JARS - Sandcat

The JavaScript RESTful Specification

Product Technical Overview

(03/09/2017)

This is a GlassCat project presentation

1. What is JARS?

JARS stands for **JavaScript API for RESTful Services**.

It is a working draft specification project which aims to provide a standard REST API to JEC[1] *(e.g. GlassCat)*.

JARS is based upon the JAX-RS specification[2].

JARS is:

- easy-to-use
- built for maintainability
- container independent
- built on top of the TypeScript decorators specification
- asynchronous and non-blocking

The GlassCat default implementation of JARS is Sandcat.

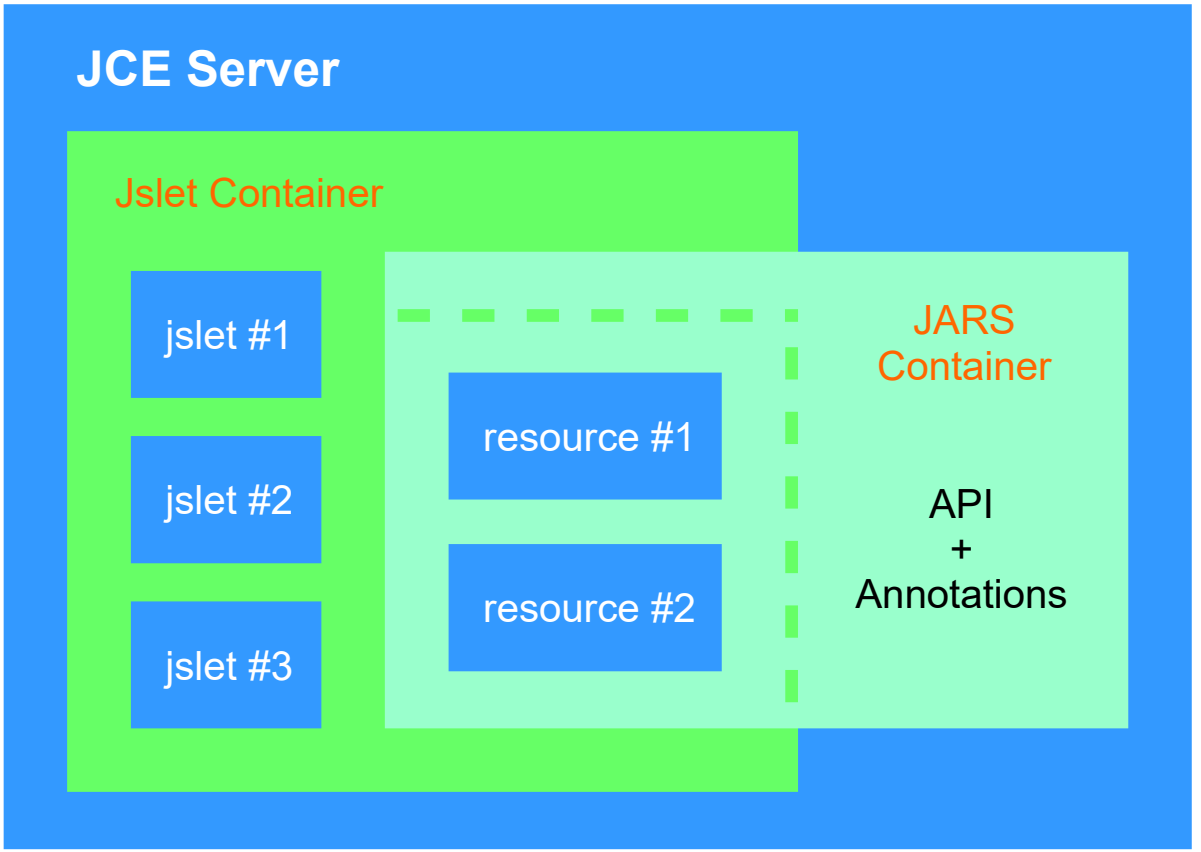
[1] JavaScript Entreprise Container [2] Java API for RESTful Services (JAX-RS)

2. Architecture

JARS is built over the JCE **Jslet API**.

It uses POJOs[1] to implement the corresponding Web resources.

