Updates to the TDS Web User Interface

Summer 2018 Unidata Student Internship
Hailey Johnson
Sean Arms, Christian Ward-Garrison
What I did this summer:

1. HTMLwriter → Thymeleaf templating
2. UI design updates
   a) Customizable CSS
   b) Extensible HTML
3. Jupyter Notebook dataset “viewer”
Thymeleaf Templating

```java
String logoUrl = htmlConfig.prepareUrlStringForHtml(htmlConfig.getInstallLogoUrl());
if (logoUrl != null) {
    sb.append("<img src=").append(logoUrl);
    String logoAlt = htmlConfig.getInstallLogoAlt();
    if (logoAlt != null) sb.append(" alt=").append(logoAlt);
    sb.append(" align='left' valign='top'">\n");
}

sb.append(" Catalog ").append(catname);
sb.append("</h1>");
sb.append("<HR size='1' noshade='noshade'>");

sb.append("<table width='100%' cellspacing='0' cellpadding='5' align='center'>\n");

// Render the column headings
sb.append("<tr>\n");
sb.append("<th align='left'><font size='+1'>")
    sb.append("Dataset")
    sb.append("</font></th>\n");
sb.append("<th align='center'<font size='+1'>")
    sb.append("Size")
    sb.append("</font></th>\n");
sb.append("<th align='right'><font size='+1'>")
    sb.append("Last Modified")
    sb.append("</font></th>\n");
sb.append("<th align='left'><font size='+1'>")
    sb.append("</th>\n");

// Recursively render the datasets
doDatasets(cat, cat.getDatasetsLocal(), sb, shade: false, level: 0, isLocalCatalog);

// Render the page footer
sb.append("</table>\n");
sb.append("</HR size='1' noshade='noshade'>")
appendSimpleFooter(sb);
sb.append("</body>\n")
sb.append("</html>\n")
```

Intro Applications Progress/Results Going Forward Demo Applications Going Forward
Thymeleaf

Natural templates

- Server-side templating engine
- Templates can be rendered as HTML by browsers
- Integrates with Spring

- Variable/Selection expressions
- Basic logical operators, conditionals, loops, mathematical operations, etc.
- Fragment expressions
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head></head>
<body>
<div th:fragment="access" class="tab-content access" id="access">
<h3>Access:</h3>
<table class="property-table">
<tr><th>Service</th><th>Type</th><th>Description</th></tr>
<tr th:each="access : ${dataset.getAccess()}">
<td><a th:href="${access.get('href')}">
  <b th:text="${access.get('serviceTypeName')}"></b>
</a></td>
<td th:text="${access.get('accessType')} != null">${access.get('accessType')}</td>
<td th:text="${access.get('serviceDesc')} == null ? '' : access.get('serviceDesc')"></td>
</tr>
</table>
</div>
</body>
</html>
Which pages use Thymeleaf?

1. Catalog pages
2. Dataset pages
3. NCSS (Grid & Point) pages

Why?

1. Simplicity
2. Efficiency
3. Consistency
Thymeleaf templating

Thymeleaf'd views:

```java
public CatalogItemContext(Dataset ds, int level)
{
    // Get display name
    this.displayName = ds.getName();

    // Get data size
    double size = ds.getDataSize();
    if (size > 0 && !Double.isNaN(size))
        this.dataSize = Format.formatByteSize(size);

    // Get last modified time
    Date lastModDateType = ds.getLastModifiedDate();
    if (lastModDateType != null)
        this.lastModified = lastModDateType.toDateTimeString();

    // Store nesting level
    this.level = level;
}

protected void populateItemContext(Dataset ds, List<CatalogItemContext> catalogItems,
                                          boolean isLocalCatalog, int level) {
    CatalogItemContext context = new CatalogItemContext(ds, level);
    context.setHref(getCatalogItemHref(ds, isLocalCatalog));
    context.setIconSrc(getFolderIconSrc(ds));
    catalogItems.add(context);
    // Recursively add subdirectories
    if (ds instanceof CatalogRef) {
        addCatalogItems(ds, catalogItems, isLocalCatalog, level + 1);
    }
}
```
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>Last Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Realtime data from IDD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Forecast Model Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Forecast Products and Analyses</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Observation Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Radar Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Satellite Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other Unidata Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unidata case studies</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Datasets</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Datasets</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Point Datasets</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CF DSG Example Datasets</td>
<td></td>
<td></td>
</tr>
<tr>
<td>For GRIB Indexing Purposes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Welcome to THREDDS Data Server top-level TDS Catalog.
Hosted by Unidata.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>Last Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Realtime data from IDD</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Unidata case Studies</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Test Datasets</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Test Grid Datasets</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Test Point Datasets</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Test Restricted Datasets</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>NARR Test</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>For GRID Indexing Purposes</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>GSD NARR Datasets</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>
Catalog https://thredds-test.unidata.ucar.edu/thredds/catalog/grib/NCEP/GFS/Global_0p5deg_ana/GFS_Global_0p5deg_ana_20180620_0000.grib2/catalog.html

