Web Resource Platform

Kevin Grimes, Jordan Padams, and Galen Hollins

Jet Propulsion Laboratory, California Institute of Technology

Contact Email: kevin.m.grimes@jpl.nasa.gov

Introduction

- Enables data and application sharing across disparate domains
- Flexible and extensible, allowing data products and applications to be modified dynamically in a distributed fashion
- Comprised of multiple disparate components which can be interacted with via a RESTful API

Three Tools, One Platform

- Product Repository (Juneberry): exposes data products over a network
- Resource Discovery (Cotinga): enables efficient and accurate search
- Tools Service: allows users on network to run tools installed on server

Product Repository – Juneberry

Description

- Makes data products uniformly accessible ("webifies them") via REST calls
- Exposes a directory of data files that exist on a user's file system to other users on the same network
- Allows users to interact with components of the files
- Plug-ins can be created for different product types, allowing individual components of exposed products of these types to be manipulated using simple web requests

Use Case – `https://pds-imaging.jpl.nasa.gov/w10n`

- PDS products along with their respective labels are exposed for download
- Enables server-side image manipulation (e.g. resizing and cropping)
- Provides subsetting for raw raster data

Examples

Suppose a data product *foo* exists on the server *host.* Further assume that *foo* exists in a webified directory *dir* and that the data type of *foo* is Graphics Interchange Format (GIF). The product *foo* may then be retrieved by a user on the network with the following REST call:

```
http://host:8080/dir/foo/0/image[]?output=gif
```

Since the image is in a format that Juneberry supports, the user may perform various actions upon it, including resizing:

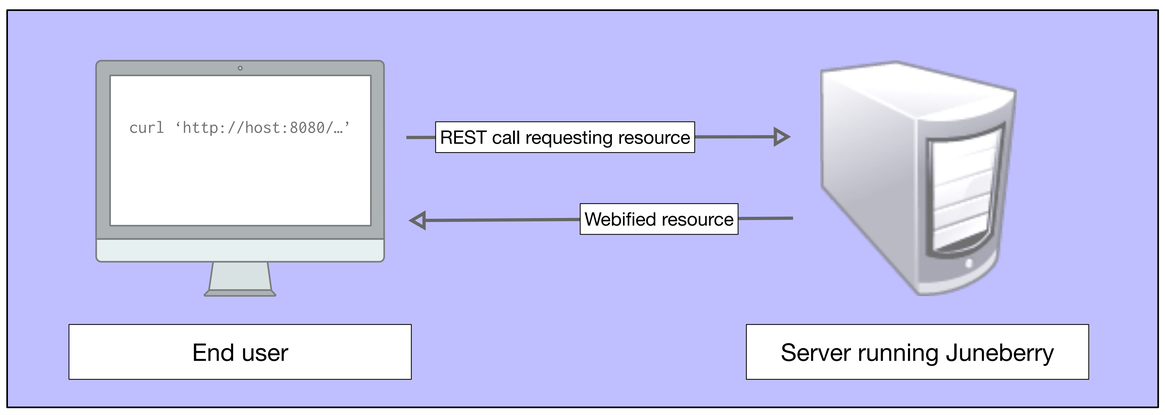
```
http://host:8080/dir/foo/0/resized/512x256[]?output=gif
```

Users may also obtain the raw raster data of the image. The following REST call would return the entire image as raw data:

```
http://host:8080/dir/foo/0/raster/data[]?output=json
```

The user may also request a subset of the image's data. For example, the following call would retrieve only the first 10 columns of the first row of data:

```
http://host:8080/dir/foo/0/raster/data[0:1,0:10,0:1]?output=json
```



Resource Discovery – Cotinga

Description

- Indexing tool that leverages Apache Lucene
- Crawls a directory of data products that have been webified by Juneberry
- Builds an index from exposed data products and their respective labels
- Enables faceted querying of the Juneberry product repository
- No need for complicated schema
- Only knowledge required is a basic understanding of Lucene query syntax

Tools Service

Description

- Service installed on machine where tools (applications) are also installed
- These tools are exposed such that they can be invoked by a remote client somewhere else on the network
- Data is returned via HTTPS to the client
- Tools Service also provides useful metadata about these tools, including usage (e.g. expected inputs and outputs)

Use Case – Video Image Communication and Retrieval (VICAR)

- VICAR is an immense software suite (~4 GB) used to manipulate images of varying origin, including Mars and Jupiter
- In order to run, VICAR needs several external libraries whose sizes sum to about 100 GB
- Therefore, in order to use VICAR in its entirety, users must obtain over 100 GB of software; additionally, they must understand how to configure it (e.g. setting various environment variables) and must have the proper machine (must be Unix or Unix-like)
- Installing VICAR in a central location and allowing users on the network to interact with it via well-defined REST calls alleviates both the learning curve and the storage concern

Example

Suppose that the Tools Service is configured on server *host* to run tool *app*. Users may then obtain usage for the tool via the following command:

```
http://host:8080/ts/app/
```

The response is returned in JSON:

```
{ "name": "app", … , "usage": { "arguments": "<String:yourName>" }, … }
```

Once an argument is provided for `yourName`, a return value (rv) is returned:

```
http://host:8080/ts/app?yourName=John
```

Once again, the response is returned in JSON:

```
{ "application": "app-0.0.1", … , "result": { "rv": "Hello, John!", … }, … }
```