JARS provides the same support for asynchronous and non-blocking I/O than jslets.



[1] Plain Old JavaScript Objects

3. The Modular Approach

All specifications that are part of the JCE specification are based on modular approach.

JARS implementations can be used independently of any container.

JARS supports autowiring.

The following piece of code is only what you need to initialize Sandcat in a GlassCat container:

```
import {BootstrapScript} from "../../../../server/com/onsoft/glasscat/domains/scripts/BootstrapScript";
import {DomainContainer} from "../../../../server/com/onsoft/glasscat/domains/containers/DomainContainer";
import {SandcatBuilder} from "../../../../server/com/onsoft/sandcat/builders/SandcatBuilder";

export class InitApp extends BootstrapScript {

  public run(container:DomainContainer):void {
    new SandcatBuilder().build(container)
                        .process((err:any)=>{});
  }
}
```

4. Summary of Annotations

Annotation	Target	Description
`@GET, @HEAD, @POST, @PUT, @DELETE, @CONNECT, @OPTIONS, @TRACE`	Method	Represent the HTTP requests that can be handled by a method.
`@ResourcePath`	Type	Specifies a relative path for a resource
`@Exit`	Field	The reference to the callback function for non-blocking support
`@Init, @Destroy`	Method	Provides access to the jslet initialization API.
`@PathParam`	Field	The value of a method parameter, extracted from the URI paths.
`@QueryParam`	Field	The value of a method parameter, extracted from the URI query parameters.
`@RequestParam`	Field	Provides access to the current HTTP request.
`@RequestBody`	Field	Provides access to the body of the current HTTP request.
`@RootPath`	Type	Specifies the resource-wide version path that forms the base URI of a set of resource classes.
`@RootPathRefs`	Type	Specifies the resference to all of the root paths for a resource class.
`@CookieParam`	Field	The value of a method parameter, extracted from the cookies.

5. Ease-of-use

JARS API is highly intuitive to learn and use.

It provides an easy support for multiple sub-routing, parameters extractions and MIME types treatments:

```
import {ResourcePath} from "../../../../server/com/onsoft/sandcat/annotations/ResourcePath";
import {GET} from "../../../../server/com/onsoft/sandcat/annotations/GET";
import {PathParam} from "../../../../server/com/onsoft/sandcat/annotations/PathParam";
import {Exit} from "../../../../server/com/onsoft/sandcat/annotations/Exit";

@ResourcePath("/hello")
export class Hello {

  @GET()
  public sayHelloWorld(@Exit exit:Function):void {
    exit("Hello World!");
  }

  @GET({
    route: "/:username"
  })
  public sayHelloUser(@PathParam username:string, @Exit exit:Function):void {
    exit(`Hello ${username}!`);
  }
}
```

6. REST APIs Versioning

Unlike JAX-RS, JARS provides support for REST APIs URL versioning.

The following sample code shows the declaration for the second version of a RESTful service:

```
@RootPath({
  path: "/versioned.api",
  ref: "v2.0",
  version: {
    prefix: "v",
    major: 2,
    minor: 0
  }
})
export class VersionedSampleApi_v_2_0 {}
```

You declare references to specific versions of an API as below:

```
@ResourcePath("/search")
@RootPathRefs(["v1.0", "v2.0"])
export class Search {}
```

Then, you can acces the ressource by using the following URLs:

```
http://mydomain.com/myservices/versioned.api/v1.0/search
http://mydomain.com/myservices/versioned.api/v2.0/search
```

GlassCat Creator

Technical expert on the Adobe Flash Platform, Pascal is a self-taught developer who focuses on implementing innovative solutions for building Rich Internet Applications.

He has created a Flex-like ActionScript GUI Framework (SPAS 3.0) and a Web Operating System (ONYX OS), with the aim to mix social networking, Web marketing and personal data managment.

His latest project (http://wooz.io) is a powerful JavaScript apps builder, for deploying real softwares in the cloud.

He wrote different XML language specifications to facilitate data exchanges over the Internet, such like Press Releases, or Curriculum Vitæ...

Email: pechemann@gmail.com
Linkedin: www.linkedin.com/in/echemann
Place of residence: French Riviera, GMT+1