Deliverable 5.5 LinkedTV front-end: video player and MediaCanvas API version 2

Daniël Ockeloen

Pieter van Leeuwen

19th September 2014

Work Package 5: LinkedTV Platform

LinkedTV

Television Linked To The Web

Integrated Project (IP)

FP7-ICT-2011-7. Information and Communication Technologies

Grant Agreement Number 287911

Dissemination level[1]	PU
Contractual date of delivery	*31st March 2014 (agreed postponement to 31st August)*
Actual date of delivery	*19th September 2014*
Deliverable number	D5.5
Deliverable name	*LinkedTV front-end: video player and MediaCanvas API version 2*
File	*LinkedTV_D5.5.docx*
Nature	*Prototype*
Status & version	*Final*
Number of pages	*32*
WP contributing to the deliverable	*WP5*
Task responsible	*Noterik*
Authors	*Daniël Ockeloen, Pieter van Leeuwen* *Noterik*
Other contributors	*-*
Reviewer	*Jan Thomsen, Condat*
EC Project Officer	*Thomas Küpper*

[1] • PU = Public
• PP = Restricted to other programme participants (including the Commission Services)
• RE = Restricted to a group specified by the consortium (including the Commission Services)
• CO = Confidential, only for members of the consortium (including the Commission Services))

Table of Contents

Table of Figures

1 Introduction

The LinkedTV media player and API has evolved from a single player and limited API in version 1 to a toolkit to allow rapid development and creation of different kind of applications within the HTML5 / multiscreen space. The main reason for this transition is that during the course of the Linked TV project different partners had different requirements for their scenarios. Instead of trying to fit all these requirements into one player and, most likely, compromise on the functionalities of the scenarios we wanted to offer something that would allow all partners a satisfiable solution.

Therefore the Springfield Multiscreen Toolkit, or short SMT, has been developed. The aim for the SMT was to allow flexibility for developing multiscreen applications. Also from a commercial point of view a toolkit with examples is more interesting than a pure player as it gives the freedom of developing new ideas with the LinkedTV platform. This allows for a more natural fit with the LinkedTV platform that also is not a fixed solution but allows for a flexible use of its modules.

This document will focus on the technical details of how the LinkedTV frontend is currently operating and how the different prototypes are built with the help of the SMT. A broader technical description of the entire LinkedTV platform can be found in D5.6 *Final LinkedTV end-to-end platform*. Here we will discuss our efforts in the process of making both the SMT and the required Springfield framework available as open source software (http://noterik.github.io) combined with outlining our future plans ensuring the usage of both. As a result this deliverable should be seen as a combination of the work we have done making the toolkit open source, creating the GitHub for both the toolkit and the underlying software layers and adding examples, wiki pages and this document.

1.1 Related Linked TV deliverables

The design requires the consideration and input of most of the LinkedTV packages. However, D5.5 strongly relates to the following deliverables in particular:

- *D3.6 LinkedTV interface and presentation engine (version 2)*
- *D5.2 LinkedTV Front-end: Video Player and MediaCanvas API*
- *D5.6 Final LinkedTV End-to-end Platform*

1.2 History of the document

Table 1: History of the document

Date	Version	Name	Comment
2014/06/23	V0.0.1	Pieter van Leeuwen, Noterik	Created initial document structure
2014/07/07	V0.1.0	Pieter van Leeuwen, Noterik	Updated 1, 2, 2.1, 2.2
2014/07/15	V0.2.0	Pieter van Leeuwen, Noterik	Updated 2.1
2014/07/21	V0.3.0	Pieter van Leeuwen, Noterik	Updated 2.1, 2.2
2014/08/02	V0.4.0	Daniel Ockeloen, Noterik	Updated 2.2.2, 2.3, 3
2014/08/04	V0.4.1	Pieter van Leeuwen, Noterik	Updated 4
2014/08/14	V0.5	Daniel Ockeloen, Noterik	Updated 5
2014/08/22	V0.6	Pieter van Leeuwen, Noterik	Updated 2 with information about Entity Proxy API.
2014/08/29	V0.6.1	Pieter van Leeuwen, Noterik	Updated 2.4
2014/09/02	V0.6.2	Jan Thomsen, Condat	Q&A
2014/09/16	V0.7	Daniel Ockeloen, Pieter van Leeuwen, Noterik	Updated according to comments from Q&A
2014/09/18	V0.7.1	Pieter van Leeuwen, Noterik	Updated with feedback from Lyndon Nixon, Modul

1.3 Abbreviations and Acronyms

Table 2: Abbreviations

Abbreviation	Explanation
API	Application Programming Interface
CDN	Content Delivery Network
Codec	Encoder / decoder for a digital data stream
DRM	Digital Rights Management
GAIN	General Analytics Interceptor
HbbTV	Hybrid broadcast broadband TV
HTML	HyperText Markup Language
IDE	Integrated development environment
J2EE	Java 2 Enterprise Edition
NIC	Network Interface Component
REST	Representational State Transfer
SME	Small and Medium Enterprises
SMT	Springfield Multiscreen Toolkit
URI	Uniform Resource Identifier
War	Web Archive
XML	Extensible Markup Language

2 Architecture

This section describes the architecture of the Springfield framework [1], the framework that the SMT uses as a basis. The main focus will be on the SMT, however for a better understanding a complete overview of the Springfield framework is given. The service-oriented architecture of the framework offers a complete video framework that is capable of handling all that is needed to store, process and stream a video. The following services are available in the framework as can be seen in Figure 1.

- Marge & Homer – Intra service communication
- Bart – Proxy communication
- Smithers – Database for metadata storage
- Eddie & Lou – Multiscreen toolkit
- Momar – Transcoding service
- Barney – User manager
- Lisa – search engine
- Nelson – Image extraction
- Edna – Image transformation and caching service
- Flanders – File metadata extraction
- Rafael – Media fragment service

The Springfield framework is together with the SMT being released as open source, see chapter 3 for more detailed information about this process.

Figure 1: Overview of the Springfield framework

For a complete understanding of the role of the Springfield framework Figure 2 depicts the overall relationship between the LinkedTV Platform and the Springfield Framework.

