identifier ("todo-app");
  Cred.Set_Application_Secret ("todo-app-secret");
  Cred.Set_Provider_URI ("http://localhost:8080/v1/oauth/token");
  Cred.Request_Token (Username, Password, "read:todo write:todo");
...
  Client.Set_Credentials (Cred'Access);
```
Client Security with OAuth2 (2/2)

- Make API calls: credentials are passed on the wire within the ‘Authorization’ header

```
List : Todos.Models.Todo_Type_Vectors.Vector;
...
    Client.List_Todos (Empty, List);
```

```
GET /v1/todos HTTP/1.1
Host: localhost:8080
Authorization: Bearer 74rE0wU.d44CPAll_kyyB2krd8bYdVYWqgmtloIR.9zyiBM
Accept: application/json
```
Server security (1/2)

- Each OpenAPI scope represented by a permission definition (generated code):

```ada
package ACL_Read_Todo
    is new Security.Permissions.Definition ("read:todo");
```

- Authentication and permission check generated in the server skeleton (generated code):

```ada
if not Context.Is_Authenticated then
    Context.Set_Error (401, "Not authenticated");
    return;
end if;
if not Context.Has_Permission (ACL_Read_Todo.Permission) then
    Context.Set_Error (403, "Permission denied");
    return;
end if;
```

https://github.com/stcarrez/swagger-ada
Server security (2/2)

- Configure the server key for OAuth2 tokens:
  
  ```
  swagger.key=Y29naGk5SGkKUG9YaTdhaHgKYWlUaGl1M3UK
  ```

- Configure the server to register the client id and secret
  
  ```
  app.list=1
  app.1.client_id=todo-app
  app.1.client_secret=todo-app-secret
  app.1.scope=none
  ```

- Configure the users allowed to authenticate
  
  ```
  users.list=1,2
  users.1.username=admin
  users.1.password=admin
  users.2.username=test
  users.2.password=test
  ```
Demo: Todo client

```ada
begin
    Configure "client.properties", C, Cred);

    C.Set_Credentials (Cred'Access);

    if Arg_Count = 2 and Command = "add" then
        Title := Swagger.To_UString (Ada.Command_Line.Argument (2));
        C.Create_Todo (Title, Todo);
        Put_Line ("Created todo " & Swagger.Long'Image (Todo.Id));
        Print (Todo);
    elsif Arg_Count = 1 and Command = "list" then
        C.List_Todos ((Is_Null => True, Value => <>), List);
        for T of List loop
            Print (T);
        end loop;
    elsif Arg_Count = 2 and Command = "del" then
        C.Delete_Todo (Todo_Id => Swagger.Long'Value (Ada.Command_Line.Argument (2)));
    elsif Arg_Count = 2 and Command = "close" then
        C.Update_Todo (Todo_Id => Swagger.Long'Value (Ada.Command_Line.Argument (2)),
            Title => (Is_Null => True, Value => <>),
```
Demo: Todo server (main)
Demo: Todo server (impl)

```
use type Swagger.UString;

-- Create a todo
overriding
procedure Create_Todo
  (Server : in out Server_Type;
   Title : in Swagger.UString;
   Result : out Todos.Models.Todo_Type;
   Context : in out Swagger.Server)
pragma Un referenced (Context);
begin
  Result.Id := Server.Next_Id;
  Result.Create_Date := Ada.Calendar.Clock;
  Result.Status := Swagger.To_UString ("waiting");
  Result.Title := Title;
  Server.Next_Id := Server.Next_Id + 1;
  Server.Tasks.Insert (Result.Id, Result);
end Create_Todo;

-- Delete the todo
overriding
procedure Delete_Todo
  (Server : in out Server_Type;
   Todo_Id : in Swagger.Long;
   Context : in out Swagger.Servers.Context_Type) is
```

https://github.com/stcarrez/swagger-ada
Demo: Running the client

```
gprbuild: "todos-client" up to date

ciceron@zeus:~/work/vacs/ada-fosdem-2018/swagger-ada-todo$ bin/todos-client
Usage: todos {list|add|del|close|update} {params}
  list  List the todos
  add <title>  Add a todo
  del <id>  Delete the todo with the given <id>
  close <id>  Change the todo status to 'close'
  update <id> <title>  Update the todo title

ERROR: get request: Couldn't connect to server
Cannot connect to the server.
ciceron@zeus:~/work/vacs/ada-fosdem-2018/swagger-ada-todo$ bin/todos-client list
404 error received
```

https://github.com/stcarrez/swagger-ada
Limitations and improvements

- Ada Strings
  (need a lot of \texttt{To\_String}/\texttt{To\_Ustring} conversions)
- Enums are treated as strings
- Circular type dependencies not handled
- Error model needs some work
- Improvements:
  - Upload file support
  - Asynchronous client operation call
Conclusion

- OpenAPI describes REST APIs
- Swagger Codegen generates Ada code for you
- Swagger Ada is a runtime library for REST client and REST server implementation
- More than 500 APIs available, write your own!
- Next step: … GraphQL?