AAL-2009-2-116

Deliverable D4.2
Applications Over The ELDER-SPACES Platform

Deliverable Type:	**PU***
Nature of the Deliverable:	**P***
Date:	**09/09/2013**
Distribution:	**WP4**
Code:	**<ELDER-SPACES_ORIGO_WP4_D4.2>**
Editor:	**ORIGO**
Contributors:	**BYTE, CYBION, e-TRikala, ORIGO, Semmelweis, SLG**

***Deliverable Type:** *PU= Public, RE= Restricted to a group specified by the Consortium, PP= Restricted to other program participants (including the Commission services), CO= Confidential, only for members of the Consortium (including the Commission services)*

** ***Nature of the Deliverable**: *P= Prototype, R= Report, S= Specification, T= Tool, O= Other*

Abstract: A report on the delivered prototype of the Elder-Spaces web portal.

The ELDER-SPACES Consortium consists of:

BYTE	Project Coordinator	Greece
ORIGO	Partner	Hungary
FTB	Partner	Germany
e-Trikala	Partner	Greece
SEMMELWEIS	Partner	Hungary
SLG	Partner	Greece
CYBION	Partner	Italy

DOCUMENT HISTORY

Version	Date	Description	Provided by
0.1	09/07/2013	Initial version	ORIGO
1.0	05/08/2013	First version	ORIGO
2.0	09/09/2013	Modified version	ORIGO

Table of Contents

List of Tables

No table of figures entries found.

List of Figures

No table of figures entries found.

Glossary

Executive Summary

Deliverable D4.2 accompanies the prototype implementation of the Elder-Spaces Web Applications.

1. Introduction

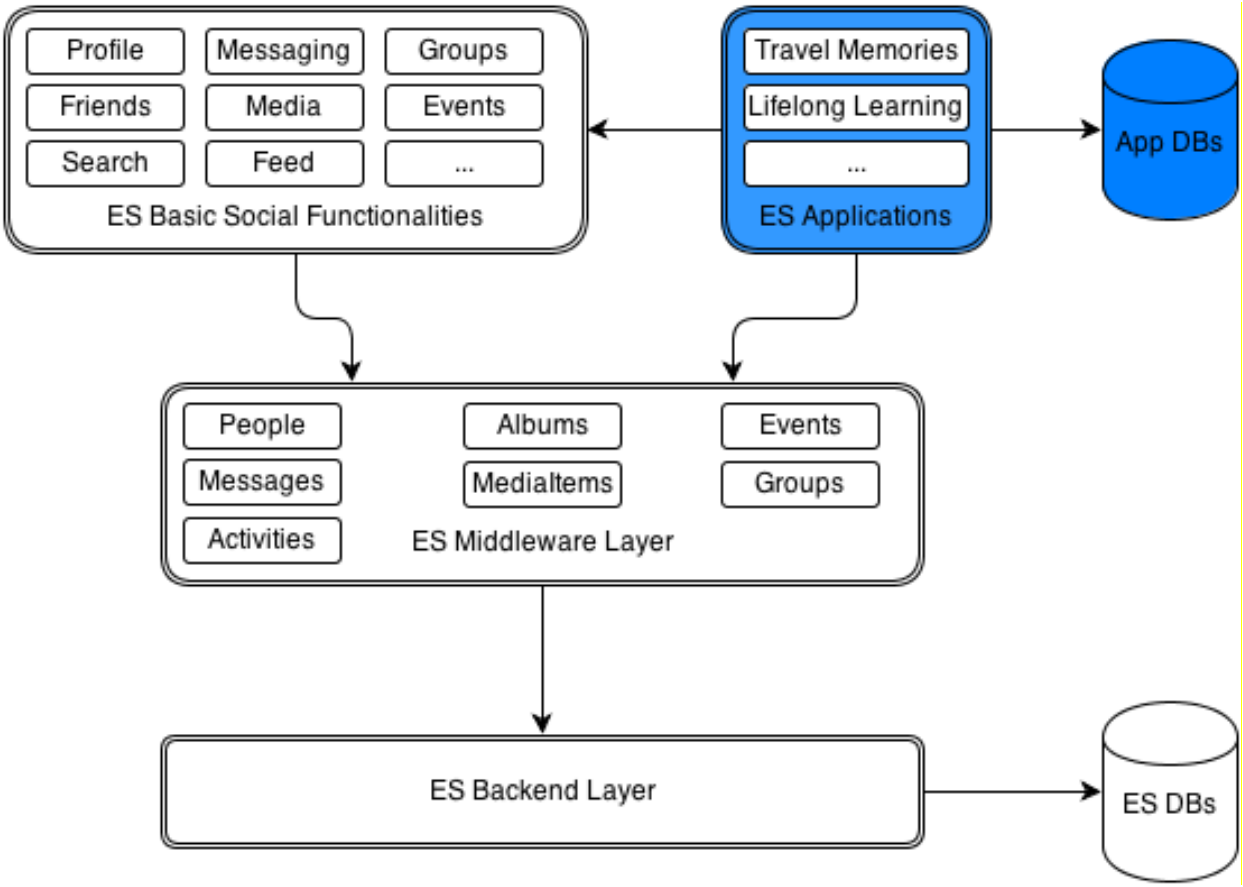
1.1 Overview

The applications of Elder-Spaces platform is like any basic function of the site, but still they are different. Applications are mixtures of the Elder-Spaces basic social functionalities expand with new features in order to provide varied possibilities of using the Elder-Spaces platform for the users.

