Service creation in Genius Web Portal

Nicola Venuti
NICE srl

Geneva, 10-12.10.2006, 2nd Tutorial for Health e-Child Project

Agenda

- How preparing a new Plugin
- How Create your own services
- The Authentication
- The Authorization Framework

XML terminology

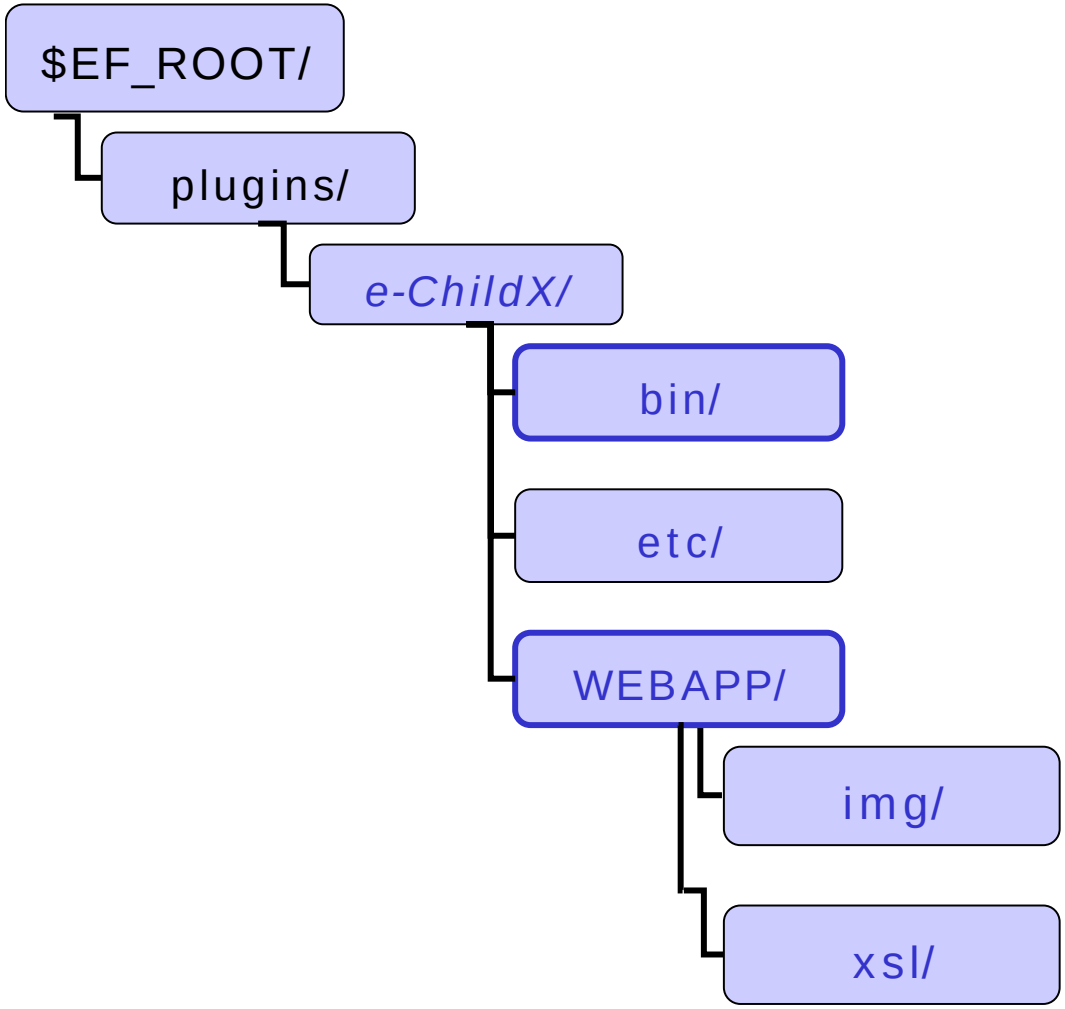
- **XML** - e**X**tensible **M**arkup **L**anguage
  a data format for structured document interchange
- **XSL** - e**X**tensible **S**tylesheet **L**anguage
  XSL describes how the XML document should be displayed
  - **XSLT** - a language for transforming XML documents
  - **XPATH** - a language for navigating in XML documents
  - **XSL-FO** - a language for formatting XML documents
- **Tag** or **Element** - an item enclosed in <>, which is part of an XML document
  - e.g. <**action**>, <**table**>
- **Attribute** - an option of a tag, which has a name and a value enclosed by single or double quotes
  - e.g. <service id="**myservice**">
- **Name space** - an optional naming convention that groups tags related to a common context
  - e.g. <**ef**:action>, <**ef**:service>
- **SDF** – Services Definition File. Xml File which contains the definition of services.

