D3.4.2 – Middleware and web server for accessing Europeana

This deliverable is software.

co-funded by the European Union

The project is co-funded by the European Union, through the ***eContentplus*** programme

http://ec.europa.eu/econtentplus

Österreichische Nationalbibliothek

EuropeanaConnect is coordinated by the Austrian National Library

ECP-2008-DILI-528001

EuropeanaConnect

D3.4.2 – Middleware and web server for accessing Europeana

Deliverable number/name	*D 3.4.2*
Dissemination level	*Public*
Delivery date	*2010-07-31*
Status	*1.0*
Author(s)	*Dennis Heinen, Tobias Hesselmann, OFFIS*

e*Content*plus

This project is funded under the *e*Content*plus* programme,
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.

Österreichische Nationalbibliothek

EuropeanaConnect is coordinated by the Austrian National Library

Summary

This deliverable summarizes the implementation of a middleware and web server for mobile access to Europeana as part of work package 3.4. We start with the presentation of the approach used to derive the user requirements for this deliverable. We then describe the human-centred design process, which is used for the development of the Europeana mobile client. After the discussion of concepts and designs for mobile device recognition, we present the related implementation and describe its integration into the Europeana portal. We continue with designs for mobile optimized presentations. Based on that, we will then present the actual implementation of the mobile user interfaces, followed by the implementation details. We also give an outlook for the future by suggesting further improvements to ensure the developments of task 3.4 can keep up with the technological developments in the next years.

Table of Contents

1 Introduction

The overall goal of EuropeanaConnect Task 3.4 is to make the rich cultural content of Europeana available to a broad spectrum of users in mobile scenarios. With the development of mobile access channels for Europeana, we enable users to access the material inside the Europeana database and benefit from the cultural content inside Europeana using their mobile clients when the use of stationary PCs is either impossible or unwanted. For reading convenience, we will refer to the Europeana mobile client application as *eMobile* in the following.

1.1 Motivation

The Europeana portal offers a state-of-the-art web interface for stationary desktop PCs. Users can access the contents of Europeana using their web browsers and explore the rich cultural works using an intuitive user interface.

Nevertheless, due to the fact that more and more users are accessing the web on the go with their mobile devices, it is necessary to optimize traditional websites for the use of mobile devices. This statement is backed by a recent study of (Gartner Inc., 2010), stating that by 2013, mobile phones will overtake PCs as the most common Web access device worldwide.

