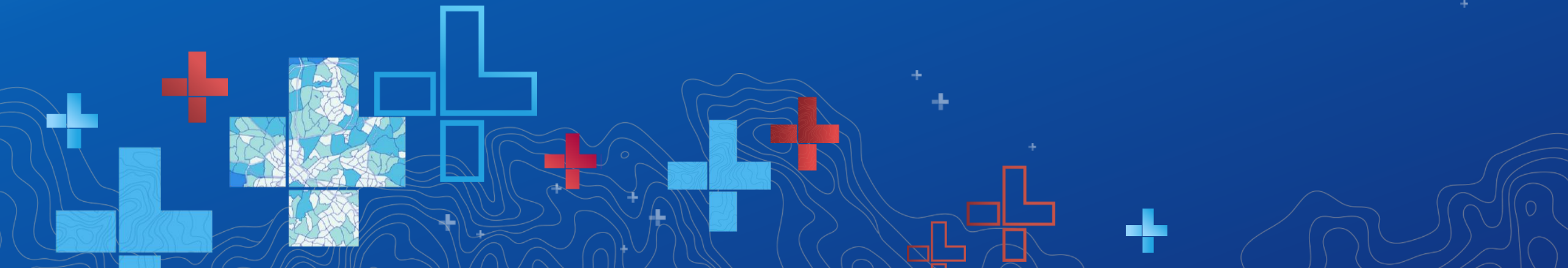
ArcGIS REST API: Advanced Techniques

Bill Major and Gary Sheppard

2020 ESRI FEDERAL GIS CONFERENCE | WASHINGTON, D.C.



ArcGIS REST capabilities

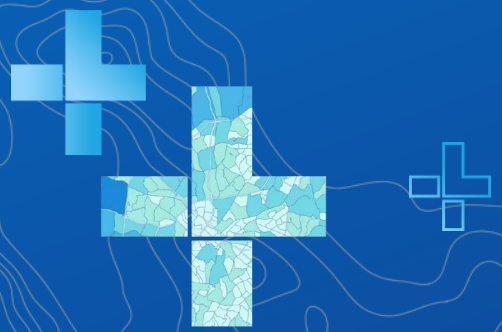
With a few examples

- **Location-based services***
  - Where is 123 Elm Street?
  - What's the fastest route to an incident?
  - What kind of customers live in a place?
- **Traditional GIS***
  - Visualization
  - Data management
  - Analysis
- **Users, groups, and items***
  - Which web maps do I own?
  - Share an app with Alpha Squad
  - Publish a hosted feature service
- **Enterprise administration**
  - Is my Portal for ArcGIS site healthy?
  - Which map and feature services get used the most?
  - Add new machines to my ArcGIS Server or Portal for ArcGIS site (i.e. horizontal scaling)

** Available in both ArcGIS Online (cloud SaaS) and ArcGIS Enterprise (cloud and on-premises)*

REST in any language

Accessing ArcGIS with or without an SDK or API



Direct SDK/API support for ArcGIS REST

ArcGIS Runtime

- Android (Java, Kotlin)
- iOS (Objective-C, Swift)
- Java (Linux, macOS, Windows)
- .NET (C#) (Android, iOS, UWP, WPF, Xamarin Forms)
- Qt (C++, QML) (Android, iOS, Linux, macOS, Windows)

ArcGIS API for Python

ArcGIS API for JavaScript (HTML5)

ArcGIS REST JS (HTML5, Node.js)

- Open source on GitHub
- Not officially supported

ArcGIS REST in other languages and platforms

- REST is language-agnostic
- Requirements:
  - HTTPS
  - HTTP GET
  - HTTP POST

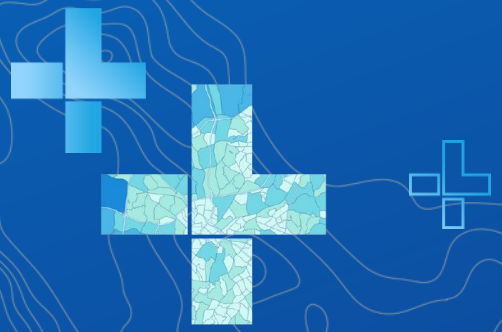
- Authentication:
  - OAuth 2.0
  - Token-based
  - Public key infrastructure (PKI)*
  - Integrated Windows Authentication (IWA)*