1.2 Relation with other tasks and work packages

This deliverable is about the development of the Elder-Spaces applications which are specificated in WP2: Travel Memories and Lifelong Learning. These applications are using the Elder-Spaces middleware layer that we deployed in WP3. The integrational and functional testing of these applications is part of WP5.

2. Architectural Characteristics of Elder-Spaces Applications



Elder-Spaces Applications extend and mix the basic social functionalities of the platform.

Applications can use any of the Middleware APIs or the implemented social functions to produce new features. If it's necessary external databases can be attached to the Applications.

3. Additional Entities for Travel Memories

3.1 The TravelMemory Object [Travel Memories]

3.1.1 Description

The entity representing the data of a user's Travel Memory.

3.1.2 Fields

Field	Type	Comment
memoryId	Integer	Automatically generated ID of the Memory
createdTime	Time (e.g.: 1374077238356)	The time when the Memory object was created
countryId	String (foreign key)	The name of the country. It's a foreign key of the Country Object
userId	String	The Elder-Spaces userId of the Memory owner person
categoryId	Integer	0 = 'I was here' 1 = 'I lived here' 2 = 'I want to go here' 3 = 'I live here now'
fromDate	String (e.g. 1975-11-12)	The beginning date of the travel Memory
toDate	String (e.g. 1975-11-12)	The end date of the travel Memory
Description	String	The description of the Travel Memory (max. 1000 characters)
albumId	String	The Elder-Spaces albumId of the album connecting to the Travel Memory

3.1.3 JSON representation

{
"userId":"13913348:elderspaces.iwiw.hu",

"createdTime":"1374527684746",
"categoryId":0,
"memoryId":30,
"fromDate":"2008-2-4",
"toDate":"2008-2-7",
"description":"Lorem ipsum som dolor sit amet",
"countryId":"Mali",
"albumId":”101”
}

3.2 The Country Object [Travel Memories]

3.2.1 Description

The entity representing the data of a country in the Travel Memories Application.

3.2.2 Fields

Field	Type	Comment
countryId	String	The name of the country
latitude	Number	The latitude parameter of the center of the country
longitude	Number	The longitude parameter of the center of the country

3.2.3 JSON representation

{
"countryId":"Hungary",
"createdTime":19.425565367893,
"categoryId":47.167743839592
}

3.3 The Memory Comment Object [Travel Memories]

3.3.1 Description

The entity representing the data of a Travel Memory comment

3.3.2 Fields

Field	Type	Comment

id	Integer	Automatically generated ID of the Memory comment
createdTime	Time (e.g.: 1374077238356)	The time when the comment was created
memoryId	Integer	The ID of the connecting Travel Memory
comment	String	The text of the comment (max 1000 characters)
userId	String	The Elder-Spaces userId of the commenting person

3.3.3 JSON representation

{
"id":13,
"memoryId":30,
"createdTime":19.425565367893,
"comment":"Lorem ipsum dolor sit amet",
"userId":"13913348:elderspaces.iwiw.hu "
}

3.4 The Lesson Object [Lifelong Learning]

3.4.1 Description

The entity representing the data of a Lifelong Learning Lesson

3.4.2 Fields

Field	Type	Comment
lessonId	Integer	Automatically generated ID of the Lesson
createdTime	Time (e.g.: 1374077238356)	The time when the Lesson was created
categoryId	Integer (foreign key)	The category of the video. It's a foreign key of the Category Object
embedCode	String	The embed code of the video
title	String	The title of the video
description	String	The short description of the video
views	Integer	The number of the video's

		views on Elder-Spaces platform
languages	String	List of ISO 3166-1 alpha-2 languages which the given video has a sound track or subtitle.

3.4.3 JSON representation

{
"lessonId":139,
"createdTime":"1374527684746",
"categoryId":2,
"hash":"dQw4w9WgXcQ",
"provider":"youtube",
"title":"Rick Astley - Never Gonna Give You Up",
"description":"Lorem ipsum som dolor sit amet",
"views":66570682,
"languages":["hu", "en", "gr"]
}

3.5 The Lesson Category Object [Lifelong Learning]

3.5.1 Description

The entity representing the data of a Lifelong Learning Lesson Category

3.5.2 Fields

Field	Type	Comment
id	String	The individual name of the Lesson Category
description	String	The short description of the Lesson Category