Dataset: GFS_Global_0p5deg_ana_20180620_0000.grib2

- Data format: GRIB2
- Data size: 66.65 Mbytes
- Feature type: GRID
- Naming Authority: adu.ucar.edu
- ID: grb/NCEP/GFS/Global_0p5deg_ana/GFS_Global_0p5deg_ana_20180620_0000.grib2

Documentation:

- **summary**: Single reference time Grib Collection
- **Reference Time**: 2018-06-20T00:00:00Z
- **NCEP Model documentation**
- **rights**: Freely available
- **processing_level**: Transmitted through Unidata Internet Data Distribution.
- **processing_level**: Read by CDM Grb Collection.
- **NCEP/AMSG Model Analyses and Forecasts page**
- **Unidata IDD Model Data page**
- **summary**: NCEP Global Forecast System Model, previously called AVN/MRF (Medium Range Forecast)
- **COMET Model (Meteorology Education and Training) documentation**
- **NCEP Model Notes**
- **summary**: NCEP GFS Model. AWIPS 230 (G) Grid. Global Lat/Lon grid. Model runs at 0, 6, 12, and 18Z. Horizontal: 361 by 726 points, resolution 0.5 degree, Lat/Lon projection. Vertically: 1009 to 10 hPa mandatory pressure levels (10 levels), surface, height above ground, pressure levels.
- **summary**: Analysis grids only.

Access:

1. OPENDAP: thredds/dodsC/grib/NCEP/GFS/Global_0p5deg_ana/GFS_Global_0p5deg_ana_20180620_0000.grib2.html
2. CdmRemote: thredds/dodsC/grib/NCEP/GFS/Global_0p5deg_ana/GFS_Global_0p5deg_ana_20180620_0000.grib2
3. CdmFeature: thredds/dodsC/grib/NCEP/GFS/Global_0p5deg_ana/GFS_Global_0p5deg_ana_20180620_0000.grib2
4. DAPA: thredds/dodsC/grib/NCEP/GFS/Global_0p5deg_ana/GFS_Global_0p5deg_ana_20180620_0000.grib2
dm.xml
5. HTTPS: thredds/fileServer/grib/NCEP/GFS/Global_0p5deg_ana/GFS_Global_0p5deg_ana_20180620_0000.grib2
7. WMS: thredds/wms/grib/NCEP/GFS/Global_0p5deg_ana/GFS_Global_0p5deg_ana_20180620_0000.grib2
8. WCS: thredds/wcs/grib/NCEP/GFS/Global_0p5deg_ana/GFS_Global_0p5deg_ana_20180620_0000.grib2
9. ISO: thredds/iso/grb/NCEP/GFS/Global_0p5deg_ana/GFS_Global_0p5deg_ana_20180620_0000.grib2
10. NCML: thredds/ncml/grb/NCEP/GFS/Global_0p5deg_ana/GFS_Global_0p5deg_ana_20180620_0000.grib2
11. UDDC: thredds/uddc/grb/NCEP/GFS/Global_0p5deg_ana/GFS_Global_0p5deg_ana_20180620_0000.grib2

Dates:

- 2018-06-20T03:24:00Z (modified)

Creators:

- DOC/NOAA/NCO/NCEP
  - email: http://www.ncep.noaa.gov/mail_liaison.shtml
  - http://www.noaa.noaa.gov/