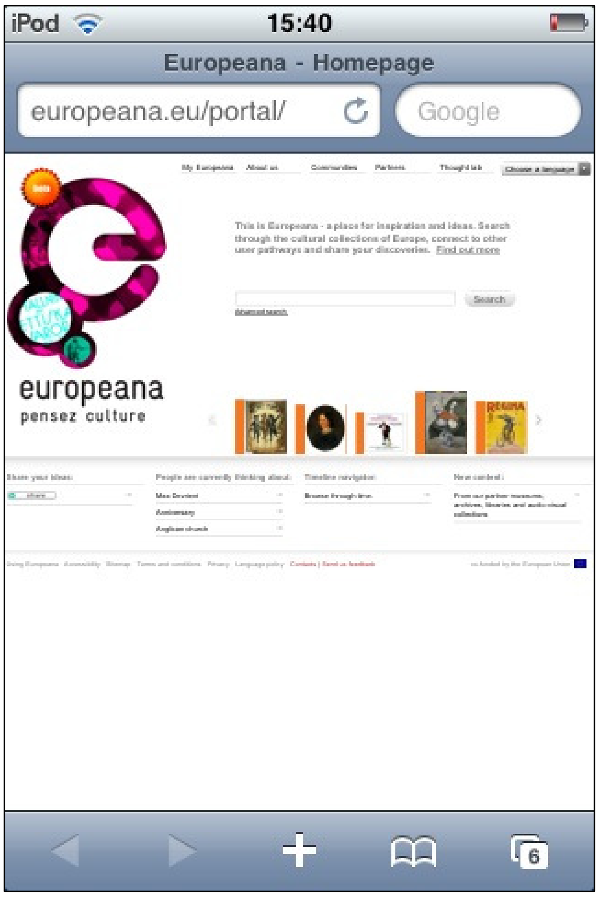


Figure 1. Europeana portal on an iPhone

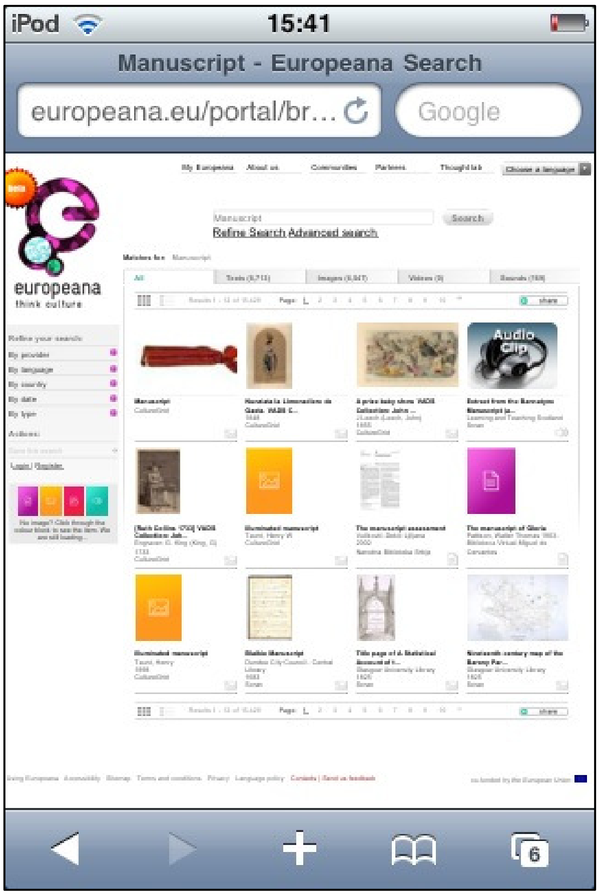


Figure 2. Search results on an iPhone

As the study points out, mobile users need to be treated differently than Desktop PC users for a number of reasons: Typically, the user experience on a mobile device is negatively influenced by a number of factors, including the limited capability of mobile devices when it comes to screen size and the missing support for state-of-the-art technologies, such as JavaScript or AJAX, in mobile browsers. For example, when viewing the Europeana Portal on a mobile device, the available screen size is not efficiently used, as the ordinary web site is naturally optimized for access by standard Desktop browsers (see Figure 1). Also, the navigation between elements (e.g. language selection and search box) is impractical, requiring a lot of zooming and scrolling on the mobile device (Figure 2).

With eMobile, we provide a service tailored to the needs of users in mobile environments, taking into account the specific capabilities of mobile browsers and the screen size of mobile devices.

The remainder of this document is structured as follows: In chapter 2, we will briefly describe our approach for the developments of Task 3.4 and explain the used process model. In chapter 3, we will sum up the results from our requirements analysis in subtask 3.4.1 and outline the identified use cases for this deliverable. Chapter 4 contains the actual design and implementation documentation. We conclude with a summary and present the next steps in the development of the mobile client in chapter 5.

2 Process Model

In this chapter we describe the approach used to define the requirements for the development of *eMobile,* the mobile access client for Europeana.

Human-Centred Design process

The design of an interactive system, in this case a mobile web application, is no trivial task. To ensure the development of a highly usable system that is efficient, effective and satisfying, which are the three main criteria for usability as defined in ISO 9241-11 (ISO, 1998), the application design needs to follow a defined process model. The document at hand is the result of the application of the HCD process, as specified in ISO 13407 (ISO, 1999). It is particularly well suited for the design of interactive systems, as it incorporates user feedback in all stages of development, which can be considered one of the most crucial aspects in software engineering.