** ArcGIS Enterprise only*

Let’s focus on three approaches

- ArcGIS API for Python
- ArcGIS REST JS
- Going it alone with a different language

Python and the REST API



Python and the REST API

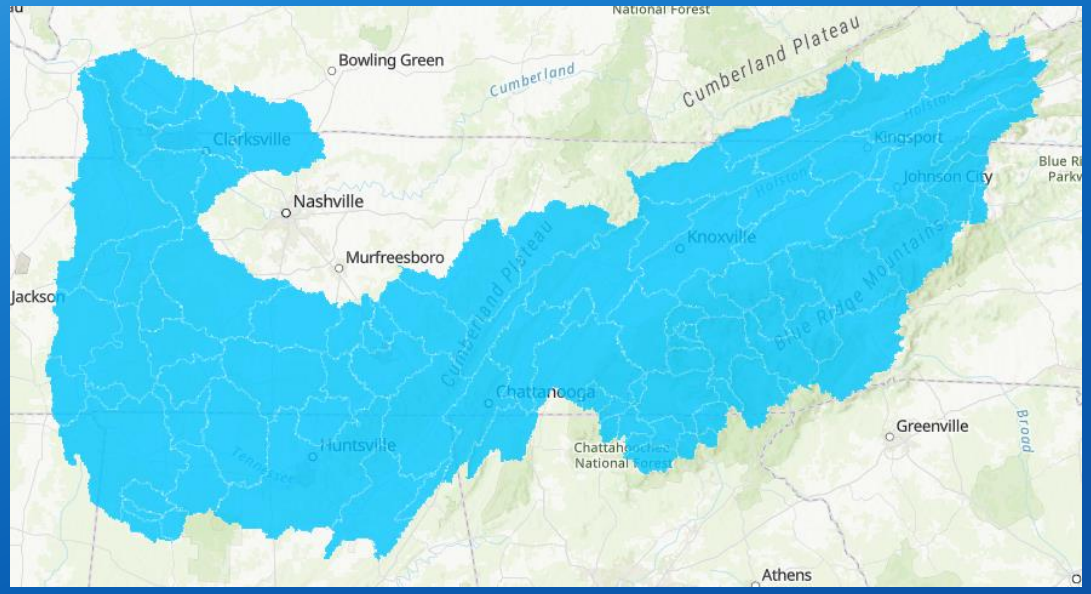
- Let’s take a tour of the REST API using Chrome Dev Tools, Fiddler, pure Python (just a little cheating), and how the Python API makes it so easy

Publishing a Hosted Feature Service

- Home app simplifies the process for you of publishing a zipped Shapefile
- BUT, there are several REST API calls to achieve this.
- /additem – adds the ZIP as an item
- /publish – publishes the ZIP as a Hosted Feature Service
- /status – check status of the HFS publication process to see when complete
- Optionally, /update to update properties on the HFS item, for example thumbnail

Publishing a Hosted Feature Service

- For this investigation, let’s look at /addItem and /publish
- /additem REST API
  - https://developers.arcgis.com/rest/users-groups-and-items/add-item.htm
- /publish REST API
  - https://developers.arcgis.com/rest/users-groups-and-items/publish-item.htm



Publish a HFS with Python

Presenter(s)

Performing Analysis a Hosted Feature Service

- Let’s say we now want to perform analysis on this HFS
- /createService all to Portal
- /submitJob to ArcGIS Server
- /update the Item with information
- Optionally, refresh….