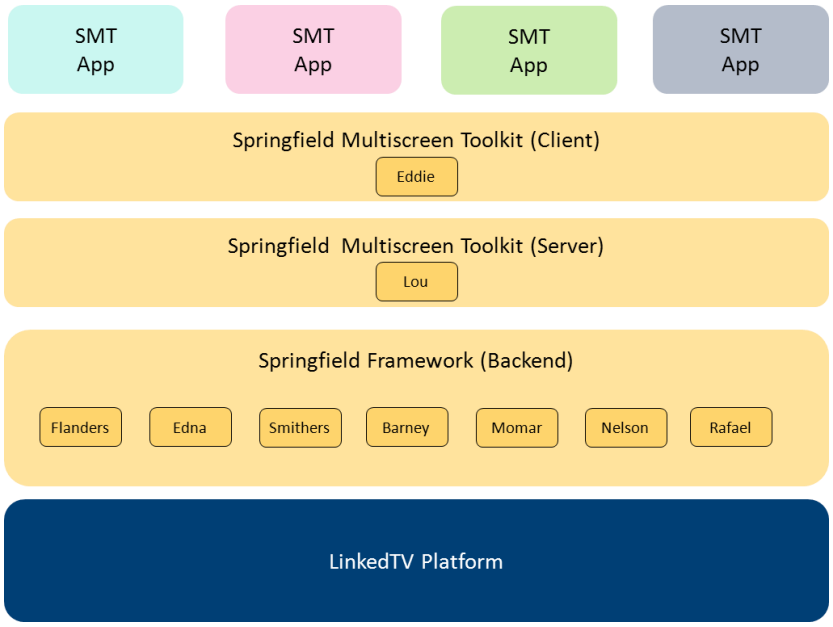

Figure 2: Relationship between the LinkedTV Platform and the Springfield framework

2.1 Springfield Framework

To adapt the Springfield framework for use with the LinkedTV platform integration was needed for being able to retrieve and process the videos inserted through the LinkedTV platform. For this an import service has been created that retrieves new videos from the LinkedTV platform. Once a video is imported the Springfield framework can start handling the video. In the next sections the Springfield services that handle the video are being discussed. Once the video from the LinkedTV platform is successfully processed by the framework it will return locators for both the videos and the thumbnails back into the platform.

2.1.1 Smithers

Smithers is the display database that stores all meta information about the video. It also maintains a local cache for most relevant information from the LinkedTV platform. This information is used by the LinkedTV players that are being developed on top of the SMT.

2.1.2 Barney

Barney is the user manager that handles user authentication and authorization for the users in the LinkedTV platform. Besides basic user information like username, one-way hashed passwords are stored in Barney, for the rest no user information is stored in Barney for privacy reasons.

2.1.3 Momar

Momar is the video transcoding service that can transcode a source video into different video formats and qualities to meet demands for (online) delivery of the video. On default videos

are transcoded in four different qualities ranging from PAL to full HD to provide adaptive streaming to any kind of device and internet connection.

2.1.4 Nelson

Nelson is the image extraction service that on default creates a thumbnail for every second of video that is being inserted in the Springfield framework. These thumbnails can be used to give a quick impression of a certain moment in the video without the need to load the video.

2.1.5 Edna

Edna handles image transformations and caching. Transformations that can be applied with Edna are image resizing and rotation. These images are also cached for performance reasons.

2.1.6 Flanders

Flanders is a metadata extraction service that extracts information about the provided video, e.g. the video codec being used, the video frame rate and the video resolution. Based on the information Momar can decide to skip processing a certain quality as the source video might already meet the desired requirements.

2.1.7 Rafael

Rafael is the media fragment service that offers progressive video for video players. It offers support for temporal media fragments[2] in a video. Also included in this service is a ticket mechanism that provides authorized access to videos. This is to meet the demands from content providers that require a form of protection on their content.

2.2 Springfield Multiscreen Toolkit

The Springfield Multiscreen Toolkit, the successor of the MediaCanvas API, can be divided into three parts. First a server side part called Lou that runs the application and handles the multiscreen aspects across multiple devices. Second a client side part called Eddie that offers client side support specific for the device, e.g. gestural support for tablets or remote support in case of HbbTV. Finally there is an application manager that allows developers to deploy, test and debug their applications.

The SMT uses a unified screen model, meaning that developers don't have to deal with multiscreen aspects like for example synchronization and the addressing of events to a certain screen. This is to allow developers to work as they were developing a single screen application while the SMT deals with the multiscreen aspects. The different screens all reside on the server in their own application space. From the server the screens are pushed to the

[2] http://www.w3.org/TR/media-frags/

client side as can be seen in Figure 3. The advantage here is that the client side requirements can be lowered as the server can render certain parts of the screen, this leads to support of a broader range of devices.

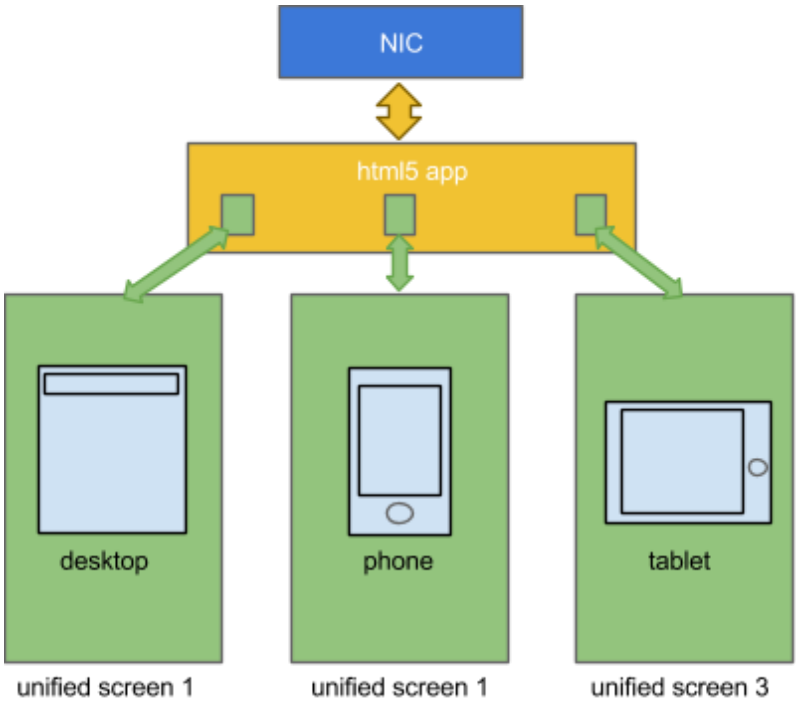

Figure 3: Communication from the application to connected unified screens

2.2.1 Application manager

The application manager was developed to allow developers to easily and quickly test their applications. In the application manager developers can use three different pages. The first (overview) page shows the applications that are currently active on the server. This means that the application has been opened at least once. Furthermore the current number of screens and connected users can be viewed and logging information is available. Also there is the possibility to locate all the screens. By pressing the related button an audio signal will be played on all connected screens so developers can find all their devices. The second page shows all the applications available on the server including versioning. From here it is possible to deploy new applications, define if the application is a development or production version and configure if new uploads should be auto deployed as development or a production version. The third page allows for user management where users can be added and altered. An impression of the pages from the application manager can be seen in Figure 4.