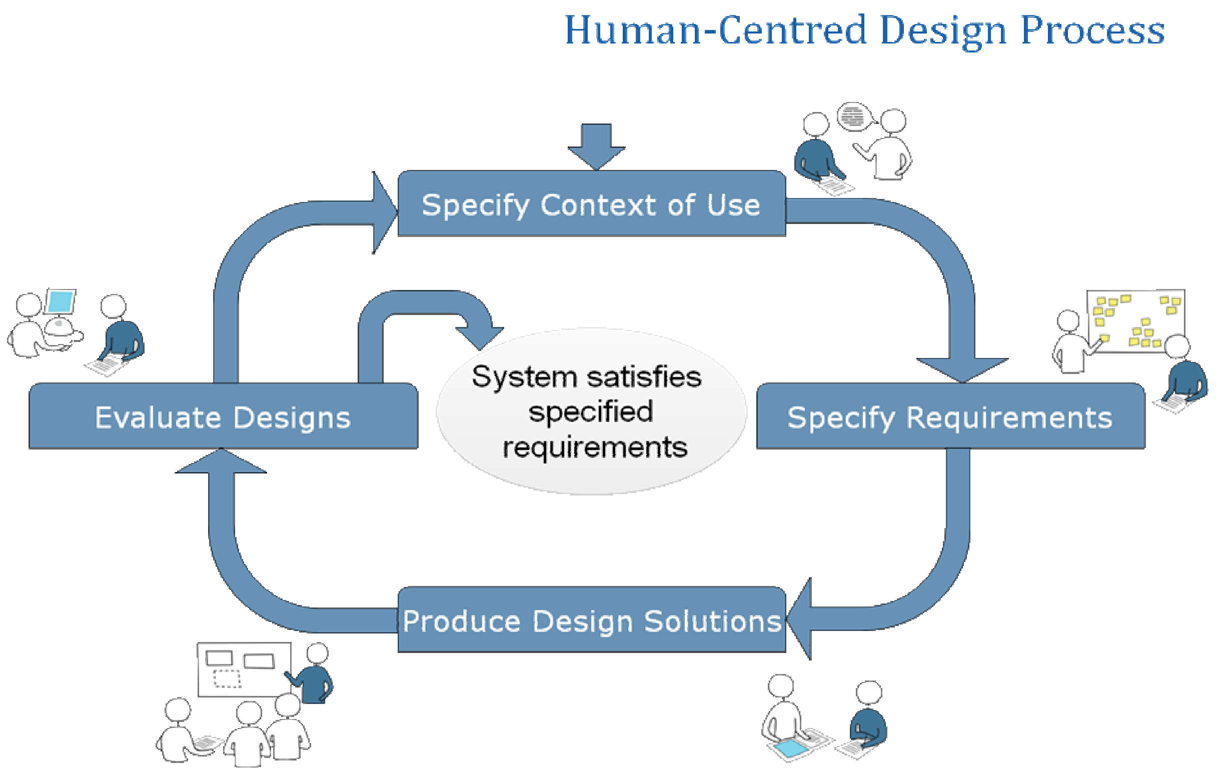


Figure 3. Human-Centred Design Process

The HCD process is illustrated in Figure 3. It consists of four steps:

1. *Specify Context of Use.* In this step, the stakeholders of the product are identified and the user environment is described. This step gives developers a "big picture" of the product and its users.
2. *Specify requirements.* The specification of requirements is the most essential step to create highly usable products. In this step, the goals of the product's users will be gathered and described in a standardized format.
3. *Produce design solutions.* Based on the first steps, the development of the actual software version is carried out.
4. *Evaluate design.* A crucial step to measure the usability of a product and to improve the product usability-wise is to perform evaluations on the product, which are conducted in this step.

The process is then repeated until the developed system satisfies the formerly specified requirements. In Deliverable 3.4.1, we have specified the Context of Use and the Requirements for a mobile client for Europeana (OFFIS Institute for Information Technology, 2009). The actual design and implementation documentation of the mobile client, which builds on the requirements defined before, is split into two documents: In this document, Deliverable 3.4.2, we describe the basic functionality of the mobile client, including the Middleware and Web Server functionality, which provides basic search functions to mobile users. In Deliverable 3.4.3., we report on functions for rich mobile devices and smartphones, including location-aware searching of Europeana content. Deliverable 3.4.4 will conclude with an evaluation of the requirements themselves and the developed mobile clients, according to the last section of the HCD process.

3 Requirements Definition

This chapter sums up the results from our requirements analysis in task 3.4.1 and outlines the identified use cases for task 3.4.2.

3.1 Summary of Mobile Operating Systems / Mobile Browsers Analysis

Since smartphones are becoming more and more important when it comes to mobile internet access, we will focus on the development of a mobile access channel for Europeana that meets the needs of mobile users with that device class. Because the development of the mobile market is constantly changing, it is important to focus on the layout engines used in modern mobile browsers. By adhering to established web standards (HTML/CSS) it is possible to create a user experience that is independent of the used operating system and mobile browser. To accomplish this goal, the different hardware capabilities of devices need to be considered, e.g. by adapting the size of a result list to the available screen area and resolution.