Restarting Map Services

- Stop and start Map Services
- More of an Administrative workflow
- Uses the /arcgis/admin/services Admin API

Automatically Executing a Notebook

- /executeNotebook to a Notebook Server
- Then, check for job status at /system/jobs

Don’t worry Linux Command Line folks…

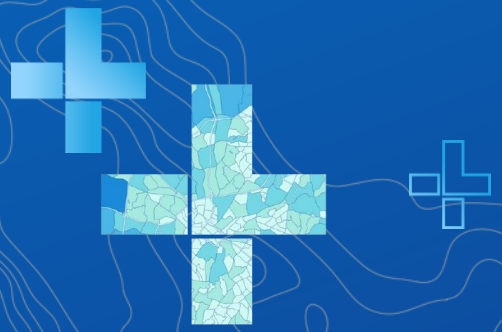
- **Curl works too!**

curl -X POST -d username=sharing1 -d password=super.secret -d referer=https://feddev.dev.geocloud.com -d f=json https://feddev.dev.geocloud.com/portal/sharing/rest/generateToken

ArcGIS REST JS

JavaScript wrappers for the ArcGIS REST API

https://esri.github.io/arcgis-rest-js/



ArcGIS REST JS vs. ArcGIS API for JavaScript

- **ArcGIS API for JavaScript**
  - Abstracts away the ArcGIS REST calls
  - Feature complete and supported
  - Includes an HTML5/Dojo map control
  - Runs only in a web browser
- **ArcGIS REST JS**
  - Binds to individual ArcGIS REST calls
  - Supports a subset of ArcGIS REST API and not supported
  - Includes no visual controls and does not include Dojo
  - Runs in a web browser and also in Node.js (i.e. apps and servers)

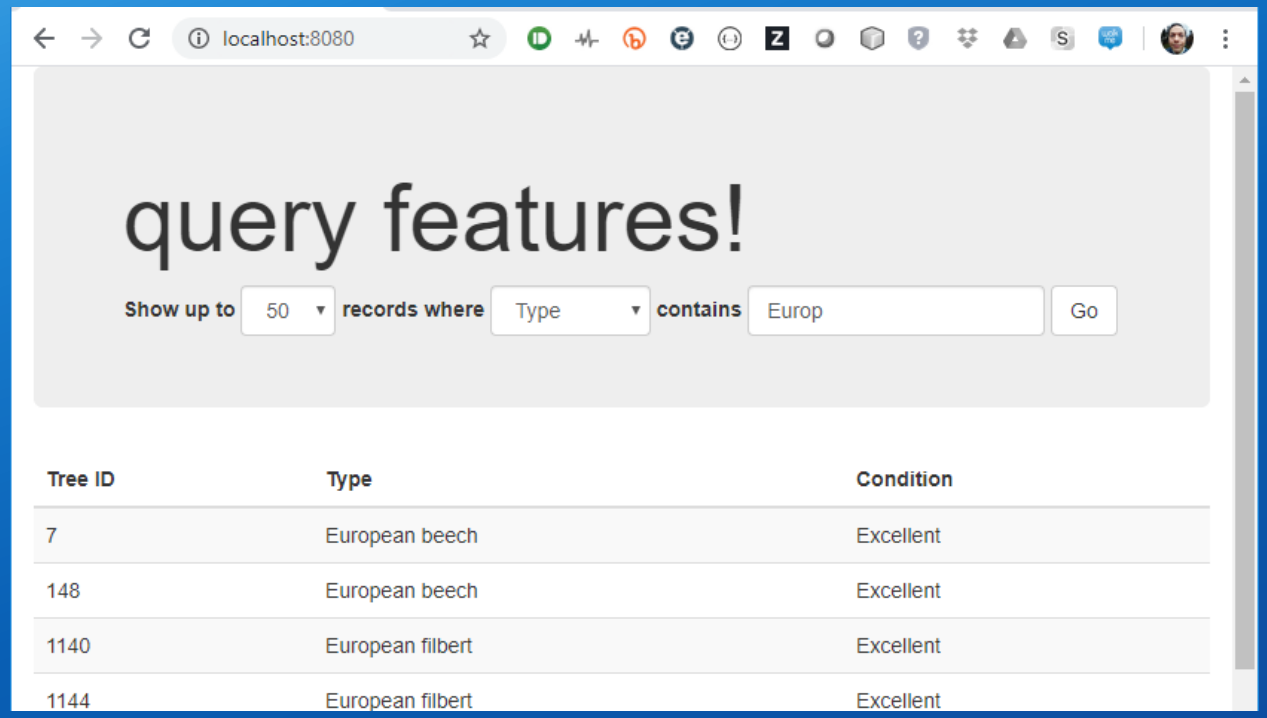
ArcGIS REST JS: easier than plain JavaScript