3.5.3 JSON representation

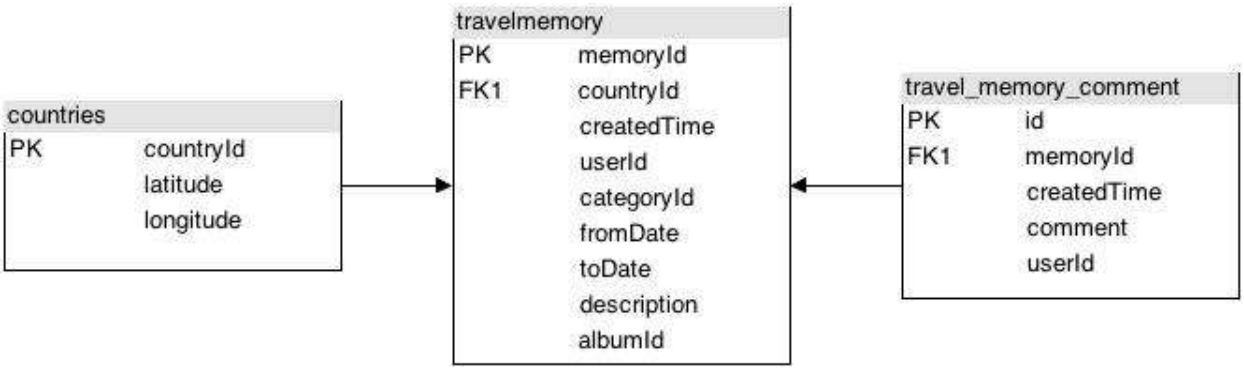
{
"categoryId":"science_education",
"description":"Lorem ipsum dolor sit amet"
}

4. Application Implementation on the Elder-Spaces Platform

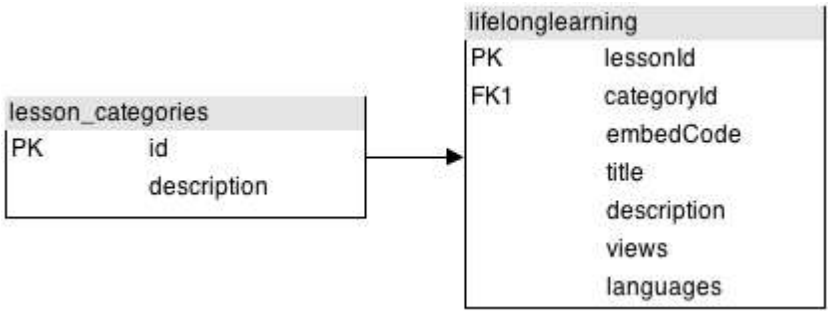
4.1 Application Databases

There is an external MySQL database conneting to each Elder-Spaces application.

4.1.1 Travel Memories Database



4.1.2 Lifelong Learning Database



4.2 Implementation on Web Platform

The Elder-Spaces web applications are integrated into the base Smyfony code of the web platform. The codes of the applications are using the same structure as the basic functionalities and connecting to several social functions of the platform.

This section provides a guideline of creating an Elder-Spaces web application through the examples of the implemented Travel Memories application.

4.2.1 Set The Application Databases

4.2.1.1 Set the parameters

/app/parameters.yml

We can set the parameters of our application databases in parameters.yml. We defined tm_db_* as the database of Travel Memories and ll_db_* as the database of Lifelong Learning:

```
tm_db_driver:   pdo_mysql
tm_db_host:     db1.elderspaces.iwiw
tm_db_port:     ~
tm_db_name:     travel
tm_db_user:     travel
tm_db_password: fow2aiHe

ll_db_driver:   pdo_mysql
ll_db_host:     db1.elderspaces.iwiw
ll_db_port:     ~
ll_db_name:     lifelong
ll_db_user:     lifelong
ll_db_password: aipiR5bu
```

4.2.1.2 The config of the databases

/app/config.yml

In Symfony we can use the built-in Doctrine service for database functions. We need to define the connections for each databases and the entity managers for each applications. The values of the following variables come from parameters.yml that we set before.

```
# Doctrine Configuration
doctrine:
    dbal:
        default_connection:   travel
        connections:
            travel:
                driver:   "%tm_db_driver%"
                host:     "%tm_db_host%"
                port:     "%tm_db_port%"
                dbname:   "%tm_db_name%"
                user:     "%tm_db_user%"
```

```
            password: "%tm_db_password%"
            charset:  UTF8
        lifelong:
            driver:   "%ll_db_driver%"
            host:     "%ll_db_host%"
            port:     "%ll_db_port%"
            dbname:   "%ll_db_name%"
            user:     "%ll_db_user%"
            password: "%ll_db_password%"
            charset:  UTF8
orm:
    default_entity_manager:   travel
    entity_managers:
        travel:
            connection:       travel
            mappings:
                EldersPortalBundle: ~
        lifelong:
            connection:       lifelong
            mappings:
                EldersPortalBundle: ~
    auto_generate_proxy_classes: "%kernel.debug%"
```

4.2.2 Create New Entities

/src/Elders/PortalBundle/Entity/

Each new Object needs a new Entity.
In case of Travel Memories, we defined three new Entities:

- TravelMemory
- Countries
- TravelMemoryComment

In case of Lifelong Learning, we defined two new Entities:

- Lesson
- LessonCategory

The declaration of an Entity has to adapt to it's table attributes in the database.

- [Entity] TravelMemory – [travel DB Table] travelmemory
- [Entity] Countries – [travel DB Table] countries
- [Entity] TravelMemoryComment – [travel DB Table] travel_memory_comment

- [Entity] Lesson – [lifelong DB Table] lifelonglearning
- [Entity] LessonCategory – [lifelong DB Table] lesson_categories
-

In the PHP file of the Entity all parameters are private variables, which can be fetched and modified via ‘getter’ and ‘setter’ functions.

