Wellness Profiles using Rules:

Note: All times follow a specific date time structure but their specific implementation is not yet realized. Therefore, all times intended to be used for the same "event" or "day" must match (i.e. forecast start time and season start times must be identical to be matched in the rule).

Note: With regards to N3 syntax, "_:" represents all instances of the subject. In actual implementation, the following format is used:

":subject_1 …."
":subject_2…."
Etc.

Season:

Defines the Season corresponding to the start time. Intended for use in a date-time system, but currently uses specific moment in time.

POSL Syntax:

```
season(?StartTime,?Season).
```

N3 Syntax:

```
_:season
        rdf:type        :Season;
        :startTime      ?StartTime;
        :value          ?Season.
```

Usage:

StartTime = The time of day that represents the season.

Season = The season corresponding to the StartTime.

Forecast:

Defines the weather corresponding to the type and start time. Intended for use in a date-time system, but currently uses specific moment in time.

POSL Syntax:

```
forecast(?StartTime,?Type,?Weather).
```

N3 Syntax:

```
_:forecast
        rdf:type        :Forecast;
        :startTime      ?StartTime;
        :aspect         ?Type;
        :value          ?Weather.
```

Usage:

StartTime = The time of day that represents the forecast.

Type = The type of weather forecast. Currently, types are **sky** or **temperature**.

Weather = The value corresponding to the sky or temperature at the given StartTime.
(i.e. sunny corresponds to sky. 27 corresponds to temperature).

Note: Temperature measured in **Celsius**

Meetup:

Defines the possible locations for individuals to “meet up” for their activities. Each location is tied to its map and activity type.

POSL Syntax:

meetup(?MapID,?Activity, ?Ambience,?Location).

N3 Syntax:

```
_:meetup
        rdf:type        :Meetup;
        :mapID          ?MapID;
        :activity       ?Activity;
        :inOut          ?Ambience;
        :location       ?Location.
```

Usage:

MapID = The map that the meetup location is assigned to. In order for profiles to share meetup locations, their MapID’s must be the same.

Activity = The kind of activity being performed.

Ambience = Whether the activity is being performed inside or outside.

Location = The name of the meetup location.

GoodDuration:

Ideally, this would determine whether or not an activity can be carried out based on the event time differences and the duration, via date time functions. But as this is not realized, each event time is stored as fact to inform the knowledge base as to whether or not it is a suitable duration time for the event.

POSL Syntax:

goodDuration(?Duration,?StartTime,?EndTime).

N3 Syntax:

```
_:goodDuration

        rdf:type        :GoodDuration;
```

```
          :duration       ?Duration;
          :startTime      ?StartTime;
          :endTime        ?EndTIme.
```

<u>Usage:</u>

Duration = The duration of the activity.

StartTime = The start time of the event.

EndTime = The end time of the event.

Yesterday:

Again, because the date time implementation is not yet realized, a yesterday value must be manually assigned for each event that will use a rule utilizing the yesterday fact.

<u>POSL Syntax:</u>

```
yesterday(?StartTime,?StartTimeYDay,?EndTime,?EndTimeYDay).
```

<u>N3 Syntax:</u>

```
_:yesterday
          rdf:type        :Yesterday;
          :startTime      ?StartTime;
          :startTimeYDay  ?StartTimeYDay;
          :endTime        ?EndTime;
          :endTimeYDay    ?EndTimeYDay.
```

<u>Usage:</u>

StartTime = The start time.

StartTimeYDay = The previous day’s start time.

EndTime = The end time.

EndTimeYDay = The previous day’s end time.

Calendar:

Each individual profile has a calendar which will have activities (separated by types) penciled in at times of their choice.

<u>POSL Syntax:</u>

```
calendar(?ProfileID,?CalendarID).
```

<u>N3 Syntax:</u>

```
_:calendar
          rdf:type        :Calendar;
          :profileID      ?ProfileID;
          :calendarID       ?CalendarID.
```

Usage:
ProfileID = The profile identification tied to the calendar.
CalendarID = The calendar identification tied to the profile.

Events:

As a user performs events, or plans to perform events, they will be registered in the knowledge base as **Possible, Planned,** (**Performing**), or **Past** events. Possible events are events that do not have a specific activity assigned to them, but the user has indicated that he/she would like to perform an activity between the times registered. Planned refers to an event that has not happened yet, but has a specific activity assigned to it. Past events are events that have already occurred.

POSL Syntax:

```
event(?CalendarID,?Aspect,?Tense,?StartTime,?EndTime).
```

N3 Syntax:

```
_:event

          rdf:type          :Event;
          :calendarID       ?CalendarID;
          :aspect           ?Aspect;
          :tense            ?Tense;
          :startTime        ?StartTime;
          :endTime          ?EndTime.
```

Usage:
CalendarID = The calendar identification tied to the event.
Aspect = The type of event planned. Typically an activity.
Tense = Possible, Planned, or Past. (As described above).
StartTime = The start time of the event.
EndTime = The end time of the event.