3.2 Results from User Survey

The survey – conducted with senior staff members from our project partner, the Royal Library of Denmark, with experience in mobile access or human factors – gives an interesting picture about the image of Europeana and the features users demand of a mobile client for Europeana. The participants gave some comments concerning the improvement of some aspects of the current Europeana web portal, particularly when it comes to quality of the search results, purpose of the web site and speed. Thus, for a mobile client, an easy operation was explicitly demanded by some of the participants. The added value of a mobile client was not obvious to the users at first sight, although they were able to identify numerous scenarios for a mobile client, including research, educational and fun use of Europeana in mobile context. It seems their opinions may be biased by the look and feel of the Europeana portal, which is not optimized for mobile devices at the current time. The most important features identified were the performance of the mobile application and the support of the different capabilities of mobile devices, e. g. different resolutions of the used displays. The users were particularly interested in social networks, good search results and, first of all, a simple, fast and interactive mobile interface to Europeana.

3.3 Functional Requirements / Use Cases

In Deliverable 3.4.1, we have formalized the functional requirements in use cases, which are summarized in the following (see (OFFIS Institute for Information Technology, 2009) for a more detailed version).

UC 1.1 Simple search
The system shall allow the user to do a simple keyword search

UC 2.1 Visualization of Search Results in text-only List
The system shall allow the user to visualize results in a text-only perspective

UC 2.2 Visualization of Search Results in gallery list
The system shall allow the user to visualize results in an image-only gallery perspective

UC 2.3 Visualization of Search Results in Mixed list
The system shall allow the user to visualize results in a mixed image/text perspective

UC 3 Visualize details of an item in the search results
The system should allow the user visualize details of a selected item in the search results

3.4 Non-functional Requirements

In contrast to functional requirements, non-functional requirements do not make a statement about the behaviour of the system, but about its quality. They are an essential part of the requirements definition, especially in the context of larger projects as Europeana, in which thousands of users are potentially working with the system each day.

The following requirements are applicable to Subtask 3.4.2. A full list and a description of each requirement can be found in D3.4.1 as part of the user requirements definition.

Usability

The usability of the mobile client is a critical aspect that demands special attention. According to DIN EN ISO 9241-11, usability is defined as the “extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.”

According to the user centred design process, we will evaluate the usability of the system in a user study as part of Task 3.4.4, after the release of RHINE in M15 and give recommendations about future improvements of the system.

Security

One of the most important non-functional requirements is security. Therefore, the system shall not store any personal information about a certain user that cannot be changed by the user him/herself. It shall not allow unauthorized individuals or programs access to any communication.

Scalability

Scalability is a critical issue for all developments in the EuropeanaConnect project. Europeana will become a central service for all Europeans and is therefore likely to experience heavy traffic from day to day. This also holds true for the mobile web client developed in this task, it thus needs to be made sure that eMobile will be scalable according to the increasing popularity of Europeana.

Extensibility

Extensibility is a quality of design that takes possible future advances into consideration and attempts to accommodate them. The system shall therefore be able to allow the addition of features without influencing existing system functions. The usage of SPRING as a framework, as recommended by the Europeana Office, assists in keeping the system flexible and extendable.

Maintainability

The code developed in this task needs to be maintained by external institutions, i. e. the Europeana Office, after the project. To ensure this, we support the development architecture proposed by the Europeana Office, concerning development platforms and tools, as well as programming language and frameworks as good as possible.

Testability

To ensure a proper testability of the code, we have developed unit tests for all critical parts of the software. Unit tests can be executed automatically to confirm the correct operation of the code after changing parts of the system. We have furthermore tested the operation of the system manually to ensure proper operation from a user centric point of view.

Platform Compatibility

The system should support a wide range of mobile devices and offer optimized versions of the Europeana portal that suit the various mobile devices used. This is also backed by the results from the user survey. All users pointed out that the support of different mobile devices would be one of the key features of mobile access to Europeana. We have incorporated multiple mechanisms to support this requirement, which are described later in this document.