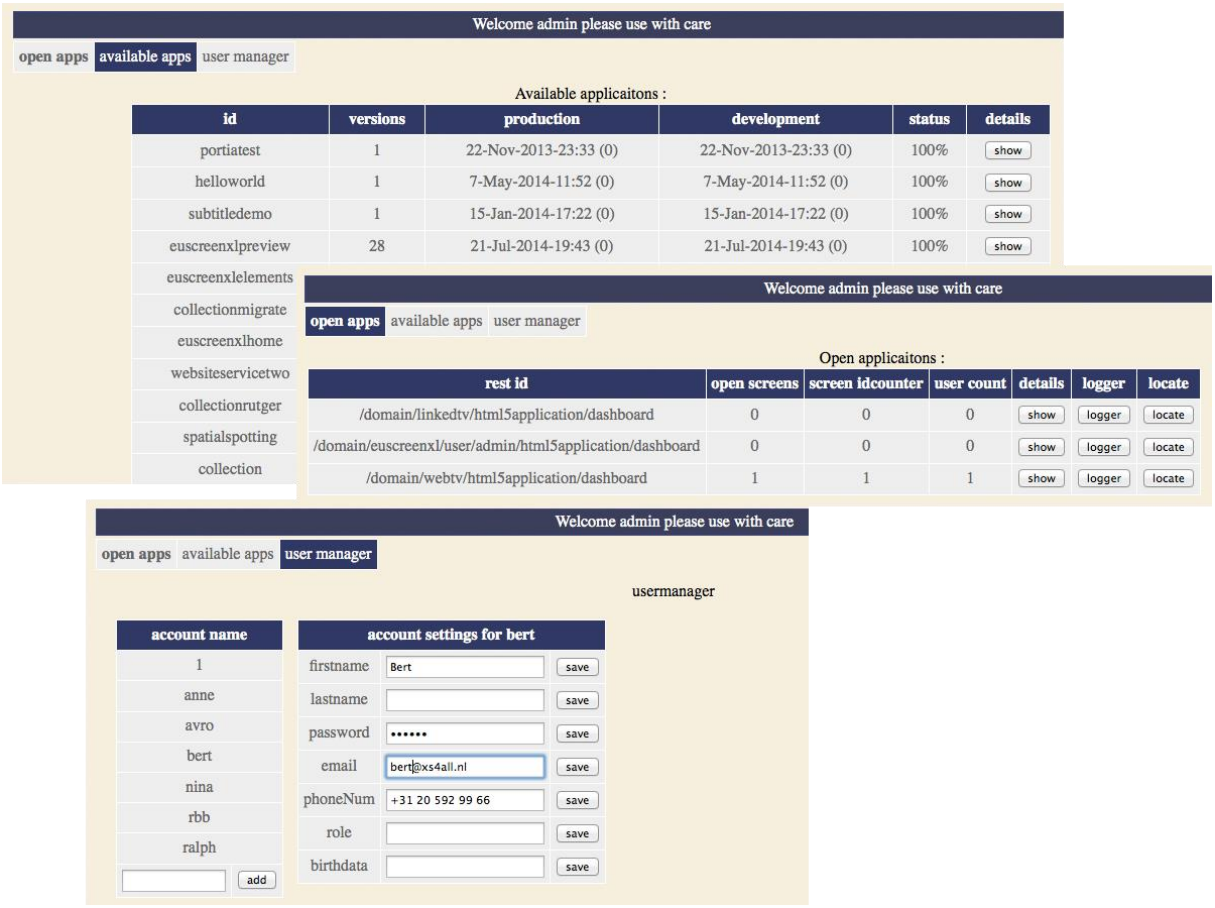

Figure 4: Overview of different pages of the application manager

Demo applications are provided with an Ant[3] build file that will pack the application together with the client side resources (images, style sheets, client side scripts) in a war file. However unlike normal war files that are deployed directly on a J2EE web server the war file needs to be uploaded through the application manager that is part of the SMT. When an application is uploaded the application manager moves the resources to the correct locations and keeps track of different versions. For every application a limited amount of versions is kept, at least one version per month, week and day is kept so that developers can always roll back to a previous version if needed. Updated applications are directly available to use after deployment.

2.2.2 Lou

Lou together with Eddie form the core of the SMT. Its purpose is to create a seamless and easy to use programming model for developers of multiscreen applications to work with. It does this by taking away programming tasks and by providing access to APIs in the Springfield framework, to services and APIs in the LinkedTV platform and even to external

[3] http://ant.apache.org

known APIs. In this document we have tried to give an overview of all the parts that - together with the wiki's and GitHub - are enough to signal that this is a large framework. It is Lou's job to abstract and hide underlying services in an application structure that is easy to understand for developers and provide an easy manner of testing and deploying the applications developers build using the application manager explained in section 2.2.1.

Herein lays the contradiction in that we need to teach developers a new way of thinking on how to develop and on how to coordinate multiple screens versus keeping it simple and using known methods. So next to the abstraction we tried to stay as close to what most developers know in the tools they use (IDE's like Eclipse[4]), packaging systems (like war files) and basic elements as with HTML5's combination of HTML, JavaScript and CSS.

As a result it is best to see Lou as a taskmaster and coordinator that uses these basic HTML5 elements to load, move and destroy these known building blocks to/from the correct devices and screens when needed. It also allows you to keep your client very light since the filling of these elements can be done within Lou before they are send to the clients instead of that it needs to have the clients connect to the resources themselves. This is a key advantage since Lou as part of Springfield is already attached to all the other services and external APIs developed.