Example: the TravelMemory Entity

```
<?php
namespace Elders\PortalBundle\Entity;
use Doctrine\ORM\Mapping\ManyToOne;
use Doctrine\ORM\Mapping\JoinColumn;
use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity
 * @ORM\Table(name="travel_memory_comment")
 */
class TravelMemoryComment
{
    /**
     * @ORM\Id
     * @ORM\Column(type="integer")
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

          /**
     * @ORM\ManyToOne(targetEntity="TravelMemory")
     * @JoinColumn(name="memoryId", referencedColumnName="memoryId")
     */
    private $memoryId;

    /**
     * @ORM\Column(type="string", length=1000)
     */
    private $comment;

    /**
     * @ORM\Column(type="string", length=100)
```

```
     */
    private $userId;

    /**
     * Set memoryId
     *
     * @param \Elders\PortalBundle\Entity\TravelMemory $memoryId
     * @return TravelMemoryComment
     */
    public function setMemoryId(\Elders\PortalBundle\Entity\TravelMemory $memoryId = null)
    {
        $this->memoryId = $memoryId;
        return $this;
    }

    /**
     * Get memoryId
     *
     * @return \Elders\PortalBundle\Entity\TravelMemory
     */
    public function getMemoryId()
    {
        return $this->memoryId;
    }

    /**
     * Get id
     *
     * @return integer
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set comment
     *
     * @param string $comment
     * @return TravelMemoryComment
     */
```

```
    public function setComment($comment)
    {
        $this->comment = $comment;
        return $this;
    }

    /**
     * Get comment
     *
     * @return string
     */
    public function getComment()
    {
        return $this->comment;
    }

    /**
     * Set userId
     *
     * @param string $userId
     * @return TravelMemoryComment
     */
    public function setUserId($userId)
    {
        $this->userId = $userId;
        return $this;
    }

    /**
     * Get userId
     *
     * @return string
     */
    public function getUserId()
    {
        return $this->userId;
    }
}
```

4.2.3 Define Forms

/src/Elders/PortalBundle/Form/

If the Application wants to get input data from the user, we need to define the Form for this data set. Symfony's form builder interface is a great tool to create new forms easily.
In case of Travel Memories we need a Form to create a new Travel Memory. In this Form we'd like to get all the necessary data of a Travel Memory from the user.

Example: the Form to create new Travel Memory

```
public function buildForm(FormBuilderInterface $builder, array $options) {
	$userId = '';
	$albumId='';

	if(!$options['my_user'] == null) {
		$userId=$options['my_user']->getId();
	}

	$builder->add('userId', 'hidden', array('data'=>$userId));
	$builder->add('albumId', 'hidden', array('data'=>$albumId));
	$builder->add('categoryId', 'choice', array(
		'choices' => array(
			'0' => 'I was here',
			'1' => 'I lived here',
			'2' => 'I want to go here',
			'3' => 'I live here now'),
		'multiple' => false,
		'expanded' => true,
		'attr' => array('class' => 'memoryCategory'),
		'label' => ' ',
		'required' => true
	));
	$builder->add('fromDate', 'date', array(
		'label' => 'From:',
		'widget' => 'choice',
		'input'  => 'array',
		'format' => 'yyyy-MM-dd',
		'empty_value' => array('year' => 'Year', 'month' => 'Month', 'day' => 'Day')));
	$builder->add('toDate', 'date', array(
		'label' => 'To:',
		'widget' => 'choice',
		'input'  => 'array',
		'format' => 'yyyy-MM-dd',
		'empty_value' => array('year' => 'Year', 'month' => 'Month', 'day' => 'Day')));
	$builder->add('description', 'textarea', array(
```

```
                'attr' => array(
                        'placeholder' => 'Write your thoughts about this country...',
                        'class' => 'memoryDescription'),
                        'label' => 'Description: ',
                        'required'  => false
                ));
}
```

4.2.4 The Routing

/src/Elders/PortalBundle/Resources/config/routing/routing.yml

In Symfony we need to define every URL routs that we'd like to use on the site. These routing definitions call the proper Controller and set the right URL pattern for the given URLs.
These routings can be also used for AJAX calls.

In case of Travel Memories we defined the following routings

```
_travel_memories:
  pattern: /travel_memories
  defaults:  { _controller: EldersPortalBundle:TravelMemories:travel_memories }

_travel_memories_get_memory:
  pattern: /travel_memories_get_memory
  defaults:  { _controller: EldersPortalBundle:TravelMemories:getMemory }

_travel_memories_post_memory:
  pattern: /travel_memories_post_memory
  defaults:  { _controller: EldersPortalBundle:TravelMemories:postMemory }

_travel_memories_check_already_exists:
  pattern: /travel_memories_check_already_exists
  defaults:  { _controller: EldersPortalBundle:TravelMemories:checkMemoryAlreadyExists }

_travel_memories_update_memory:
  pattern: /travel_memories_update_memory
  defaults:  { _controller: EldersPortalBundle:TravelMemories:updateMemory }

_travel_memories_upload_photo:
  pattern: /_travel_memories_upload_photo
  defaults:  { _controller: EldersPortalBundle:TravelMemories:uploadPhoto }

_travel_memories_delete_memory:
```

```
  pattern: /travel_memories_delete_memory
  defaults:  { _controller: EldersPortalBundle:TravelMemories:deleteMemory }

_travel_memories_create_comment:
  pattern: /_travel_memories_create_comment
  defaults:  { _controller: EldersPortalBundle:TravelMemories:createComment }
```