Preparing a new Plugin

- Create Your Directory under plugin
- Edit the SDF (XML code)
  - Start from simple services and evolve them
- (optional)Edit the layout (XSL code)
  - Official web sites can be the perfect source of images and styles
- **Do not edit system . xml / xsl files**
  - Upgrade and porting would be much more difficult

Creating a SDF– Plugin Structure

- directory structure



Services Creations Steps

The steps you have to follow in order to make your new service accessible to the users, are:

- Choose the Service Definition File that will include the new service;
- Write the <ef:service> tag and give it an unique identifier. This identifier must be located in the id attribute of the tag;
- Add a <ef:name> tag which will be used as the readable text for the hyper-link in the page containing the services. Clicking on the hyper-link will execute the service;
- Add an <ef:action> tag;
- As a text node of the <ef:action> tag, add the command that must be executed. for example, ${EF_ROOT}/plugins/ef/bin/ef.test. Note the use of the ${EF_ROOT} syntax: this makes your Genius services independent from the Genius installation directory;
- As a child node of the <ef:action> tag, add the <ef:result> tag. The attribute type of <ef:result> suggests to the Genius Server what kind of output it should expect. (XML, HTML or Simple Text);
- Finally, close the <ef:action> and the <ef:service> tags.

<ef:agent> in Detail..

- Sub tags of <ef:agent>
  - [optional] **<ef:name>** – title of the web page
  - [optional] **<ef:info>** – welcome message of the opening page
  - [optional] **<ef:location>** – location of Remote Agent(s)
    - [required] *host* – hostname/IP address of EF Agent host
    - [required] *port* – TCP port used by EF Agent
  - [optional] **<ef:include>** – external XML libraries of functions
    - [required] *xml* – path to an SDF which contains XML libraries
  - [optional] **<ef:spooler>** – definition of default data
    - [required] *server* – server-side absolute path of spoolers
    - [optional] *agent* – agent-side absolute path of spoolers
    - [optional] *ttl* – defines how long the spoolers will be accessible format: [DD]d[HH]h[MM]m[SS]s, 0(remove immediately), -1(not create)
  - [required] **<ef:folder** *id="root"***>** – first folder of the service tree.

The commands to creates a new plugin

- mkdir /opt/genius/enginframe/plugins/e-Child*X*
- cd /opt/genius/enginframe/plugins/e-Child*X*
- mkdir bin
- mkdir etc
- mkdir WEBAPP
- cd WEBAPP/
- cp ../../genius/WEBAPP/com.enginframe.genius.xml .
- cp ../../genius/WEBAPP/com.enginframe.genius.xsl .
- mkdir images
- cp ../../genius/WEBAPP/layout.css .
- cp ../../genius/WEBAPP/layout.xsl .
- touch e-Child*X*.xml
- ln -s e-Child*X*.xml index.xml

https://yourserver/genius/e-ChildX/

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet href="layout.xsl" type="text/xsl"?>

<ef:agent id="e-ChildX" authority="os"
        xmlns:ef="http://www.enginframe.com/2000/EnginFrame">

<ef:location host="127.0.0.1" port="9999"/>
 <ef:include xml="${EF_LIB}/xml/com.enginframe.system.xml"/>
  <ef:name>GENIUS Grid Portal</ef:name>
  <ef:info><H1>Welcome to Health e-Child Project
   Portal...</H1></ef:info>

  <ef:spooler server="${EF_SPOOLER_DIR}" ttl="1d"/>

  <ef:folder id="root">
    <ef:name>Health e-Child Services</ef:name>
     <!-- Services go here -->
   </ef:folder>
</ef:agent>
```

A simple Xml Code for your Service...