---

**Notice:** This document is a snapshot of the metadata associated with the dataset `GFS_Global_0p5deg_ana_20180620_0000.grib2`. The metadata provides information about the data, including its format, size, and various access methods. The dataset is a gridded forecast model produced by the National Center for Environmental Prediction (NCEP) and is available through various data access protocols such as OPENDAP, CDM Remote, and HTTPS. The dataset covers a single reference time of 2018-06-20T00:00:00Z and includes global latitude/longitude grids with horizontal resolution of 0.5 degrees and a range of mandatory pressure levels from 1009 to 10 hPa. Users can access the data through different portals and protocols, and the creators are the National Oceanic and Atmospheric Administration's National Centers for Environmental Prediction.
Dataset: DEM_Mtnzs_20180227.nc
Catalog: http://localhost:8081/thredds/catalog/rtkMtnzs/catalog.html

dataFormat: NetCDF
featureType: Grid
dataSize: 12756088
id: rtkMtnzs/DEM_Mtnzs_20180227.nc

Access:

<table>
<thead>
<tr>
<th>Service</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenDAP</td>
<td>Data Access</td>
<td>Access dataset through OPeNDAP using the DAP2 protocol.</td>
</tr>
<tr>
<td>DAP4</td>
<td>Data Access</td>
<td>Access dataset through OPeNDAP using the DAP4 protocol.</td>
</tr>
<tr>
<td>HTTPServer</td>
<td>Data Access</td>
<td>HTTP file download.</td>
</tr>
<tr>
<td>NCML</td>
<td>Metadata</td>
<td>Provide NCML representation of a dataset.</td>
</tr>
</tbody>
</table>

Documentation

Variables:

- Vocabulary []
  - PX (eastings) = Longitude coordinates of area polygon.
  - PY (northings) = Latitude coordinates of area polygon.
  - X (eastings) = Longitude of collected data points.
  - Xg (eastings) = Longitude of gridded data.
  - Y (northings) = Latitude of collected data points.
  - Yg (northings) = Latitude of gridded data.
  - Z (meters) = Elevations of collected data points.
  - Zg (meters) = Elevations of gridded data.
UI Updates

- Existing feature: user contributed stylesheet
- Use templates for consistency:
  - Page templates look for user contributed stylesheets which correspond to that page
    - “standard.css” – loaded on all pages
    - “catalog.css” and “dataset.css” – loaded on catalog/dataset views
- Page layout is reliably structured:
  - Elements have classes and ids
UI Updates

- Thymeleaf fragments: portable templates
  - Page templates structured as a rigid skeleton which loads fragments
  - Plugin points: e.g. header, footer, content-section
  - User adds fragments to `content/thredds/templates/tdsTemplateFragments.html`
- Overridable fragments: custom or default
- Optional fragments: custom or none
UI Updates

Customizable CSS

Extensible HTML

**Template resolvers:**
- Default: `SpringResourceTemplateResolver`
- Custom: `TdsExtensibleTemplateResolver`
  - Resolvable patterns: “ext:*”
  - Looks for matches in the content directory

```html
a.<div th:fragment="header-container">
  <div id="header-buffer"></div>
  <div id="header" th:insert="{ext:tdsTemplateFragments :: header} ?: ~(templates/commonFragments :: header-content)">
  </div>
</div>

b.<div class="content" th:if="{ext:tdsTemplateFragments :: catalogCustomContentBottom}"
  th:replace="{ext:tdsTemplateFragments :: catalogCustomContentBottom}"/>
</div>
```
UI Updates

Customizable CSS

Extensible HTML

Example: Contributing multiple fragments

templates/tdsTemplateFragments.html

```html
<div th:fragment="datasetCustomContentBottom">
    <div th:replace="~{ext:additionalFragments/myFragments :: mySectionHeader}"/>
    <div th:replace="~{ext:additionalFragments/myFragments :: mySectionContent}"/>
</div>
```

templates/additionalFragments/myFragments.html

```html
<div th:fragment="mySectionHeader" class="section-header">My Section Name</div>
<div th:fragment="mySectionContent" class="section-content">Your contributed content here.</div>
```
Jupyter Notebook Service

Siphon THREDDS Jupyter Notebook Viewer
Dataset: GOES16_CONUS_20180719_000228_0.47_1km_30.1N_87.1W.nc4

Dependencies:
- Siphon: pip install siphon
- matplotlib: pip install matplotlib
- ipywidgets: pip install ipywidgets or conda install -c conda-forge ipywidgets
- Using Jupyter Notebook: jupyter nbextension enable --py widgets
- Using JupyterLab: requires nodejs