Performance

Since mobile internet connections still have to deal with lower bandwidths than regular internet connections, one of the main focus points for performance considerations should be the amount of transferred data. It shall only transfer data relevant to his/her query that makes sense in a mobile environment. E.g. a high resolution image should only be loaded if the user wants to view and actively selects it.

3.5 Constraint Requirements

There are also constraints in the form of technical demands that are made by the Europeana office. These are described in detail in the Guidelines for the use of EuropeanaLabs (Siebinga, et al., 2009). We address and refer to them (e.g. testing and external libraries) during the course of this document.

In the following, we will describe the actual design and implementation of eMobile, which adheres to the requirements presented in this chapter.

4 Design and Implementation

The main requested feature in the user survey conducted in D3.4.1 – Catalogue of user requirements is to support heterogeneous mobile devices, including a presentation of search results and images that fits specific capabilities of the used mobile device. Similar suggestions are presented by the industry (see Gartner Inc., 2010), making the support of different devices a main objective of Task 3.4.2.

In order to realize this feature, we have developed an *Adaptive Web Client* (see Figure 4), which

1. automatically identifies whether the Europeana web site is accessed by an ordinary desktop PC or a mobile device;
2. detects the capabilities and features of the accessing device and
3. adapts the look and feel of the web site to the device in use.

Figure 4 shows an illustration of the eMobile system architecture. At first the system detects whether the actual request to the Europeana web site is made from a stationary desktop PC or from a mobile device. If a desktop PC is used, the user is redirected to the standard Europeana Portal. If a mobile device is used instead, the system carries out a device identification, detecting the capabilities of the mobile devices and the **Mobile Web Browser** used. In order to overcome the heterogeneity and limitations of mobile devices, we have implemented an **Adaptive Web Client**, which acts as a middleware between the Mobile Device and the Europeana Database. It encapsulates queries made by the mobile device, forwards them to the Europeana database and collects the query results, which are then processed and adapted to the respective mobile device. We have split the Adaptive Web Client into two parts: Basic and advanced services. The basic search functionality covers the formulation of queries, the browsing of search results and the displaying of detailed information for items inside the database, which are described in this chapter. The advanced services, which include location aware search functionality, are described in detail in Deliverable 3.4.3.

