CSAIL

Max Van Kleek (emax@csail.mit.edu)
Paul André (pa2@ecs.soton.ac.uk)
Michael Bernstein (msbernst@mit.edu)
Mikko Perttunen (mikko.perttunen@ee.oulu.fi)
David Karger (karger@mit.edu)
Rob Miller (rcm@mit.edu)
mc schraefel (mc@ecs.soton.ac.uk)

personal context-adaptive and reactive automation using heterogeneous data sources on the Web

atomsmasher

atomsmasher is a framework which enables data from multiple web sites to drive robust, context-aware reactive behaviors. In contrast to existing data mashups such as Yahoo Pipes, IBM's DAMIA, and Intel MashMaker, which can produce visualizations and aligned feeds from multiple data sources, atomsmasher enables novice web programmers to program arbitrary web-based actions, that can use new data from the web about the world for making scripts world-adaptive and reactive.

atomsmasher achieves this by building a single, consolidated representation of data aggregated from arbitrary sources on the web, providing Javascript programming language abstractions for specifying behaviors and accessing this representation, and a rule engine for efficiently computing how and when actions should be taken. These abstractions, designed to make it easy for web designers and novice programmers to devise complex adaptive behaviors are illustrated and explained below.

how it works

behavior specification

Users can specify behaviors in Javascript (JS) by constructing special "if" statements that refer to atomsmasher query variables and/or state model variables in their condition clause. These variables operate in different ways.

atomsmasher query variables - represent query sets to the atomsmasher KB and as in object relation models (ORM) use Javascript objects as proxies to represent items. Fields of these proxies correspond to properties of items in the KB; e.g. `friends().name` corresponds to the names of all friends. The query set can be reduced by applying boolean predicates to fields of query variables. For example, `friends().address.city.eq(places("Cambridge"))` would represent a query set of friends filtered to only those that live in Cambridge.

atomsmasher state model - represent persisted aspects of the user and environmental state, such as the user's location, task/activity/ongoing event or music listening state. The state model is driven by special behavior rules called state behaviors specified by the user.

data aggregation

atomsmasher can connect to virtually any web-based data source, including RSS/ATOM feeds, REST-based or XML-RPC based web APIs, well as through any custom connector Javascript code. For each such data source, atomsmasher applies generic input processing (parsing) functions to initially extract the data, followed by custom code to transform the particular source's data format into instances of the atomsmasher ontology. This per-source code, called atom splitters are written as modules that can be shared so that others can benefit from each person's work.

try it!

Users of Firefox 3.0 can try a prototype implementation at
`http://plum.csail.mit.edu/atomsmasher` .
Tell us what you think!

examples

1. Update my facebook status whenever I go somewhere.

```
if (my.location)
  setFBStatus("Max is at " + my.location.name);
```

2. Set my away state and RSS reader mode when in meetings

```
if ( my.activity.type.is(Activities.Meeting) ) {
  // public availability state
  setIMAvailability("Away: At a meeting");
  // filter dashboard to only relevant items
  setDashFilter(function(x) {
     return x.tags.
     intersect(my.activity.tags).size() > 0; });
}
```

3. Remind me when I have a far away appointment in advance

```
var max_speed_mps = .8; // walking in meters per second
var dist = calevent.location.dist(my.location);
var timeneeded = (dist.meters()/max_speed_mps).seconds();
var timeuntil = calevent.startTime.minus(Time.now());
var extratime = (5).minutes();

if ( timeuntil.lt(timeneeded.plus(extratime)) ) {
 showReminder("Excuse me sir, your "+ calevent.type.name +
   + " " + calevent.name + " scheduled for " +
   + " " + calevent.startTime.shortString() +
   + " is at " + calevent.location.name +
   ". I'd head over there soon if I were you.");
}
```

4. When I text myself "goodshows", find concerts featuring artists that I have listened to recently, and friends who are free tonight that have listened to them recently too.

```
if (New.type==Message.Text && New.contents=="goodshows") {
  // retrieve all events
  var cs = events({type:'concert', start:Now.day()});
  var playedmusic = audioscrobbler.recentPlaylist();
  var freefriends =
     friends().filter(function(friend) {
     return friend.cal.
            freebetween(Today.hour(18),Tomorrow);
  });
  var goodshows = cs.filter(function(c) {
     return c.location.nearTo(my.location,miles(2)) &&
     playedmusic.artist.eq(c.artist);
  });
  reply(New, freefriends.musicPlaylist.artist
        .eq(goodshows.artist));
}
```