2.2.2.1 Application structure

Figure 5 shows the application structure for the helloworld sample application that is available as an introduction into the SMT. We added this as a screenshot because it should make developers feel comfortable since it shows a near normal directory structure of a packaging system that has been in use for over a decade. We see a web application with a *source file* area holding the 'controlling' java sources. We see *build*, *tools* and *libs* directories as they would expect and a *build file* they would normally use to build the application. We also see a *WebContent* directory as is normal within a web application holding the HTML5 parts (HTML, JavaScript and stylesheets) the controller will use on the devices and screens.

[4] https://www.eclipse.org/

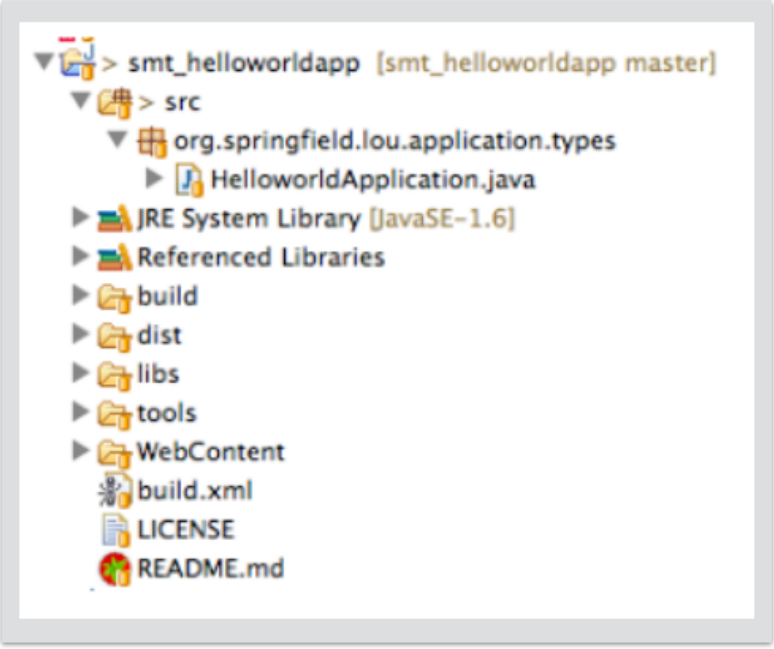

Figure 5: Structure of Lou for the *helloworld* application

To get a complete picture of how applications work we advise people to read the GitHub pages and checkout the examples we provide there. Figure 6 shows how little is needed for a basic application that already has multiscreen elements, loads a stylesheet, some content and sets a message of 'hello world' to each new screen that joins the application. The SMT application provides many of these 'handlers' like the 'onNewScreen' call where developers can react on changes without having to know all the underlaying work done by the framework.

```
* HelloworldApplication.java
package org.springfield.lou.application.types;
import org.springfield.lou.application.*;

public class HelloworldApplication extends Html5Application{

    public HelloworldApplication(String id) {
        super(id);
    }

    public void onNewScreen(Screen s) {
        loadStyleSheet(s, "generic");
        loadContent(s, "titlepart");
        s.setContent("defaultoutput", "Hello World !");
    }

}
```

Figure 6: The controller for the helloworld application

As a second example Figure 7 shows a very basic but real world second screen application where the main screen is shown on an HbbTV device and we use a tablet as a remote.

```
* BasicVideoApplication.java
package org.springfield.lou.application.types;
import org.springfield.fs.Fs;

public class BasicvideoApplication extends Html5Application{

    public BasicvideoApplication(String id) {
        super(id);
    }

    public void onNewScreen(Screen s) {
        Capabilities cap = s.getCapabilities();
        loadStyleSheet(s, "generic");
        int mode = s.getCapabilities().getDeviceMode();
        if (cap.getDeviceMode()==cap.MODE_HBBTV) {
            loadContent(s, "video");
            s.setRole("mainscreen");
            s.setContent("video",getCEHtmlVideoTag());
            this.componentmanager.getComponent("video").put("app", "play()");
            loadContent(s, "overlay");
        } else {
            s.setRole("controller");
            loadContent(s, "controller");
        }
    }
```

Figure 7: Part of a 'video' application for HbbTV with a second screen remote

In this case we show just the 'onNewScreen' call that is called by the platform when a new screen is detected. In this case this is about 50% of the whole app. You can see how easy it is to first query the capabilities of the screen in question and then load the correct elements that we want on it. You can see we check if it is an HbbTV device and if so we load the main video and set the role of the screen to main screen so we can easily talk to it based on that property at a later time. If it is not an HbbTV device we assume it is the controller (phone, tablet or desktop) and load up the controller that has buttons like play and pause.

2.2.2.2 Available APIs

As the SMT aims to be used not only by LinkedTV (see chapter 5) there has been decided to integrate LinkedTV specific features in external APIs instead of in the core of the toolkit. By means of this functionalities are available, but not interweaved into the toolkit. The different APIs are developed with requests and feedback from the different partners that have used the SMT for implementing their multiscreen applications.

Channel API

With television broadcasters in mind we defined channels and episodes, where a channel could be a continuous live television channel or a predefined channel. It contains a list of episodes for the defined channel.

```
public Channel(String domain, String channel)
```

Creates a channel for the given domain and channel name.

```
public List<Episode> getEpisodes()
```

Returns a list of episodes from the channel.

```
public Episode getLatestEpisode()
```

Returns the latest episode from the channel.

Episode API

Episodes hold not only the video but also related information like chapters, annotations and enrichments taken from the LinkedTV platform.

```
public Episode(String mediaResourceId)
```

Creates an episode for the given media resource id.

```
public String getMediaResourceId()
```

Returns the media resource id for this episode.

```
public String getStillsUri()
```

Returns the base URI for the video stills.

```
public int getDuration()
```

Returns the duration of the episode.

```
public String getStreamUri()
```

Returns the video stream URI.

```
public String getStreamUri(int quality)
```

Returns the video stream URI for the given quality. The following qualities are available for LinkedTV:

- 2 – 640x360, 800kbps
- 3 – 1280x720, 1.4Mbps
- 4 – 1920x1080, 2.7 Mbps
- 5 – 320x180, 100kbps

```
public String getTitle()
```

Returns the title for the episode.

```
public String getPresentationId()
```

Returns the presentation id for this episode for the Noterik display database.

```
public FSList getAnnotations()
```

Returns a FSList of annotations for the episode.

```
public FSList getChapters()
```

Returns a FSList of chapters for the episode.

```
public FSList getAnnotationsFromChapter(FsNode chapter)
```

Returns a FSList of annotations for the given chapter.

```
public FSList getEnrichmentsFromAnnotation(FsNode annotation)
```

Returns a FSList of enrichments for the given annotation.