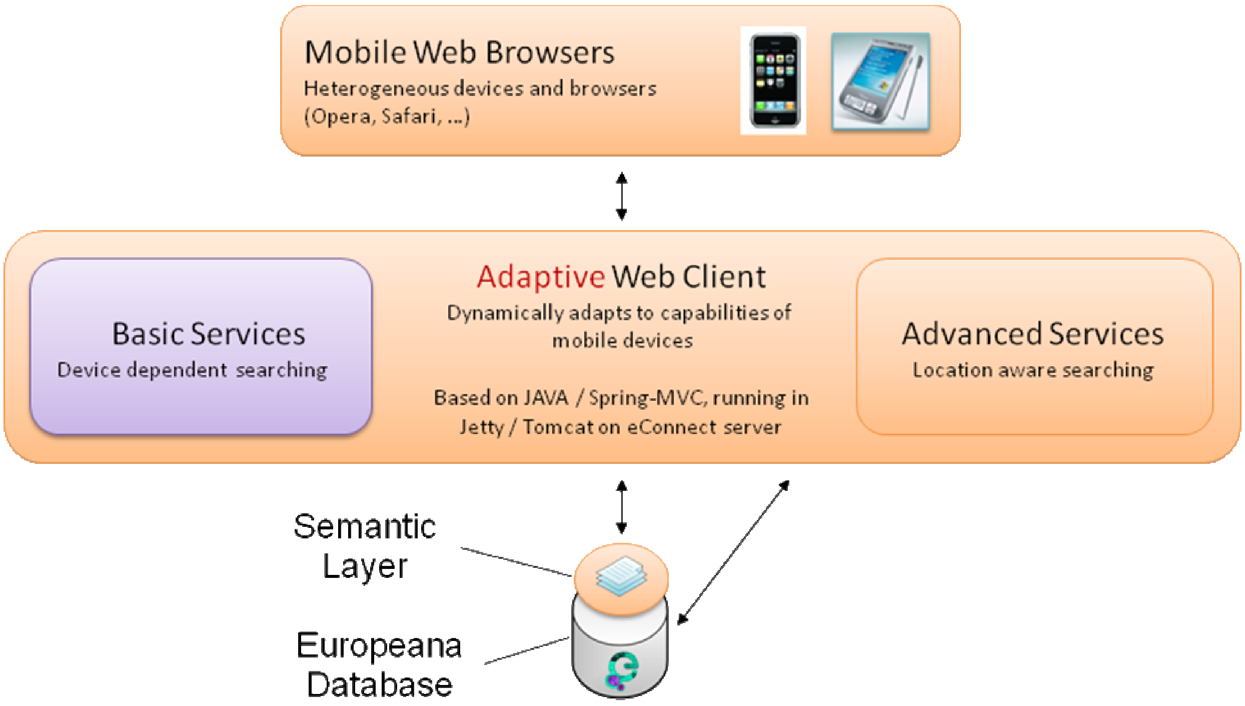

Mobile Web Browsers
Heterogeneous devices and browsers (Opera, Safari, ...)
Adaptive Web Client
Dynamically adapts to capabilities of mobile devices
Based on JAVA / Spring-MVC, running in Jetty / Tomcat on eConnect server
Basic Services
Device dependent searching
Advanced Services
Location aware searching
Semantic Layer
Europeana Database

Figure 4. eMobile System architecture

In the following, we will explain the necessary concepts for the first two steps of the system, the identification of the accessing device and the detection of its capabilities.

4.1 Concepts for mobile device recognition

Basically, there are two ways of recognizing devices accessing a web site. In the *client-based* approach, JavaScript functions are executed on the accessing device. Most browsers implement the `navigator` object that can be used to query different device properties (e.g. `appCodeName`, the internal browser name or `appVersion`, the browser version). The device then registers itself as a stationary or mobile device to the server. This approach has several drawbacks, including the fact that many mobile browsers only offer very limited JavaScript functionality. Furthermore, users may deactivate JavaScript completely, rendering this approach fairly unreliable. Instead, we are using a server-sided technique, which relies on logic implemented on the web server side, making this approach independent from the used browser and device.

Concepts for device recognition

There are several server-sided methods that help in determining the identity of a particular device (Georgieva, 2007) which are outlined in the following.

HTTP Headers

Web browsers and servers use the HTTP protocol to transfer information on the WWW. Each HTTP request (see Figure 5) contains various field-value pairs in its header. The most useful information for device recognition is the User-Agent Header. It identifies the client device and contains information about the browser, operating system and sometimes hardware information.

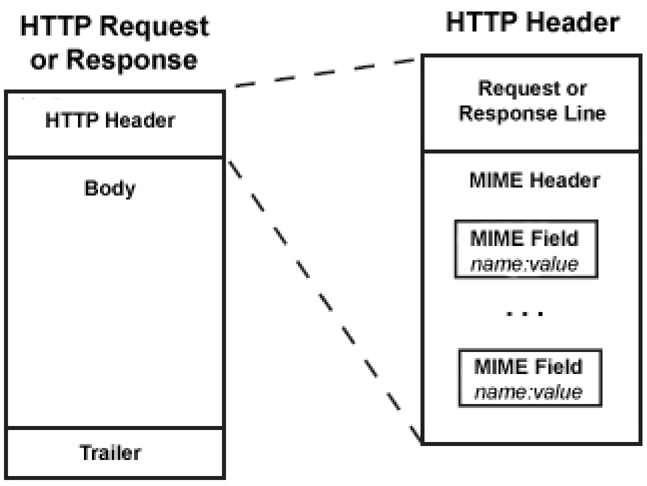


Figure 5. HTTP Request/Response and Header Structure (Apache Software Foundation, 2010)

Excerpt from an HTTP-Request header of a mobile device:

```
Connection          Keep-Alive
                                   […]
Accept:             text/xml,application/xml,application/xhtml+xml,text/ht
                    ml;q=0.9,image/png,*/*;q=0.5
Accept-Charset      iso-8859-1,utf-8;q=0.7,*;q=0.7
Accept-Encoding     gzip, deflate, x-gzip, identity; q=0.9
Accept-Language     de;q=1.0,nl;q=0.5,en;q=0.5,fr;q=0.5,it;q=0.5,tr;q=0.5
User-Agent          Mozilla/5.0 (SymbianOS/9.4; Series60/5.0 Nokia5800d-
                    1/50.0.005; Profile/MIDP-2.1 Configuration/CLDC-1.1 )
                    AppleWebKit/525 (KHTML, like Gecko) Version/3.0
                    BrowserNG/7.2.3
                                   […]
x-wap-profile       http://nds1.nds.nokia.com/uaprof/Nokia5800d-1r100-
                    2G.xml
```