In case of Lifelong Learning we defined the following routings

```
_lifelong_learning:
  pattern: /lifelong_learning
  defaults:  { _controller: EldersPortalBundle:LifeLongLearning:lifelong_learning }

_lifelong_learning_get_lessons:
  pattern: /lifelong_learning_get_lessons
  defaults:  { _controller: EldersPortalBundle:LifeLongLearning:lifelong_learning_get_lessons }

_lifelong_learning_increase_views:
  pattern: /lifelong_learning_increase_views
  defaults:  { _controller:
EldersPortalBundle:LifeLongLearning:lifelong_learning_increase_views }

_lifelong_learning_search:
  pattern: /lifelong_learning_search
  defaults:  { _controller: EldersPortalBundle:LifeLongLearning:lifelong_learning_search }
```

4.2.5 The Controller

/src/Elders/PortalBundle/Controller/

The main logic and the functions of the Application is defined in the Controller. All the Elder-Spaces Entities, Forms and Managers which are used by the application have to be included at the beginning.

In case of Travel Memories we need to

- do People requests (get the user's and friends data), Album and MediaItem requests (upload and get photos connecting to the Memory) connecting to the main databases via the Middleware layer's API
- do Travel Memory related requests (save new memory, comment on a memory, get friends' memories, etc.) connecting to the application database via Doctrine
- work with TravelMemory, Countries, TravelMemoryComment, OSPerson, OSAlbum and OSMediaItem Entites
- use the TravelMemories Form

Example: Controller includes

```
namespace Elders\PortalBundle\Controller;

use Symfony\Component\HttpFoundation\Response;
use Elders\PortalBundle\Manager\RequestPeopleManager;
use Elders\PortalBundle\Manager\RequestAlbumManager;
use Elders\PortalBundle\Manager\RequestMediaItemManager;
use Elders\PortalBundle\Entity\TravelMemory;
use Elders\PortalBundle\Entity\TravelMemoryComment;
use Elders\PortalBundle\Entity\Countries;
use Elders\PortalBundle\Entity\OSPerson;
use Elders\PortalBundle\Entity\OSAlbum;
use Elders\PortalBundle\Entity\OSMediaItem;
use Elders\PortalBundle\Form\TravelMemoriesForm;
```

In the Controller we need to define an Action to each Routing. The names of these Actions have to be the same that we gave in the _controller parameters of the routing.yml.
For example in the *_travel_memories_post_memory* routing we wrote *EldersPortalBundle:TravelMemories:postMemory* as the value of the *_controller* parameter, so we need a function named *postMemoryAction* in the Controller.

Example: the postMemoryAction function

```
public function postMemoryAction(){
        $request = $this->getRequest();
        $countryId = $request->query->get('countryId');
        $em = $this->getDoctrine()->getManager('travel');

        $country = new Countries();
        $country = $em->find('EldersPortalBundle:Countries', $countryId);

        if(!$country){
                $country = new Countries();
                $country->setId($countryId);
                $country->setLatitude(floatval($request->query->get('lat')));
                $country->setLongitude(floatval($request->query->get('long')));

                $em->persist($country);
        }

        $tm = new TravelMemory();
        $tm->setCreatedTime($request->query->get('createdTime'));
        $tm->setCountryId($country);
```

```
        $tm->setUserId($request->query->get('userId'));
        $tm->setCategoryId($request->query->get('categoryId'));
        $tm->setFromDate($request->query->get('fromDate'));
        $tm->setToDate($request->query->get('toDate'));
        $tm->setDescription($request->query->get('description'));
        $tm->setAlbumId($request->query->get('albumId'));

        $em->persist($tm);
        $em->flush();
        return new Response('Created memory id '.$tm->getMemoryId());
}
```

As you can see in the previous example, we use Symfony's built-in Doctrine service for the application database requests. We set the proper entity manager (travel) for the request as an input parameter:

```
$em = $this->getDoctrine()->getManager('travel');
```

4.2.6 Create the Layout

/src/Elders/PortalBundle/Resources/views/Apps/

Now we have everything to create the layout of the Application. We use HTML, JavaScript and PHP codes here to define the looks of the Application and we chose JSON as the format of the communication.

Example: how to get the proper Routing

```
"<?php echo $view['router']->generate('_travel_memories') ?> "
```

Example: how to create a Form

```
<form action="<?php echo $view['router']->generate('_travel_memories') ?>"
        method="post" <?php echo $view['form']->enctype($TravelMemoriesForm) ?>>
        <?php echo $view['form']->widget($TravelMemoriesForm); ?>
        <input type="submit" />
</form>
```

Example: how to make an AJAX request to a Controller (JavaScript)

```
function saveMemory (countryCode, long, lat) {
…
var postUrl = "<?php echo $view['router']->generate('_travel_memories_post_memory'); ?>";
var params = {
        createdTime:time,
```

```
        countryId:countryCode,
        userId:"<?php echo $me->getId(); ?>",
        categoryId:$("#TravelMemoriesForm_categoryId input[type='radio']:checked").val(),
        fromDate:$("#TravelMemoriesForm_fromDate_year").val()+"-
"+$("#TravelMemoriesForm_fromDate_month").val()+"-
"+$("#TravelMemoriesForm_fromDate_day").val(),
        toDate:$("#TravelMemoriesForm_toDate_year").val()+"-
"+$("#TravelMemoriesForm_toDate_month").val()+"-
"+$("#TravelMemoriesForm_toDate_day").val(),
        description:$("#TravelMemoriesForm_description").val(),
        albumId:0,
        lat:lat,
        long:long
};
params = jQuery.param(params);
var saveMemoryReq = $.post(postUrl+"?"+params, "json");
saveMemoryReq.done(function(data) {
        $('#jquery-wrapped-fine-uploader').fineUploader('uploadStoredFiles');
        getMarkers();
        infoWindow.close();
});
}
```