GAIN API

For personalization as described in D4.6 it is required to communicate the user interactions with the player back to the LinkedTV platform, by means of this the platform can personalize the recommendations showed in the player. The events are communicated to GAIN [2] that processes the events.

```
public GAIN(String accountId, String applicationId)
```

Creates a GAIN object with the given account id and application id.

```
public void application_new()
```

Signals the new application has been created in Lou.

```
public void application_remove()
```

Signals the application has been removed from Lou.

```
public void screen_new(String screenId)
```

Signals a new screen has been created with the given screen id.

```
public void screen_remove(String screenId)
```

Signals the given screen has been closed.

```
public void screen_orientation(String screenId, String orientation)
```

Signals the given screen has changed orientation. Currently only supports “portrait” and “landscape”.

```
public void player_play(String screenId, String mediaresourceId,
```

```
String videoTime)
```

Signals a video started playing on the provided screen, with the given media resource id at the given video time.

```
public void player_pause(String screenId, String mediaresourceId,
String videoTime)
```

Signals a video was paused on the provided screen, with the given media resource id at the given video time.

```
public void player_stop(String screenId, String mediaresourceId,
String videoTime)
```

Signals a video was stopped at the provided screen, with the given media resource id at the given video time.

```
public void user_login(String userId, String screenId)
```

Signals a user logged in with the given user id at the provided screen.

```
public void user_logout(String userId, String screenId)
```

Signals a user logged out with the given user id at the provided screen.

```
public void user_bookmark(String userId, String objectId, String
screenId)
```

Signals the given user bookmarked the given object at the provided screen.

```
public void user_select(String userId, String objectId, String
screenId)
```

Signals the given user selected the given object at the provided screen.

```
public void user_viewtime(String userId, String objectId, String
screenId, String viewtime)
```

Signals the view time of the given object by the given user on the provided screen.

```
public void updateEntities(List<GAINObjectEntity> entities)
```

Update the list of entities that is currently on a screen.

```
public void sendContextRequest(String context, String videoTime)
```

Forward an external context request with the current video time.

```
public void sendKeepAliveRequest()
```

Sends a keep alive request so GAIN nows the application is still being used.

Entity proxy

As the players need more information to show besides the annotations and enrichments an entity proxy has been created within the LinkedTV project that gets relevant information

about a DBPedia[5] entry from an enrichment. The entity proxy currently supports two levels of information retrieval. On default a label, thumbnail URI and abstract are provided. For persons this information is extended with birth date, death date, birth place, death place, nationality and profession. For artists also the art styles are provided.

```
public EntityProxy(String dbpediaUri)
```

Creates a new entity proxy object with the given dbpedia URI.

```
public String getLabel()
```

Get the label from the dbpedia entry.

```
public String[] getThumbnailUri()
```

Get a list of thumbnail URIs from the dbpedia entry.

```
public String getAbstract()
```

Get the abstract from the dbpedia entry.

```
public String getBirthDate()
```

Get the birth date from the dbpedia entry if the entity is a person.

```
public String getDeathDate()
```

Get the death date from the dbpedia entry if the entity is a person.

```
public String getBirthPlace()
```

Get the birthplace from the dbpedia entry if the entity is a person.

```
public String getDeathPlace()
```

Get the death place from the dbpedia entry if the entity is a person.

```
public String getNationality()
```

Get the nationality from the dbpedia entry if the entity is a person.

```
public String[] getProfession()
```

Get a list of professions from the dbpedia entry if the entity is a person.

```
public String[] getArtStyle()
```

Get a list of art styles from the dbpedia entry if the entity is an artist.

[5] http://dbpedia.org/

2.2.3 Eddie

Eddie is the client side part of the SMT. The core functionality of Eddie is to open a communication channel with Lou, in order to have the multiscreen functionalities on the client. Once the initial communication channel has been established with a new screen, components are being pushed to the client using the communication channel that has been created. The components that are being pushed are depending on the application, the role of the screen (e.g. main screen, second screens) and the capabilities of the device.

2.2.3.1 Structure

Figure 8 shows the structure of Eddie for the *demolinkedtv* example application. On the application level so-called actionlists can be defined that script actions that are coming from the SMT. In the example a handler for a new screen is defined. Components are what the client will see as output. The different component can be used on one or all types of screens, depending on the application. Every component can consist of both an HTML template and a JavaScript that handles additional client side actions. Further style sheets can be incorporated in the application. Several style sheets can be defined for different devices; these are automatically pushed if the capabilities of a device are detected by the SMT. Finally generic client side libraries that are not part of a component can be added. In the example application in Figure 8 additional libraries for handling gestures are being added.