Access a dataset
With the TDS catalog url, we can use Siphon to get the dataset named datasetName.
Jupyter Notebook service

Purpose:

- Return ipynb file pre-populated with catalog URL & Dataset name
- Demo access to datasets via Siphon
- Simple data visualization

```
In [2]:
catUrl = "http://localhost:8081/threads/catalog/rtkmznzos/catalog.xml"

datasetName = "DEM_Mtnzs_20171019.nc"
```

### Access a dataset

With the TDS catalog url, we can use Siphon to get the dataset named `datasetName`.

```
In [4]:
catalog = TDSCatalog(catUrl)

In [5]:
ds = catalog.datasets[datasetName]
ds.name
```

Out[5]: `DEM_Mtnzs_20171019.nc`
Jupyter Notebook service

How it works:

1. On TDS startup:
   a) TDS creates a cache to store Notebooks and their mappings to datasets
   b) Parses all ipynb files in the “Notebooks” directory as NotebookMetadata objects; saved permanently in the cache
   c) Registers “JupyterNotebookViewer” as a data viewer

2. On Dataset page load
   a) TDS accesses or creates a mapping to the appropriate Notebook

3. On Notebook service request
   a) NotebookController reads the mapped ipynb file and inserts the Catalog URL and Dataset Name where appropriate
   b) Returns the edited Notebook file
Mapping Notebooks to Datasets

Edit Notebook Metadata

Manually edit the JSON below to manipulate the metadata for this notebook. We recommend putting custom metadata attributes in an appropriately named substructure, so they don't conflict with those of others.

```json
{
  "kernelspec": {
    "name": "python3",
    "display_name": "Python 3",
    "language": "python"
  },
  "language_info": {
    "name": "python",
    "version": "3.6.5",
    "mimetype": "text/x-python",
    "codemirror_mode": {
      "name": "ipython",
      "version": 3
    },
    "pygments_lexer": "ipython3",
    "nbconvert_exporter": "python",
    "file_extension": ".py"
  },
  "viewer_info": {
    "accept_catalogs": [
      "http://localhost:8081/thredds/catalog/rtkMatanzas/catalog.html"
    ],
    "order": 2
  }
}
```
Jupyter Notebook service

Mapping notebooks to datasets:

```java
private class NotebookMetadata {
    public String filename;
    public boolean accept_all;
    public List<String> accept_datasetIDs;
    public List<String> accept_catalogs;
    public List<String> accept_dataset_types;
    public int order;
}

@Component
public class JupyterNotebookServiceCache {
    static private final Logger logger = LoggerFactory.getLogger(JupyterNotebookServiceCache.class);

    @Autowired
    TdsContext tdsContext;

    private List<NotebookMetadata> allNotebooks;

    private Cache<String, NotebookMetadata> notebookMappingCache;

    public boolean isValidForDataset(Dataset ds) {
        // Implementation
    }

    public int compareNotebookForDataset(Dataset ds, NotebookMetadata md) {
        // Implementation
    }
```
Contributing your own Notebook:

1. Create your notebook (ipynb)
2. Update the “viewer_info” metadata
3. Place notebook in `content/thredds/notebooks`
TDS “in the wild”

Demo #1
Demo #2

TDS not “in the wild”
Ok, that’s kinda cool, but so what?
Goals/Benefits

- Administrative users:
  - Curate data more effectively
  - Target users and use cases

- End users:
  - Lower barrier-to-entry for data use
    - Web navigation
    - Access and usage examples/instructions
  - Wider range of users
Use cases

- Large repositories:
  - Data discoverability, visualization, access
- Education:
  - Interactive instructional notebooks
- Publication:
  - Small data repositories as supplementary material
  - Verify methods and results
### Future work

**Baseline work:**
- Document contributing Notebooks
- Add tests
  - Notebook service
  - UI tests
Future work

- Notes from building a demo:
  - Free-form metadata type for datasets and catalogs
  - Top content-section for dataset page
  - Pattern-matching for notebook registration
Future work

• More features:
  • Multiple Notebook viewers per dataset (end user’s choice)
  • Distribute common templates for features
Summer 2018: a visual summary

Thank you, everyone!