KP development with SmartSlog

SmartSlog: http://sourceforge.net/projects/smartslog/

Instructors:

Lomov Aleksandr
<lomov@cs.karelia.ru>

Vanag Pavel
<vanag@cs.karelia.ru>

SmartSlog: installation

Export global variables (edit «~/.bashrc» file)

```
export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/home/{USER}/inst/lib/pkgconfig
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/{USER}/inst/lib
```

OR

Edit your “~/.bashrc” file

- Open **~/.bashrc** in any editor
- Add export variables

SmartSlog: installation

Install SmartSlog

- Download SmartSlog (SmartSlog_kappa.tar.gz)
  http://cs.karelia.ru/
- Unpack
  tar -zxf SmartSlog_kappa.tar.gz
- cd SmartSlog_kappa/smartslog_kp_library
- Install

```
./configure --prefix=/home/{USER}/inst
make
make install
```

SmartSlog: example

Example

- Download example (smartslog_example.tar.gz)
  http://cs.karelia.ru/
- Unpack
  tar -zxf smartslog_example.tar.gz
- cd Media
- Edit Makefile in ./phone and ./player folders

  change “INCLUDES” to “-I/home/{USER}/inst/include/smartslog/”
  add “-L/home/{USER}/inst/lib” to “EXPAT_AND_SCEW”
- Compile example

  make

Why SmartSlog?

Advantages

- Oriented to low-performance devices
- Support multilingual ontology library generation
- Developer thinks in terms of ontology entities
- Mechanism of traffic reducing
- Complex queries
- Optimizations
- Ontology manipulations
- Based on KPI_Low

Hello world ontology

SmartSlog

1. **API**

   SmartSlog-Manual.odt

   ...

2. **Basic scenarios**

   - Initialize connection to Smart Space
   - Send individual to Smart Space
   - Receive individual from Smart Space
   - Search individual by pattern
   - Subscription to individual's properties
   - Asynchronous subscription

SmartSlog: Initialize connection

```
init_ss();
register_ontology();

if (ss_join(get_ss_info(), "KP name") ==
-1)
{
    /* Error handle */
}

/*
... KP logic ...
                    */

ss_leave(get_ss_info());
```

SmartSlog: Send individual

```
individual_t * phone = new_individual(CLASS_MOBILEPHONE);

/* insert individual with manual UUID */
init_individual_with_uuid(phone, «phone1»);

/* filtering by pattern */
individual_t *pattern = new_individual(CLASS_MOBILEPHONE);
phone->uuid = strdup(«uuid1»);
set_property_by_name(phone, PROPERTY_CURRENTVOLUME,
«value»);
/* … */
set_property_by_name(pattern, PROPERTY_CURRENTVOLUME, NULL);

if (ss_insert_by_pattern(phone, pattern) < 0)
        /* error */
```

SmartSlog: Receive individual

```
individual_t * phone = new_individual(CLASS_MOBILEPHONE);

/* set uuid of individual you want to receive */
phone->uuid = strdup(«uuid1»);

/* populate to local store */
prop_val_t *property=ss_get_property(phone, CLASS_MOBILEPHONE);
/* ... */
{ /* Unable to query */ }

/* ... Or use ss_get_poperty ... */

free_individual(phone);
```

SmartSlog: Searching individual

```
individual_t *pattern =
new_individual(CLASS_MOBILEPHONE);
list_t * individuals = NULL;

/* set properties for individual you are searching */
set_property_by_name(pattern, ...);
set_property_by_name(pattern, ...);
/* ... */

/* populate to local store */
individuals = ss_get_individuals_by_pattern(pattern);

if (individuals == NULL)
{ /* Unable to query */ }

free_individual(pattern);
```

SmartSlog: Subscription

```
individual_t *phone = new_individual(CLASS_MOBILEPHONE);
property_t * property = (property_t *)
        get_property_type(CLASS_MOBILEPHONE, PROPERTY_ISCALLING-
>name);
subscription_container_t *container = new_subscription_container();

/* Prepare to subscription */
list_t *properties = list_get_new_list();
list_add_data(property, properties);
add_individual_to_subscription_container(conatiner, phone, properties);

/* Subscribe to defined properties */
if (ss_subscribe_container(conatiner, false) != 0)
{ /* Handle error */ }

while (true) {
    const prop_val_t *p_val = get_property(phone, PROPERTY_ISCALLING-
>name);
    /* New value in p_val. Working with new properties */
    wait_subscribe(conatiner); }
```

SmartSlog: Hello World

Task for you

- Use **“Hello World”** ontology
  (already generated, download “helloworld.tar.gz”)
- **First KP** should initialize whole individual and publish some properties
- **Second KP** should subscribe to published properties

<u>Additional:</u>

- Extend Second KP to **asynchronous** subscription