Lab #3:
KP development with SmartSlog

Instructor: Sergey Marchenkov

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 C_KPI

SmartSlog

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: installation

- Set environment variable
  ```
  ~> export LIBRARY_PATH=$HOME/inst/lib64
  ```
- Add it to `~/.bashrc` file
- Download archive SmartSlog_dapi.tar.gz
- Unpack
  ```
  ~> tar -zxf SmartSlog_dapi.tar.gz
  ~> cd SmartSlog_dapi
  ```
- Install
  ```
  ~> mkdir build
  ~> cd build
  ~> cmake -DCMAKE_INSTALL_LIBDIR=$HOME/inst/lib64
  -DINCLUDE_INSTALL_DIR=$HOME/inst/include ..
  ~> make
  ~> make install
  ```

SmartSlog: example

Run example

- Run SIB
- Open Media demo folder

```
~> cd ~/SmartSlog_dapi/demos/high_api/media
```

- Edit SIB IP and port in *media.h* (line ~7)

```
#define KP_SS_ADDRESS "127.0.0.1"
#define KP_SS_PORT <use cutesib port>
```

- Compile example

```
~> cd ~/SmartSlog_dapi/build/
~> make
```

- Run (in different terminals)

```
~> cd demos/high_api/media
~> ./phone
~> ./player
```

SmartSlog: Initialize connection

```
sslog_init();

sslog_node_t *node = sslog_new_node( "KP name", “X”,
                                     "127.0.0.1”, YOUR_PORT);
register_ontology();

if (sslog_node_join(node) != SSLOG_ERROR_NO)  {
        printf(“Get last error: %s\n", sslog_error_get_last_text());
        /* Error handle */
}

/*
… KP logic …
*/

sslog_node_leave(node);
sslog_shutdown();
```

SmartSlog: Insert individual

```
char *uri = “http://example.ru#phone1”;

sslog_individual_t *phone = sslog_new_individual(CLASS_MOBILEPHONE, uri);

sslog_insert_property(phone, PROPERTY_ISCALLING, “calling_value”);

...

if (sslog_node_insert_individual(node, phone) != SSLOG_ERROR_NO) {
        /* Error handle */
}
```

SmartSlog: Get individual

```
sslog_individual_t *phone = sslog_node_get_individual_by_uri (node, uri);

char *p_val = (char *) sslog_node_get_property(node, phone,
                                                PROPERTY_ISCALLING);

if (p_val == NULL) {
        /* Error handle */
}
/* ... */
```

SmartSlog: Subscription

```
char *uri = “http://example.ru#phone1”;

sslog_individual_t *phone = sslog_new_individual(CLASS_MOBILEPHONE, uri);

sslog_subscription_t  *subscription = sslog_new_subscription(node, false);

list_t *properties = list_new();
list_add_data(properties, PROPERTY_ISCALLING);
sslog_sbcr_add_individual(subscription, phone, properties);

 if (sslog_sbcr_subscribe(subscription) != SSLOG_ERROR_NO)  {
        /* Error handle */
}

while (true) {
        char *p_val = (char *) sslog_get_property(phone, PROPERTY_ISCALLING);
        /* New value in p_val. Working with new properties */
        sslog_sbcr_wait(subscription);
}
```

Task

- Use **“HelloWorld”** ontology
  (already generated, download archive helloworld.tar.gz)

- **First KP** should initialize whole individual and publish some properties
- **Second KP** should subscribe to published properties

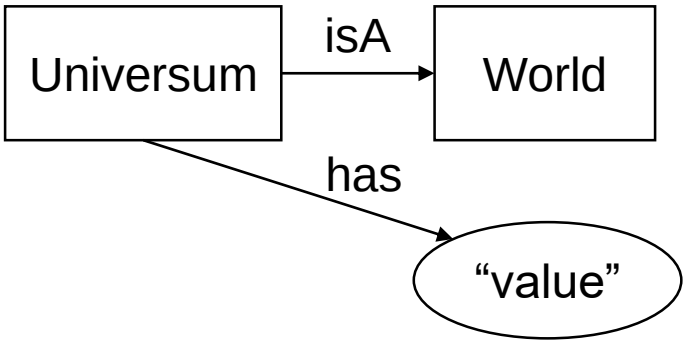
Optional:

- Extend Second KP to **asynchronous** subscription (use sslog_sbcr_set_changed_handler())

HelloWorld ontology

CLASS_UNIVERSUM;
CLASS_WORLD;

PROPERTY_ISA;
PROPERTY_HAS;



Use CLASS_UNIVERSUM and PROPERTY_HAS (data-property).

Publisher:
1) creates an individual from universum class with some uuid;
2) manipulates with the 'has' property;
3) Inserts individual to the smart space.

Consumer:
1) creates the individual from universum class with the same uuid;
2) subscribes to the 'has' property
3) receives new data by subscription.