4.2.6.1 Travel Memories Specifics

In case of Travel Memories we use Google Maps API for the world map with a KML layer (travel_memories.kmz – generated by KMLer http://www.mi-perm.ru/gis/programs/kmler) on the map canvas to separate the countries.

About the usage of the Google Maps KML layer a specification can be found here:
https://developers.google.com/maps/documentation/javascript/layers#KMLLayers

Example: JavaScript map initialization via Google Maps API using a KML layer

```
map = new google.maps.Map(document.getElementById("map-canvas"), mapOptions);
var georssLayer = new google.maps.KmlLayer({
        url: 'http://kond.origoapps.iwiw.hu/travel_memories.kmz',
        map: map,
        suppressInfoWindows: true,
        preserveViewport: true
        });

$("#dialog-memories").dialog({
        resizable: false,
```

```
        autoOpen: false,
        height:400,
        width: 790,
        modal: true,
        position: { my: "center center", at: "center center", of: "#map-canvas" },
        open: function(){
        jQuery('.ui-widget-overlay').bind('click',function(){
                jQuery('#dialog-memories').dialog('close');
        })
}
});

google.maps.event.addListener(georssLayer, 'click', function(kmlEvent) {
        createInfoWindowContent(kmlEvent.featureData, kmlEvent.latLng);
});
```

Example: a country entity in the KML

```
<S_country>
        <name>Hungary</name>
        <visibility>0</visibility>
        <description><![CDATA[<a href='http://www.google.com/search?q=Hungary'>Search
Google</a> <BR><a
href='http://maps.google.com/?ll=47.1677438395918,19.4255653678934&spn=0.02,0.02&t=h'>
Maps Google</a> <BR>]]></description>
        <styleUrl>#khStyle548_3</styleUrl>
        <Style>
                <PolyStyle>
                        <color>78b0f1ae</color>
                </PolyStyle>
        </Style>
        <Polygon>
                <outerBoundaryIs>
                        <LinearRing>
                                <coordinates>
                                        20.26102,46.11485,0 19.56528,46.17278,0
                                18.81702,45.91296000000001,0 18.4075,45.74833,0
                                17.66097,45.83875,0 16.60787,46.47623,0 16.34903,46.84236,0
                                16.1118,46.86972000000001,0 16.50486,47.00676000000001,0
                                16.45201,47.41284,0 16.71389,47.54388,0 16.45055,47.69805,0
                                17.05389,47.70944000000001,0 17.16639,48.0125,0
                                17.78728,47.74643,0 18.66305,47.75972,0
                                18.84972000000001,47.81777,0 18.84534,48.04913,0
```

```
                                    19.47243,48.08944000000001,0 19.63055,48.23389,0
                                    19.93986,48.13625,0 20.65291,48.56166000000001,0
                                    21.43819,48.57534,0 21.78097,48.34069,0
                                    22.15144,48.41192000000001,0 22.8948,47.95454,0
                                    22.01389,47.51062000000001,0
                                    21.17736,46.29736000000001,0 20.26102,46.11485,0
                                    </coordinates>
                            </LinearRing>
                        </outerBoundaryIs>
                        <tesselate>1</tesselate>
                </Polygon>
                <FIPS_CNTRY>HU</FIPS_CNTRY>
                <GMI_CNTRY>HUN</GMI_CNTRY>
                <ISO_2DIGIT>HU</ISO_2DIGIT>
                <ISO_3DIGIT>HUN</ISO_3DIGIT>
                <CNTRY_NAME>Hungary</CNTRY_NAME>
                <LONG_NAME>Hungary</LONG_NAME>
                <SOVEREIGN>Hungary</SOVEREIGN>
                <POP_CNTRY>10310410</POP_CNTRY>
                <CURR_TYPE>Forint</CURR_TYPE>
                <CURR_CODE>HUF</CURR_CODE>
                <LANDLOCKED>Y</LANDLOCKED>
                <SQKM>92174.03999999999</SQKM>
                <SQMI>35588.4</SQMI>
                <COLOR_MAP>4</COLOR_MAP>
</S_country>
```