ArcGIS REST JS:

```
import { getUser } from "@esri/arcgis-rest-portal";
// pass in a username and get back information about the user
getUser(`gsheppard`)
  .then(response) // {firstName: "Gary", description: “engineer", ...}
```

JavaScript without ArcGIS REST JS:

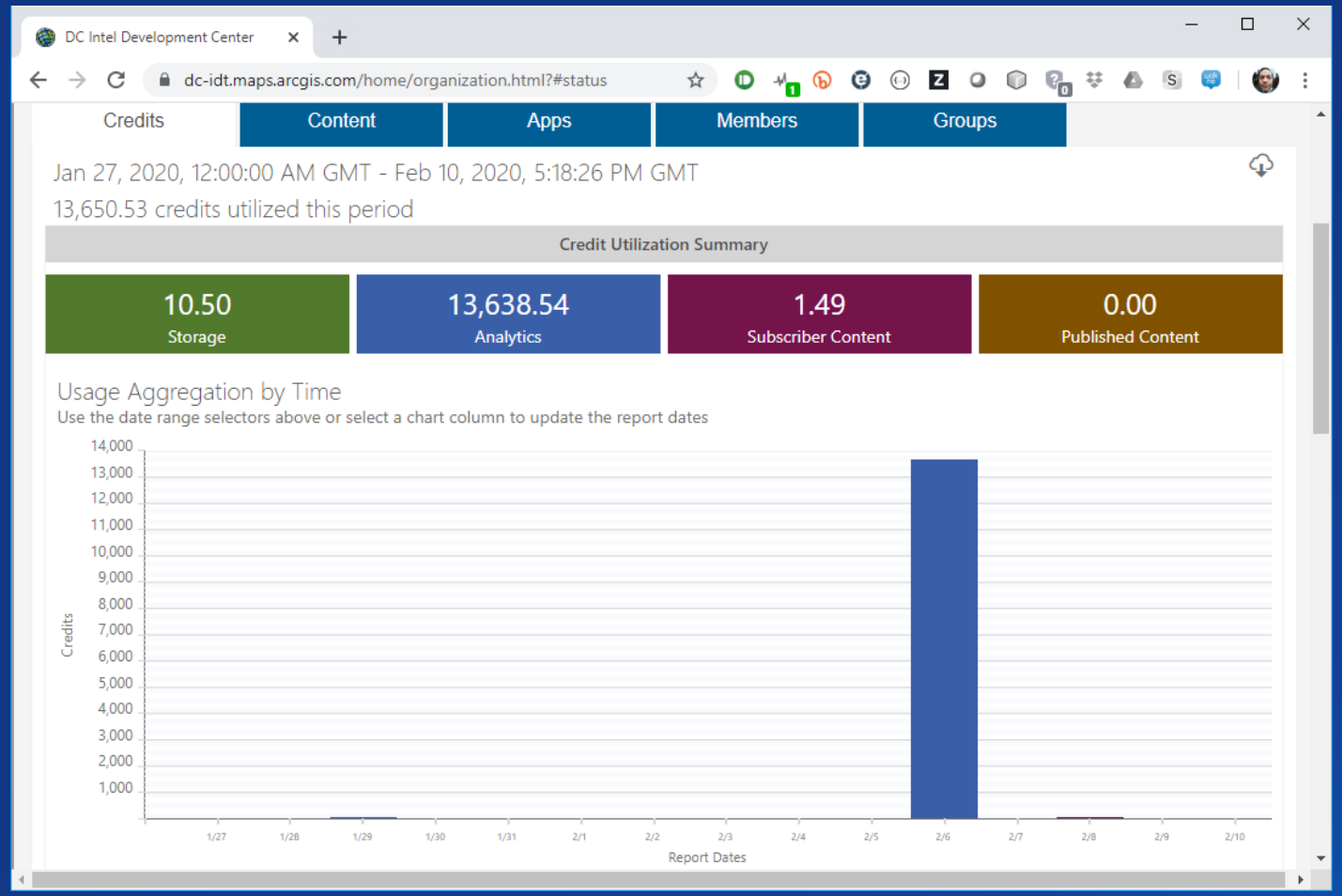
```
// construct the url yourself and don't forget to tack on f=json
const url = "https://www.arcgis.com/sharing/rest/community/users/gshep?f=json";
var xhr = new XMLHttpRequest();
xhr.onreadystatechange = function() {
  if (xhr.readyState == XMLHttpRequest.DONE) {
    xhr.responseText; // {firstName: "Gary", description: "engineer", ...}
  }
}
xhr.open('GET', url, true);
xhr.send(null);
```