```
  <ef:folder id="test">
    <ef:name>Simple Services</ef:name>
     <ef:service id="test-service" authority="os">
      <ef:name>zip services</ef:name>
      <ef:option id="file" label="File to compress"
type="rfb" />



   </ef:service>
  </ef:folder>
```

service.e-Child*X*.sh

- vi /opt/genius/enginframe/plugins/e-Child*X*/bin/service.e-Child*X*.sh

```
#!/bin/sh

gzip $file
```

Add the action in the service..

```
<ef:name>Health e-Child Services</ef:name>
    <ef:folder id="test">
      <ef:name>Simple Services</ef:name>
       <ef:service id="test-service" authority="os">
        <ef:name>zip services</ef:name>
        <ef:option id="file" label="File to compress"
   type="rfb" />

<ef:action id="submit" label="zip a file">
         $EF_ROOT/plugins/e-ChildX/bin/service.e-ChildX.sh
         <ef:result type="text/plain" />
        </ef:action>

     </ef:service>
    </ef:folder>
```

Add same options..

```
<ef:option id="file" label="File to compress" type="rfb" />

 <ef:option id="level" label="Compression level" type="list">
 <ef:option id="9">maximum</ef:option>
 <ef:option id="4">medium</ef:option>
 <ef:option id="1">minimum</ef:option>
</ef:option>

<ef:action id="submit" label="zip a file">
   $EF_ROOT/plugins/e-ChildX/bin/service.e-ChildX.sh
   <ef:result type="text/xml" />
</ef:action>
```

service.eChild.sh

- vi /opt/genius/enginframe/plugins/e-ChildX/bin/service.e-ChildX.sh

```
#!/bin/sh


gzip -$level $file
cp $file.gz ${EF_SPOOLER}
$EF_ROOT/plugins/ef/bin/ef.show.spooler $EF_SPOOLER_URI
```

How the EF embedded scripts work...

```
this EF script (as all the others)
   $EF_ROOT/plugins/ef/bin/ef.show.spooler $EF_SPOOLER_URI

is dinamically expanded in:
<ef:show-spooler
   uri="spooler:///opt/genius/enginframe/spoolers/yourUserna
   me/spoolerDir/" sub=""/>
```

Authentication

- **None**
  - Services executed as the user who runs Tomcat
  - Services launched only by the local agent, no remote execution
- **Standard Unix Authentication (/etc/passwd or NIS)**
  - Authentication achieved via an internal checkpassword program
- **HTTP Basic Authentication**
  - managed externally by Tomcat (or front-end web server)
  - handled internally by http plugin
- **LDAP Authentication**
  - managed externally by a standard LDAP Server
  - handled internally by ldap plugin
- **ActiveDirectory Authentication – Windows Domain**
  - managed externally by a Windows Domain Controller
  - handled internally by activedirectory plugin

Plug a new Authentication Authority

- Genius provides a flexible way to plug a new Authentication Authority into the system
- It allows to create custom mechanisms for authenticating users
- Three steps to create a custom authentication
  - Create a **login** file: **`$EF_ROOT/etc/`*`authority_name`*`.login`**
  - Create the **authentication script**: **`$EF_ROOT/plugins/`*`authority_name`*`/bin/ef.auth`**
  - Use the custom *`authotity_name`* in the *`<ef:agent>`* or *`<ef:service>`* tags:
- Authority OS under **`$EF_ROOT/plugins/os`** is a good example

Plug a new Authentication Authority
Login File

- Login file defines the parameters needed for authentication
- It must reside in the **`$EF_ROOT/etc`** directory
- The name must coincide with the new authority name + the extension **`.login: authority_name.login`**
  - E.g. `ldap.login`, `myauthority.login`
- Login file is an XML file with the following structure

```
<ef:login title="login_form_title"
    xmlns:ef="http://www.enginframe.com/2000/EnginFrame">
   <ef:signature label="Username" type="text" id="_username"/>
   <ef:signature   label="login_field_label"
                   type="{text|password}"
                   id="authentication_parameter_name"/>
    …
</ef:login>
```

Plug a new Authentication Authority
Authentication Script

- Authentication script is the actual implementation of the custom authentication procedure
- Script name must be: **`ef.auth`**
- The script must reside in the following directory: **`$EF_ROOT/plugins/`*`authority_name`*`/bin`**
- It receives in the standard input the authentication parameter values separated by **'\0'** in the same order as defined in the login file.
  - E.g. "`demoUser\0demoPassword\0`"
- After cehcking the credentials the script must produce one the following XML results in the standard output:

