Kepler Exoplanet Viewer

Team: Stargazers
Ryan Nett, Wilson Leong, and Andrew Pirondini

Overview

Takes data from Kepler (a NASA telescope) archives.

Imports that data to the database.

Displays the data for users.

There is a lot of data (almost 300 solar systems), need useful controls.

- Using a database allows for easy filtering and storage.
- Could also expand with constellation or visual location data.

Project Goals

- Viewing information of all planets within a solar system.
- Finding out which solar systems have habitable planets.
- Filtering of solar systems to find interesting ones.

Demo

Internals

UI

- Created using JavaFX inside of SceneBuilder
- 3 Panes in one large view
    - Solar systems table and filter
    - Detailed information
    - Visualization of Solar System
- Uses event handlers to add functionality to the view

Database

Two tables and a view.

- Planets: planet data, keyed by star name and planet letter
- Stars: star data, keyed by star name
- SolarSystems (View): join on star name, adds number of goldilocks planets data.

Database

View to hold number of goldilocks planets per solar system:

```
select * from
planets P join
(
select S1.starName, sum(P1.goldilocks) as golds from
planets P1 join stars S1 using(starName)
group by P1.starName
) G
using(starName)
join stars S using(starName)
```

Backend

Passing solar system information to the front end:

1. Joined stars and planets databases (on star name) to put all planets that belonged to the same star within the solar system together in contiguous tuples (what the query that created the SolarSystems view did).
2. No projection of attributes allowed.
3. Lists of attributes, comparison operators (i.e. ‘=’ or ‘>’), values and boolean operators (i.e. ‘and’) were passed in by the frontend to query from the database and filter the results for its desired data.

Parser

Parses the Kepler data. This is data that NASA made available and people have gone over to find exoplanets.

Converts some types to enums and calculates the goldilocks range.

Inserts in two statements, using VALUES (), (), ()...

Conclusion

What We Learned and Project Challenges

- Filling in missing data (i.e. missing planets) from original data.
- Visible parts of the program could still run despite exceptions being thrown under the hood.
- Scaling the size of the star such that it is not too small for the user to see.
- SSH into a unix server & port forwarding to connect to database. Jsch (Java Secure Channel) library mostly took care of this