ArcGIS REST JS

Gary Sheppard

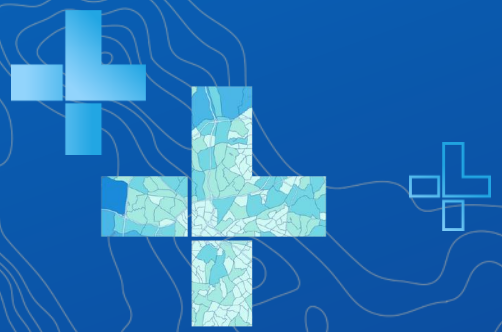
A word to the wise about ArcGIS Online premium services



This could be you!

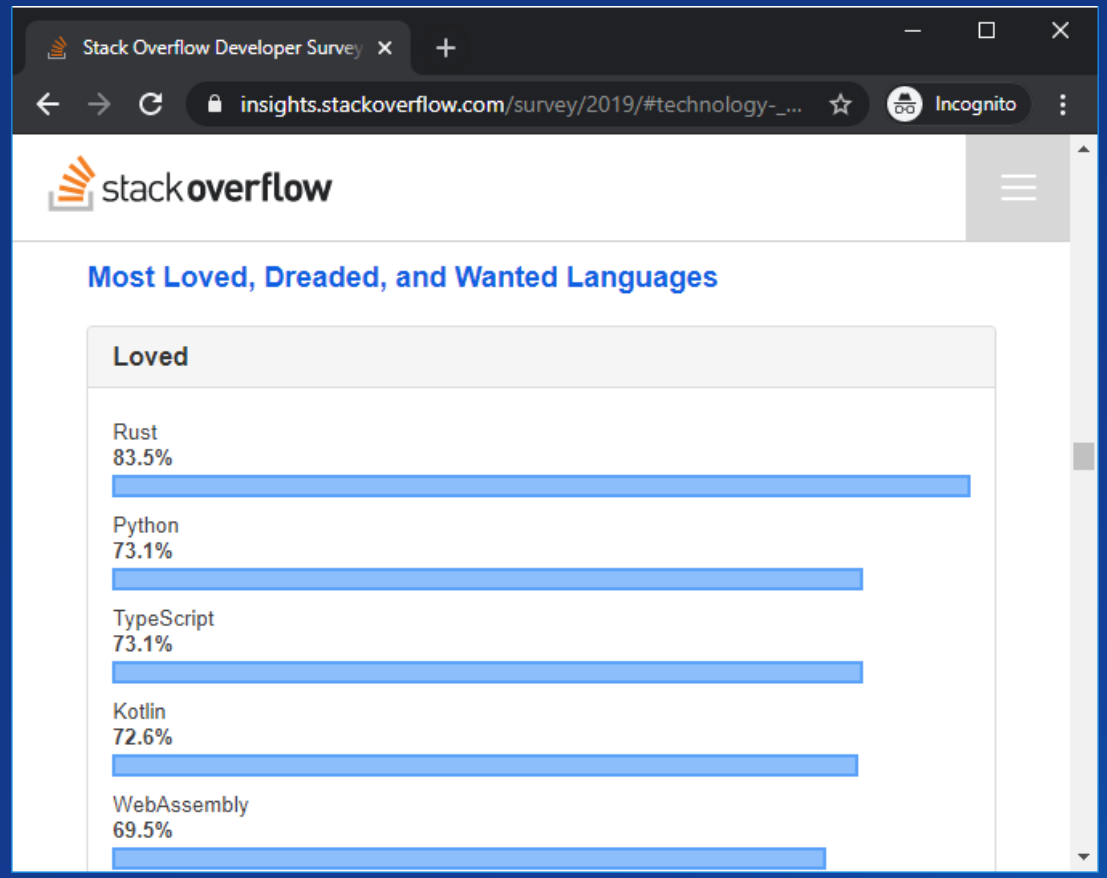
Going it alone

Using ArcGIS REST via the Rust programming language



Rust

- Curly-brace language
- Compiled
- Memory safety via ownership
- Libraries are called “crates”
- HTTP client crate: `reqwest`



https://insights.stackoverflow.com/survey/2019/

ArcGIS REST in Rust: token-based authentication

```
let mut params = HashMap::new();
let f_json = String::from("json");
params.insert("username", username);
params.insert("password", password);
params.insert("referer", referrer);
params.insert("f", &f_json);

match client.post("https://www.arcgis.com/sharing/rest/generateToken")
        .form(&params).send().await {
    Ok(response) => {
        match response.text().await {
            Ok(text) => Ok(json::parse(text.as_str()).unwrap()),
            Err(err) => Err(Box::new(err)),
        }
    },
    Err(err) => Err(Box::new(err)),
}
```

ArcGIS REST in Rust: feature service editing

```
let features_string = json::stringify(features);
let mut params = HashMap::new();
params.insert("f", "json");
params.insert("features", features_string.as_str());

match client.post(
    "https://services.arcgis.com/V6ZHFr6zdgNZuVG0/ArcGIS/rest/services/IncidentsRepor
t/FeatureServer/0/addFeatures"
).form(&params).send().await {
    Ok(response) => {