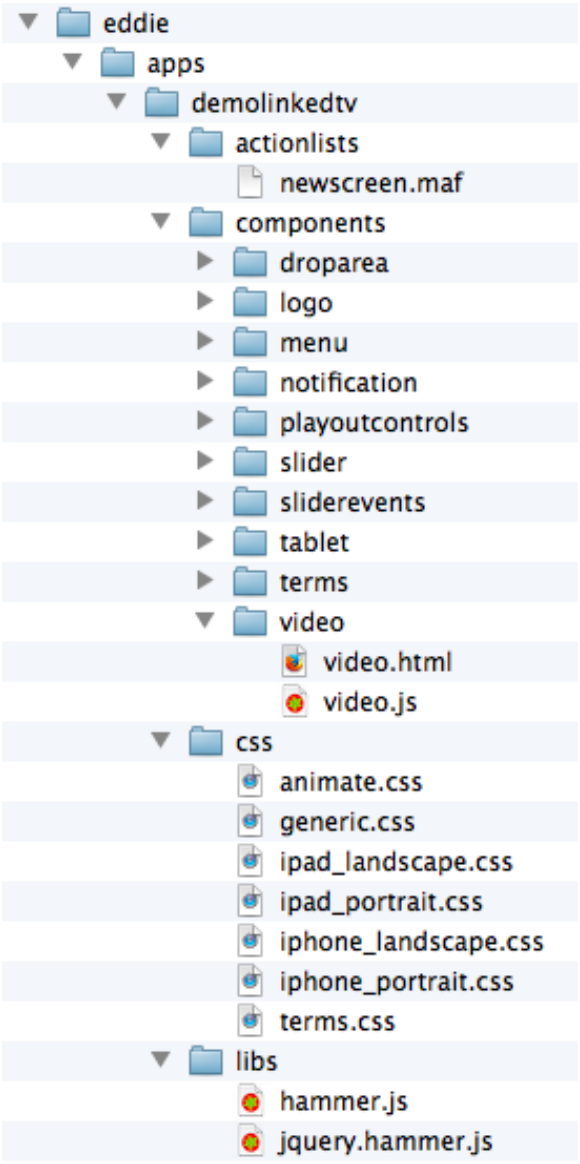

Figure 8: Structure of Eddie for the *demolinkedtv* application

2.3 HbbTV support

The SMT is aimed to offer full HTML5 support but the reality is that even on desktop browsers the HTML5 specifications are more an aim point for browser makers than a reality. There is currently no browser available with a full HTML5 implementation. As a result our design philosophy is to implement the lowest and most available solution that is supported on as many devices as possible. Using this and allowing developers to tune their applications dynamically per screen type we were able to implement support for HbbTV 1.5 devices without many problems. In the support libraries we added several elements for use on HbbTV boxes like direct support for QR codes, the CE-HTML methods for video playout/control and direct support for IR-remotes.

With the help of RBB[6] and IRT[7] we tested several HbbTV compatible devices including brands like Philips, Sony, Samsung and Inverto (VolksBox). We expect this work to continue and that the platform will continue to gain features that allow us to use low-power and cheap HbbTV implementations found in the market today. We feel the structure of this toolkit that tries to do most of the work inside the server and keep the clients as basic as possible to be a good fit and real world tests seem to confirm this.

2.4 Handling personalization

As already seen in section 2.2.2.2 there is an available GAIN API for sending player events or context events from external devices (e.g. Kinect) for personalization usage. These events will lead to personalization of the annotations and enrichments returned from the LinkedTV Platform where the personalization takes place. The SMT hides most of this process and only requires a user to be logged in. For a better understanding of this personalization process Figure 9 shows how both the annotations and enrichments are being retrieved from the platform and the events are being sent to the platform. Both personalized and non personalized annotations and enrichments are retrieved by using a single call. Only when a user is logged in the results will differ based on the personalization. Due to this layer of abstraction the SMT does not need to deal with any changes in the personalization, it simply relies on the LinkedTV Platform for giving the correct annotations and enrichments for usage in the current user session of the player.

[6] http://www.rbb-online.de/

[7] http://www.irt.de

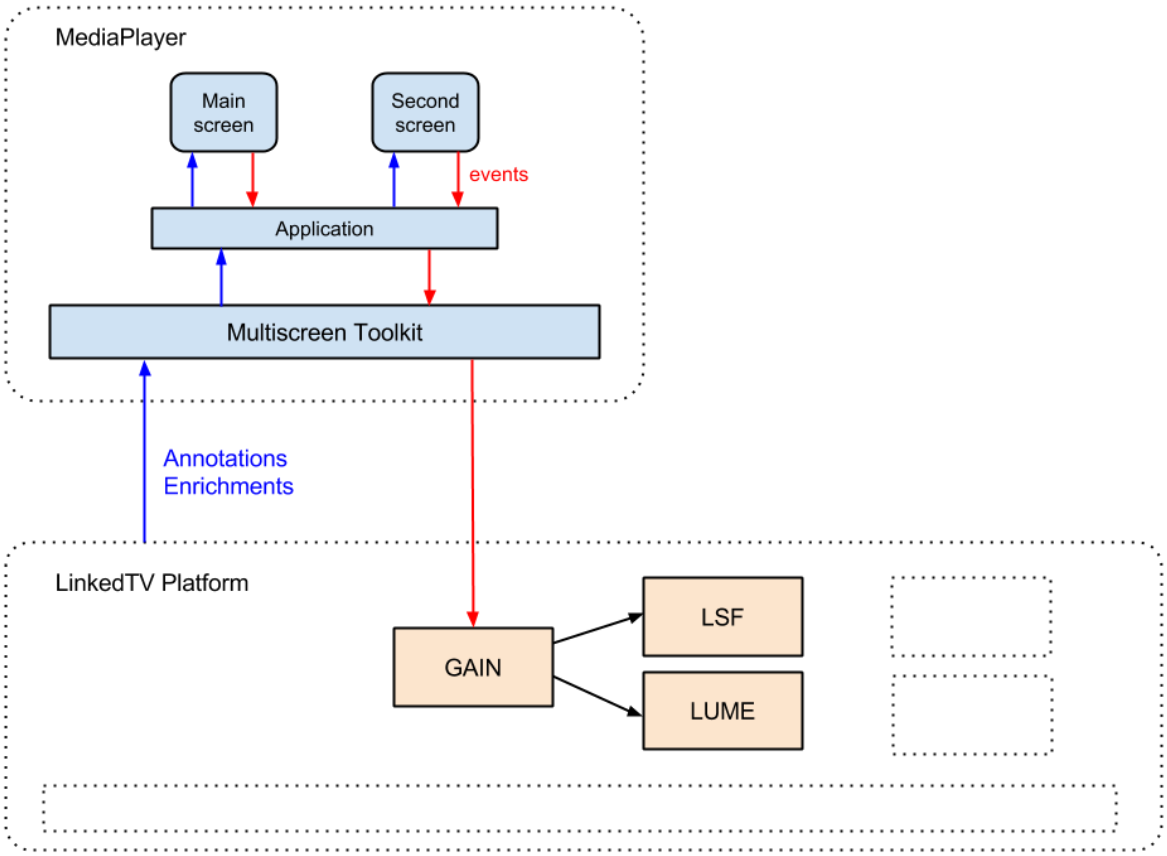

Figure 9: Personalization process workflow

3 Open source

This chapter outlines the work done in making the Springfield framework and the SMT open source. Currently all the source code from the Springfield framework and the SMT are available under a GPL version 3 license[8] on GitHub. This makes it possible for external developers to run the Springfield framework and develop with the help of the SMT. However making software open source is a continuous process that evolves over time including creating and maintaining a community of stakeholders. In the next sections we describe the outline for the next phases of our planned efforts even if they fall outside the scope of the LinkedTV project.