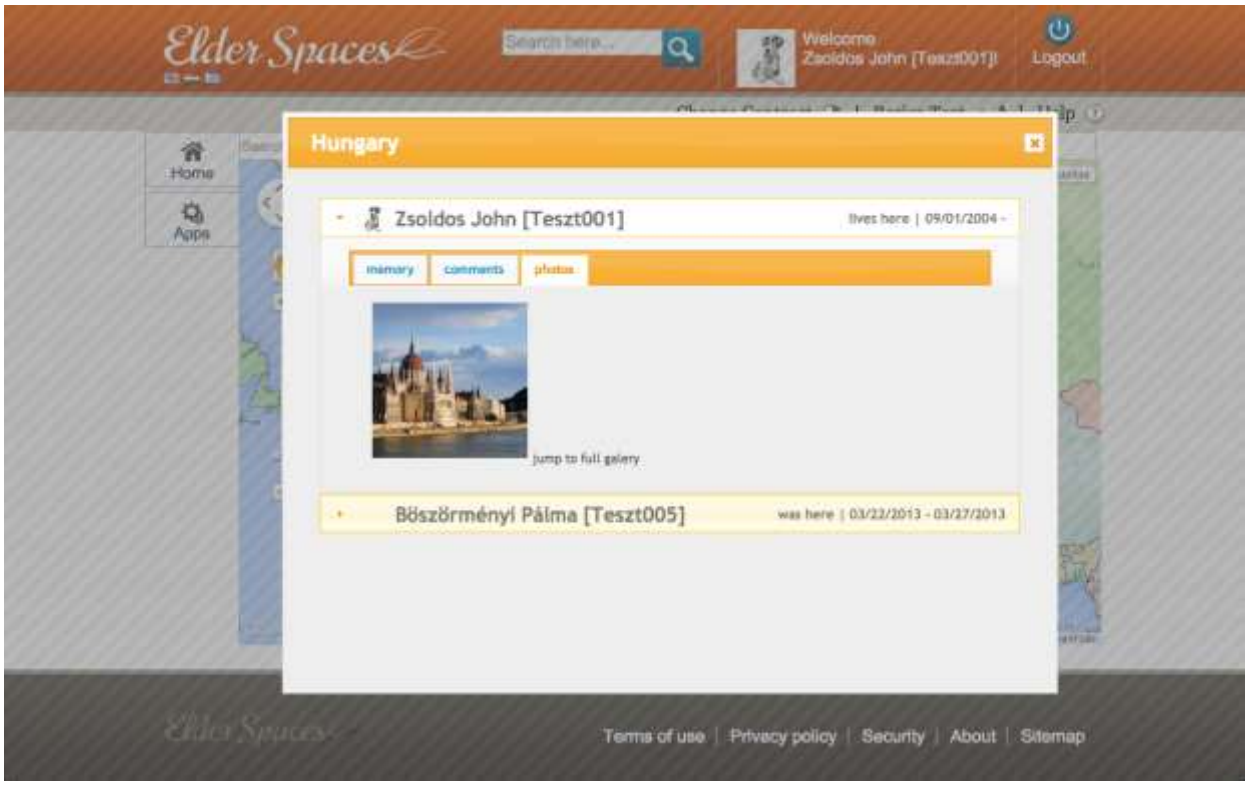
4.2.6.2 Lifelong Learning Specifics

In case of Lifelong Learning we use video embed codes from different providers to show video lessons to the users. We upload the list of video lessons manually into the system to avoid "trash" contents. We also set the list of languages in which the video lesson can be understandable; so the list of languages which the video has a sound track or subtitle. The application fetches which languages the user can speak from the user's profil, and shows only the videos that are understandable for the given user.

4.3 Screenshots

4.3.1 Travel Memories

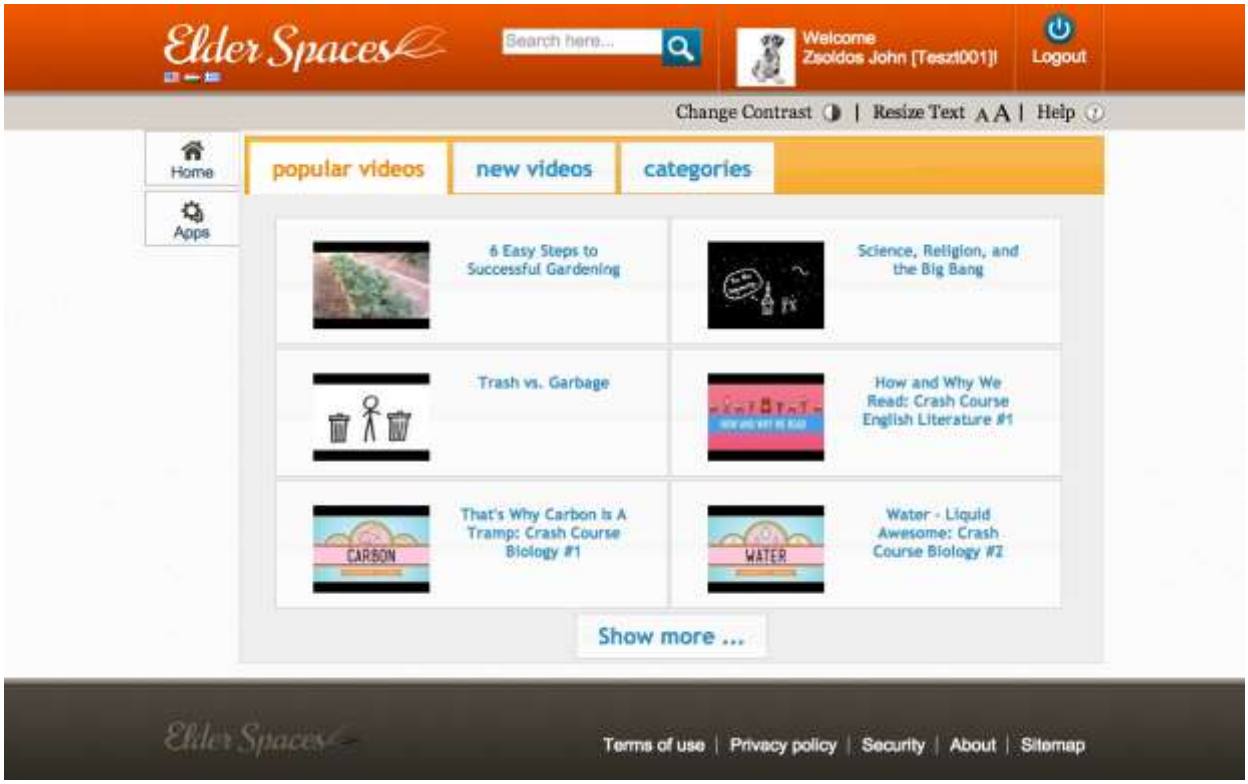
The main screen. Green signs show the countries where the user has an own Memory, yellow signs shows the countries where the user's friends' have Memories.