        match response.text().await {
            Ok(text) => Ok(json::parse(text.as_str()).unwrap()),
            Err(err) => Err(Box::new(err)),
        }
    },
    Err(err) => Err(Box::new(err)),
}
```

ArcGIS REST in Rust: buffer and query

Buffer

```
async fn buffer(client: &reqwest::Client, geometry_type: &str, geometry: JsonValue,
buffer_distance_m: &i32) -> BoxResult<JsonValue> {
    let geometries: String = json::stringify(object!{
        "geometryType" => geometry_type,
        "geometries" => array![ geometry ]
    });
    let distance_string = buffer_distance_m.to_string();
    let mut params = HashMap::new();
    params.insert("f", "json");
    params.insert("geometries", geometries.as_str());
    params.insert("inSR", "4326");
    params.insert("outSR", "4326");
    params.insert("distances", distance_string.as_str());
    params.insert("unit", "9001"); // meters
    params.insert("geodesic", "true");
```

To be continued…

ArcGIS REST in Rust: buffer and query

Buffer: the exciting conclusion

```
    match client.get(format!("{}/buffer",
GEOMETRY_SERVICE_URL).as_str()).query(&params).send().await {
        Ok(response) => {
            match response.text().await {
                Ok(text) =>
Ok(json::parse(text.as_str()).unwrap().take()["geometries"].take()[0].take()),
                Err(err) => Err(Box::new(err)),
            }
        },
        Err(err) => Err(Box::new(err)),
    }
}
```