Map:

Each individual can have multiple maps to use for their meet up locations (which would be based on where they intended to be for the time they have chosen).

POSL Syntax:

```
map(?ProfileID,?MapID).
```

N3 Syntax:

```
_:map
```

```
    rdf:type          :Map;
    :profileID        ?ProfileID;
    :mapID            ?MapID.
```

Usage:
ProfileID = The profile identification assigned to the map.
MapID = The map identification assigned to the profile.

Fitness:

A user will define their expected fitness levels (on a scale from 1-10) for each day or time, so that an appropriate activity and location can be chosen.

POSL Syntax:

```
fitness(?ProfileID,?StartTime,?ExpectedFitnessLevel).
```

N3 Syntax:

```
_:fitness

    rdf:type                :Fitness;
    :profileID              ?ProfileID;
    :startTime              ?StartTime;
    :expectedFitness        ?ExpectedFitnessLevel.
```

Usage:
ProfileID = The profile identification corresponding to the fitness level.
StartTime = The time of day when the user has the expected fitness level.
ExpectedFitnessLevel = The user’s expected fitness level for that time of day (on a scale from 1-10).

Level:

Each location will have its own defined fitness level set by the user or calculated previously based on the duration of the activity. It will be unique to the individual, activity, and location. Again, duration is stored as date time but is not implemented, so strings must be matched identically.

POSL Syntax:

```
level(?ProfileID,?Activity,? Ambience,?Location,?Duration,?FitnessLevel).
```

N3 Syntax:

```
_:level

    rdf:type          :Level;
    :profileID        ?ProfileID;
```

```
        :activity           ?Activity;
        :inOut              ? Ambience;
        :location           ?Location;
        :duration           ?Duration;
        :fitnessLevel       ?FitnessLevel.
```

Usage:
ProfileID = The profile identification corresponding to the fitness level.
Activity = The type of activity being performed for the fitness level.
Ambience = Whether the activity is indoors or outdoors.
Location = The meet up location where the activity is being performed.
Duration = How long the activity is performed (recorded as date time).
FitnessLevel = The required fitness level required by the user for the corresponding values.

GroupSize:

Each user may define minimum and maximum extremes for the number of participants required in a group to corresponding to a specific activity. When activities are queried, the requested group min and max values are checked against the user's recorded preferences.

POSL Syntax:

```
groupSize(?ProfileID,?Activity,?Ambience,?Min,?Max).
```

N3 Syntax:

```
_:groupSize
        rdf:type            :GroupSize;
        :profileID          ?ProfileID;
        :activity           ?Activity;
        :inOut              ?Ambience;
        :min                ?Min;
        :max                ?Max.
```

Usage:
?ProfileID = The profile identification corresponding to the group size requirements.
?Activity = The activity for which the group size requirements are focused on.
?Ambience = Whether the activity is indoors or outdoors.
?Min = The minimum number of participants.
?Max = The maximum number of participants.

MyActivity:

The objective of the previous wellness rules is to accurately suggest a wellness activity to the user according to their profile preferences. Each activity's requirements can be described by the user, adding or removing as many as they like. Although, each activity must have certain base rules in order for them to follow the wellness profile schema.

POSL Syntax:

activity(?ProfileID,?Activity,?Ambience,?MinRSVP,?MaxRSVP,?StartTime,?EndTime,?Location,?Duration,?FitnessLevel).

N3 Syntax:

```
_: myActivity

        rdf:type          :MyActivity;
        :profileID        ?ProfileID;
        :activity         ?Activity
        :inOut            ? Ambience;
        :minRSVP          ?MinRSVP;
        :maxRSVP          ?MaxRSVP;
        :startTime        ?StartTime;
        :endTime          ?EndTime;
        :location         ? Location;
        :duration         ?Duration;
        :fitnessLevel     ?FitnessLevel.
```

Usage:

ProfileID = The profile identification corresponding to the suggested activity.
Activity = The type of suggested activity determined.
Ambience = Whether the suggested activity is indoors or outdoors.
MinRSVP = Submitted minimum number of participants.
MaxRSVP = Submitted maximum number of participants.
StartTime = The start time of the suggested activity.
EndTime = The end time of the suggested activity.
Location = The meet up location for the suggested activity.
Duration = The duration of the suggested activity.
FitnessLevel = The required fitness level of the suggested activity.

Required Rules to Fit Schema:

```
calendar(p0001,?Calendar),
event(?Calendar,Run:Activity,possible,?StartTime,?EndTime),
participation(1:Integer,6:Integer,?MinRSVP,?MaxRSVP),
 map(p0001,?Map),
 meetup(?Map,run,out,?Place),
 level(p0001,run,out,?Place,?Duration,?Level),
greaterThanOrEqual(?ExpectedFitness,?Level),
goodDuration(?Duration,?StartTime,?EndTime.
```

* Many other rules can be implemented such as seasonal requirements, whether requirements, temperature requirements etc… *

Currently Used Activity Types:

Walk

Run

Swim

Skate

Yoga

Hike

Baseball