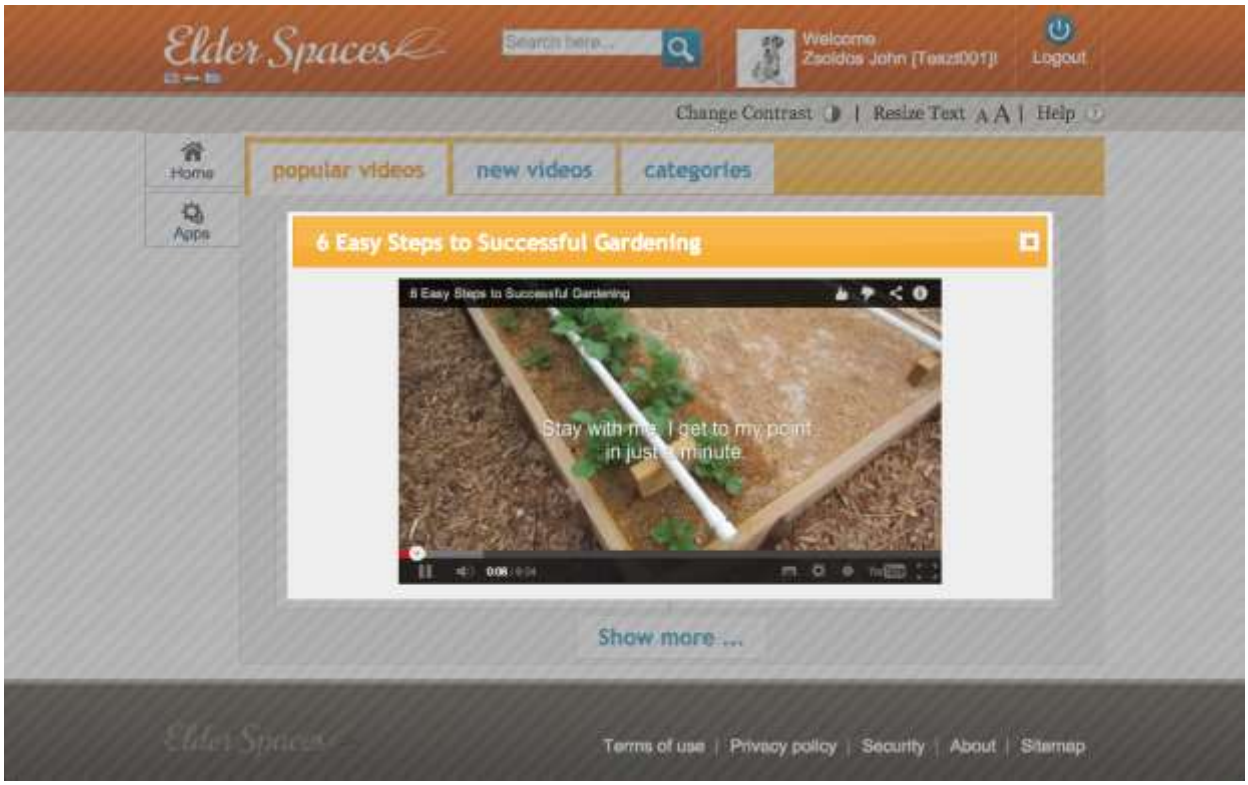
Travel Memories screen. The Travel Memories of the selected country are listed here. By clicking on the bar of a user's Travel Memory we can read the selected user's description of his/her memories regarding to the given country; we can read the comments of the Memory and also leave our own comments as well; and we can see the photos of the Memory uploaded by the owner user. When a user uploads photos to a country in Travel Memories, the photos upload to the user's profile as well in a new album with the name of the country.

Creating new Memory screen.

4.3.2 Lifelong Learning



The main screen of Lifelong Learning.



The video watching screen.

The categories screen.

5. Conclusions

Applications are good for expand the basic social functionality of the Elder-Spaces platform. It's easy to create and integrate new Applications thanks to the Middleware API and the well organized structure of Symfony framework.

The possibility of creating new and exciting Applications in Elder-Spaces platform is given by the wide variety of social functions. We can cogitate new Application ideas using Events, Groups, People, Activites, Notifications, Albums, Photos or even Messages, there's no functionality of Elder-Spaces that an Application can't use.

Appendix

References