ArcGIS REST in Rust: buffer and query

Query: set up parameters

```
let mut params = HashMap::new();
params.insert("f", "json");
params.insert("outFields", "*");
params.insert("where", match where_clause {
    Some(_where_clause) => _where_clause,
    _ => "0=0"
});
let stringified_geometry;
match geometry_type {
    Some(_geometry_type) => {
        match geometry {
            Some(_geometry) => {
                stringified_geometry = json::stringify(_geometry);
                params.insert("geometry", stringified_geometry.as_str());
                params.insert("geometryType", _geometry_type);
                params.insert("inSR", "4326");
            },
            _ => {}
        }
    },
    _ => {}
}
```

ArcGIS REST in Rust: buffer and query

Query: execute query

```
match client.post(
    format!("{}/query", feature_layer_url).as_str()
).form(&params).send().await {
    Ok(response) => {
        match response.text().await {
            Ok(text) => Ok(
                json::parse(text.as_str())
                    .unwrap().take()["features"].take()),
            Err(err) => Err(Box::new(err)),
        }
    },
    Err(err) => Err(Box::new(err)),
}
```

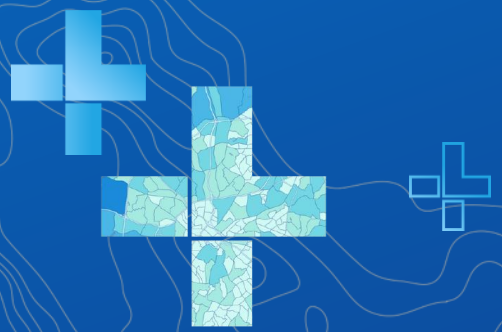
ArcGIS REST in Rust: creating items

```
let mut params = HashMap::new();
params.insert("token", token.as_str());
params.insert("referer", referrer.as_str());
params.insert("f", "json");
params.insert("type", "Color Set");
params.insert("typeKeywords", json::stringify(array!["My Custom Item Type Keyword"]).as_str());
params.insert("title", "An Item That My App Can Use");
params.insert("text", json::stringify(object!{
    "my_field_1" => "Value 1",
    "my_array_field" => vec_of_values
}).as_str());

let result = client
    .post(format!(
        "https://www.arcgis.com/sharing/rest/content/users/{}/addItem", &username
        ).as_str())
    .form(&params)
    .send()
    .await;
```

Outer rim of the galaxy

ArcGIS REST interfaces outside of Online, Server, and Portal



ArcGIS GeoEvent Server

- **GeoEvent Server itself:**
  - GeoEvent Server REST API documentation
  - Example: https://realtimegis.esri.com:6143/geoevent/rest
- **ArcGIS Server stream services:**
  - ArcGIS REST API: Stream Services
  - Discovering stream services (poor man's way)
    - Web search for "ArcGIS REST Services Directory" StreamServer
    - Tip: don't count on someone else's stream service!

ArcGIS Notebook Server

- A new server role in ArcGIS Enterprise
- Spatial analysis meets data science
- Server-side Python notebooks
- REST interface:
  - https://developers.arcgis.com/rest/enterprise-administration/notebook/overview.htm

ArcGIS Mission Server

- A new server role in ArcGIS Enterprise
- Situational awareness and mission management
- https://www.esri.com/mission
- About to be released with ArcGIS 10.8
- REST interface
  - Used in initial release
  - Not documented or officially supported in initial release
  - Subject to change!

ArcGIS REST resources

- Online help: https://developers.arcgis.com/rest/
- ArcGIS Enterprise help: links from Portal and Server REST pages
  - E.g. https://services.arcgisonline.com/arcgis/rest/, click “Help” or “API Reference”
  - E.g. https://maps.esri.com/portal/sharing/rest, click “API Reference”
- ArcGIS Tutorials: https://developers.arcgis.com/labs/?product=rest-api&topic=any
- ArcGIS API for Python: https://developers.arcgis.com/python/
- ArcGIS REST JS: https://esri.github.io/arcgis-rest-js/
- Rust REST examples: https://github.com/garysheppardjr/arcgis-rest-demos