3.1 First phase

The first phase is the phase that took place during the last year as part of this LinkedTV deliverable D5.5. During this phase all the required code has been made publicly available and has been checked to be in a state that a trained developer can start to use the code with minimal help. Together with selected partners from the LinkedTV project we started workshops as an introduction to developing using the SMT, supplied them with APIs and documentation. Also we started with community tools. We have chosen GitHub[9] for this as they offer free tools for hosting open source software and provide community features like issue tracking, wiki pages and code statistics. Finally a website for marketing both the Springfield Framework and the SMT has been designed and put online at http://noterik.github.io/ as can also be seen in Figure 10.

[8] http://www.gnu.org/copyleft/gpl.html

[9] https://github.com/

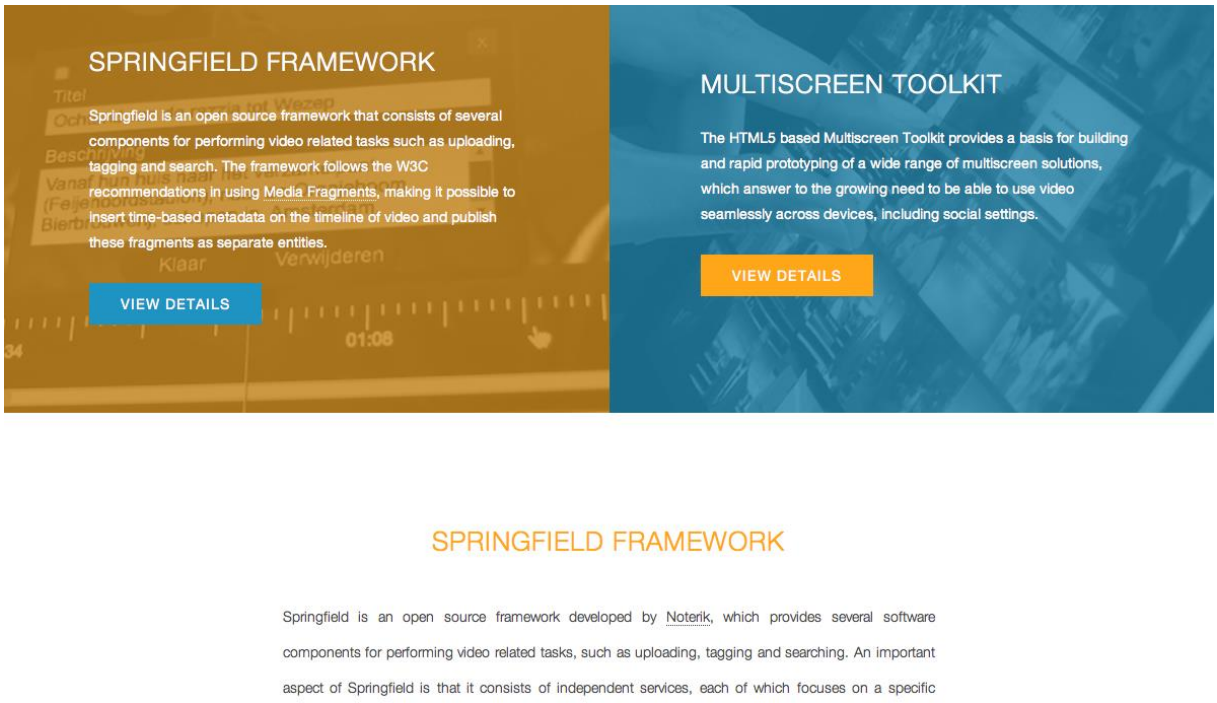

Figure 10: Open source website for both the Springfield Framework and SMT

3.2 Second phase

The second phase is planned to take place in the next 3 to 6 months but will partly take place outside the scope of the LinkedTV project. During this period we plan to have meetings with stakeholders from the open source community to get feedback on the current state of the project and get advice how to act as both a private company and a maintaining body of an open source community. Also we want to make guidelines with respect to the responsibility of maintaining the projects, and see how we can attract external developers that can contribute to both the Springfield framework and the SMT.

We also plan to have a second cluster running outside the Noterik scope for deployment, which can preferably be supported by a LinkedTV partner in a cost sharing model at an open cloud provider that allows easy and quick scaling of resources. Further promotion is planned by means of public presentations about the Springfield framework and the SMT that should inform people, projects and companies about our efforts and get attention and support.

3.3 Third phase

The third phase is planned to take place in the next 12 to 24 months and should be considered as a long-term vision. For this reason it is not yet a clear defined path, but outlines the direction that is currently envisioned. As the planning indicates this will and cannot be part of the LinkedTV project.

Most important for us is to increase the usage of both the Springfield framework and the SMT and to have a defined release life cycle. Together with the community we want to research

options for shared project and client acquisition. If successful we eventually will look into durability and stability of the community by setting up a foundation. This will be done under supervision of Daniel Ockeloen who has in the past worked on MMBase[10], an open source CMS with a strong focus on multimedia. MMBase is currently looked after by the MMBase foundation[11].

[10] http://www.mmbase.org

[11] http://www.mmbase.org/?page=19662

4 LinkedTV Front-end Applications

This chapter gives a short overview of the two applications that have been developed as a LinkedTV front-end with the help of the SMT. Both applications have been developed with a different focus with respect to the content and the intended audience based on the different scenarios and personas.

4.1 HbbTV news application

Noterik and Condat worked together on the realization of an HbbTV compatible news application that fits with the RBB news episodes. This application can be used with only an HbbTV as it is shown in Figure 11. It allows the user to request the available chapters in the current episode, these are listed on the bottom of the screen. The user can browse through the chapters and see the current chapter highlighted. The user can also navigate to another chapter and start the video playout from this chapter. If the user wants to know more the user can request information cards about a certain chapter that give some more information about the annotations available in the chapter. When the information cards are shown on the screen the video is paused so the user can focus on reading and browsing through the information cards.

Figure 11: HbbTV news application showing an information card

A second screen is not any longer required for this application, however support for this has not been dropped. Connecting a second screen will give the same information as available in the HbbTV.

4.2 Culture application

The Culture application that has been developed by CWI using the SMT focuses on the usage of the *Tussen Kunst & Kitsch* programme from a tablet. The application is initiated from a tablet where the user first selects an episode from *Tussen Kunst & Kitsch*. The video starts playing on the tablet accompanied with the chapters as can be seen in Figure 12.