```
<?xml version="1.0"?>
<ef:auth xmlns:ef="http://www.enginframe.com/2000/EnginFrame">
    <ef:result>
        <ef:grant/>          Success!
    </ef:result>
</ef:auth>
```

```
<?xml version="1.0"?>
<ef:auth xmlns:ef="http://www.enginframe.com/2000/EnginFrame">
    <ef:result>
        <ef:deny/>           Failed!
    </ef:result>
</ef:auth>
```

Plug a new Authentication Authority
Authentication Script

- Example

```
#!/bin/sh

# Get credentials from <STDIN>
_credentials="`/usr/bin/tr '\0' '\240'`"
_username="`echo \"${_credentials}\" | awk -F '\240' '{print $1}'`"
_password="`echo \"${_credentials}\" | awk -F '\240' '{print $2}'`"

# Check if credentials are correct
if [ "${_username}" = “demo” –a “${_password}” = “ef4test” ] ; then
   cat <<EOF
<?xml version="1.0"?>
<ef:auth xmlns:ef="http://www.enginframe.com/2000/EnginFrame">
     <ef:result> <ef:grant/> </ef:result>
</ef:auth>
EOF
  exit 0
fi
# Wrong authentication
cat <<EOF
<?xml version="1.0"?>
<ef:auth xmlns:ef="http://www.enginframe.com/2000/EnginFrame">
     <ef:result> <ef:deny/> </ef:result>
</ef:auth>
EOF
exit 1
```

Plug a new Authentication Authority
Use New Authentication

- Write the custom authentication in the **`<ef:agent>`** or **`<ef:service>`** tag.
  - As the value of the attribute **`authority`**
- Example

```
<?xml version="1.0"?>
<ef:agent id="tutorial" authority="authority_name"
    xmlns:ef="http://www.enginframe.com/2000/EnginFrame">
  ...
</ef:agent>
```

- It is possible to set the custom authority as the default one, changing **`EF_DEFAULT_AUTHORITY`** inside the **`$EF_ROOT/conf/server.conf`** file

The Authorization Framework

- Authorization System is aimed to authorize ***user*** accesses to ***resources***
  - allowing or denying ***operations***
  - according to a set of predefined ***policies***.
- **Actor** concept abstracts users. An actor can represent
  - single user
  - group of users
- EnginFrame/Genius **resources** are:
  - **folders**
  - **services**
  - service **options**
  - service **action**
  - service **output**
- Authorization **policies** are defined by ***Access Control Lists***

Authorization Configuration

- The process of setting up the Authorization System include the following steps:
  - Definition of *actors*
  - Definition of *access control lists*
  - Binding access control lists to Genius resources
- Definition of *actors* and *access control lists* occur in the Authorization configuration file:
  - **$EF_ROOT/conf/authorization.xconf**
- *ACLs binding* to resources occurs **directly into SDFs**
- The authorization configuration file is an **XML file** read **dynamically** upon changes.

$EF_ROOT/conf/authorization.xconf

```
<ef:authorization xmlns:ef="http://www.enginframe.com/2000/EnginFrame">
 <ef:acl-actor-list>
  <ef:acl-actor id="efadmins" type="efgroup">
   <ef:info>EnginFrame Administrators</ef:info>
   <ef:acl-member type="efuser">${EF_ADMIN}</ef:acl-member>
   <ef:acl-member type="efuser">falzone</ef:acl-member>
  </ef:acl-actor>
 </ef:acl-actor-list>
 <ef:acl-list>
  <ef:acl id="admin-only">
   <ef:info>Privileged Execution for Admins</ef:info>
   <ef:acl-priority>deny</ef:acl-priority>
   <ef:acl-allow>
    <ef:actor id="efadmins">
     <ef:read/>
     <ef:write/>
     <ef:execute/>
.....
```

ACL in SDF

```
<ef:apply-acl select="admin-only" selectorType="simple">
  <ef:folder id="Only_For_Admin">
    <ef:name>Only For Admin</ef:name>
    <ef:service id="test-ACL" authority="os">
     <ef:name>Test ACL</ef:name>
     <ef:info><H1>All services inside are visible only
   at the Administrators</H1></ef:info>
    </ef:service>
  </ef:folder>
</ef:apply-acl>
```