Print Your Certificate of Attendance

Print Stations Located in 150 Concourse Lobby

Tuesday

12:30 pm – 6:30 pm
Expo
Hall B

5:15 pm – 6:30 pm
Expo Social
Hall B

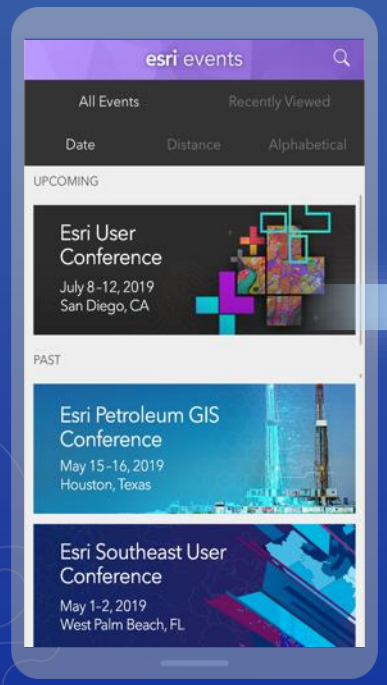
Wednesday

10:45 am – 5:15 pm
Expo
Hall B

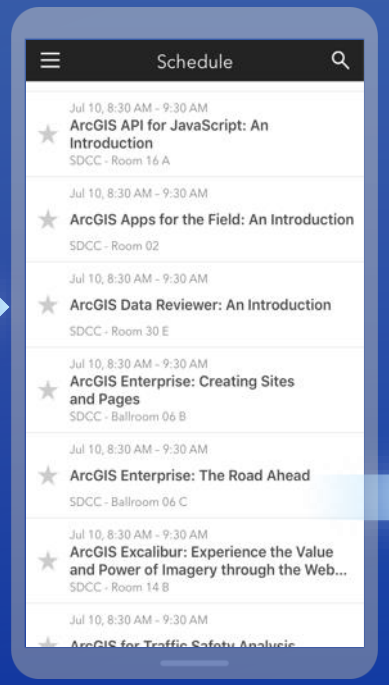
6:30 pm – 9:30 pm
Networking Reception
Smithsonian National Museum
of Natural History

Please Share Your Feedback in the App

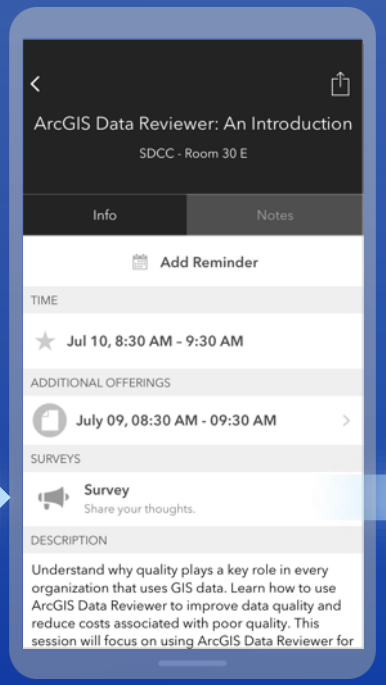
Download the Esri Events app and find your event



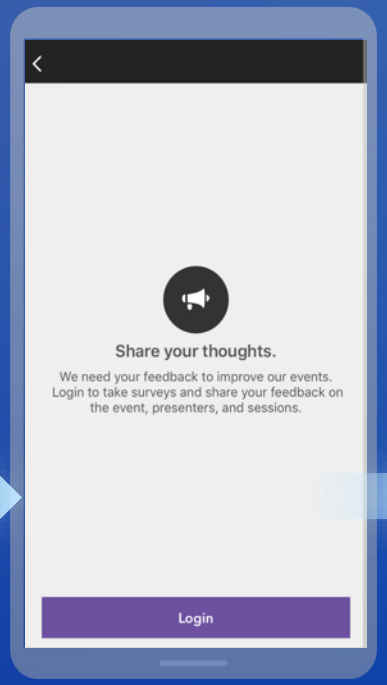
Select the session you attended



Scroll down to "Survey"



Log in to access the survey



Complete the survey and select "Submit"