Figure 12: LinkedTV culture application running on a tablet

The application has support for multiple screen and can send the video to a connected television. When a user wants to know some more background information about the current chapter he can simply request this information by pressing the (i) button and more information appears as can be seen in Figure 13.

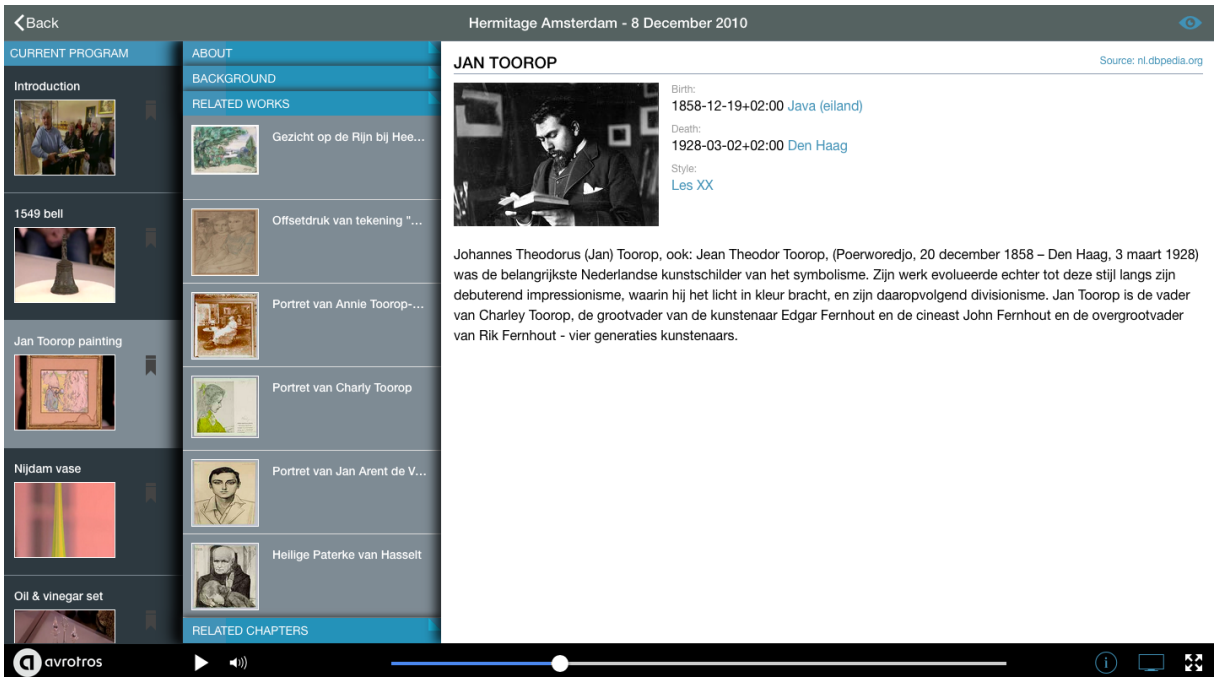

Figure 13: LinkedTV culture application showing extra information on a tablet

5 European project collaborations

Noterik has made its core technology available to the LinkedTV platform as is usual with European projects. Work on the Springfield framework has been started in 2007 and was ongoing development since then. The Springfield framework is used by several other projects and by more than a dozen commercial clients. As outlined in the DoW [3] the software developed during the LinkedTV project is released as open source but we also feel it is important to explain a bit more about other European projects that benefit from the work performed during the LinkedTV project. We will briefly explain how they relate to each other and what cross development has been done and how the Springfield framework and the SMT will develop further within these projects.

5.1 EUscreenXL

EUscreenXL[12] is the 4th installment of a serie of projects started with 'Birth of TV'[13], 'VideoActive'[14] and 'EUscreen'[15]. The project brings together 30 partners from 21 European member states. The network was established in 2006 while the EUscreen Foundation was founded in 2013 to ensure a long-term sustainability of the network. Within the EUscreenXL project Noterik is responsible for backend services (using the Springfield framework) and building the portal with a wide range of publish formats, including apps using the SMT. The start of EUscreenXL came about one year after the start of LinkedTV project, therefore earlier work on the Springfield framework for VideoActive and EUscreen served as a base for LinkedTV. While the Springfield framework was optimized and the SMT was developed during the LinkedTV project we can reuse these technologies offering multiscreen solutions for EUscreenXL. The new portal of EUscreenXL is completely being developed as a multiscreen application consisting of several reusable components that with the help of the application manager (section 2.2.1) can be deployed and updated while under development. Over the next year it is expected that other technical partners from EUscreenXL will become involved in our open source development and research the possibilities for making a connection with the EUscreen foundation. A new service developed during the EUscreenXL project is Portia. Portia extends the SMT with tools that allow for quick portal development. Further Rafael (section 2.1.7) will be extended in such a way to offer DRM and CDN features as requested by some partners in the EUscreenXL project.

[12] http://www.euscreenxl.eu

[13] http://www.birth-of-tv.org

[14] http://www.videoactive.eu

[15] http://euscreen.eu

5.2 Europeana Space

The Europeana Space[16] project aims to reuse Europeana[17] content within the creative industry. Noterik as one of the SME partners facilitates the TV pilot to provide an open environment for the development of applications and services based on digital cultural content. Within the TV pilot Noterik will provide the Springfield framework and will host a series of workshops in preparation of a large hackathon that is planned for 2015. During the workshops and hackathons the SMT will act as a base for the developers to create multiscreen applications around content from the partners LUCE[18], Sound & Vision and RBB. The workshops will test and improve the readiness of the SMT for multi partner and multi discipline rapid prototyping. After the hackathon the created applications will be judged and a limited number will be invited to be co-developed so these can be shared in Europeana Labs[19]. Project partners will continue to develop prototypes around local social video and around broadcast video applications.

[16] http://www.europeana-space.eu/
[17] http://www.europeana.eu/
[18] http://www.cinecitta.com/
[19] http://labs.europeana.eu/

6 References

[1] Springfield Framework, http://www.noterik.nl/products/webtv_framework/

[2] GAIN API documentation, http://inbeat.eu/gain/docs/rest

[3] LinkedTV Description of Work – Part B, p81