User Agent Profile

A mobile device usually sends a device-specific value within the header of an HTTP request, containing the URL to its User Agent Profile (UAProf). A UAProf file describes the capabilities of a mobile handset, including Vendor, Model, Screensize, Multimedia Capabilities, Character Set support, etc. However, there are several reasons why device recognition should not be entirely based on UAProfs:

- Device manufacturers provide UAProfs voluntarily, and not all devices have a UAProf.
- Some UAProfs are unavailable, due to outdated links, company shutdowns, etc.
- Quality between UAProfs varies, since there is no industry-wide quality standard for data within each field of a profile.

Excerpt from UAProf for a Nokia mobile device:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF>
<!-- Start of uaprofile -->
   <rdf:Description rdf:ID="Profile">
<!-- Start of Hardware component -->
      <prf:component>
         <rdf:Description rdf:ID="HardwarePlatform">
         [...]
            <prf:ImageCapable>Yes</prf:ImageCapable>
            <prf:InputCharSet>
               <rdf:Bag>
                  <rdf:li>ISO-8859-1</rdf:li>
                  <rdf:li>ISO-10646-UCS-2</rdf:li>
                  <rdf:li>US-ASCII</rdf:li>
                  <rdf:li>UTF-8</rdf:li>
               </rdf:Bag>
            </prf:InputCharSet>
            <prf:Keyboard>PhoneKeyPad</prf:Keyboard>
            <prf:Model>5800 XpressMusic</prf:Model>
            <prf:NumberOfSoftKeys>2</prf:NumberOfSoftKeys>
            [...]
            <prf:ScreenSize>360x640</prf:ScreenSize>
            <prf:ScreenSizeChar>17x13</prf:ScreenSizeChar>
            <prf:TextInputCapable>Yes</prf:TextInputCapable>
            <prf:Vendor>Nokia</prf:Vendor>
         </rdf:Description>
<!-- End of Hardware component -->
      </prf:component>
[...]
<!-- Start of BrowserUA component -->
      <prf:component>
         <rdf:Description rdf:ID="BrowserUA">
            <prf:BrowserName>Nokia</prf:BrowserName>
            <prf:BrowserVersion>4.0</prf:BrowserVersion>
            <prf:FramesCapable>Yes</prf:FramesCapable>
            <prf:HtmlVersion>4.1</prf:HtmlVersion>
            <prf:JavaAppletEnabled>No</prf:JavaAppletEnabled>
            <prf:JavaScriptEnabled>Yes</prf:JavaScriptEnabled>
            <prf:JavaScriptVersion>1.2</prf:JavaScriptVersion>
            <prf:PreferenceForFrames>No</prf:PreferenceForFrames>
            <prf:TablesCapable>Yes</prf:TablesCapable>
            <prf:XhtmlVersion>2.0</prf:XhtmlVersion>
            <prf:XhtmlModules>
               <rdf:Bag>
                  <rdf:li>XHTML1-struct</rdf:li>
                  <rdf:li>xhtml-basic10</rdf:li>
               </rdf:Bag>
            </prf:XhtmlModules>
         </rdf:Description>
<!-- End of BrowserUA component -->
      </prf:component>
[...]
   </rdf:Description>
<!-- End of Uaprofile-->
</rdf:RDF>
```

Wireless Universal Resource File (WURFL)

The Wireless Universal Resource File (WURFL) is an open source community-driven project that provides a configuration file containing information about the features of mobile devices available on the market. The maintainer’s main goal for this file is to support maximum information for wireless devices. UAProf device profiles are one of the sources of device capability information for WURFL, which maps the UAProfile schema to its own with many other items and boolean fields divided into capability groups like device markup, multimedia capabilities and more. Devices are identified by the User-Agent header in web request and mapped to a profile in the resource file.

Excerpt from WURFL capabilities for a Nokia mobile device:

```
Group: product_info
device_os: Symbian OS                    model_name: 5800 XpressMusic
nokia_series: 60                         device_os_version: 9.4
has_qwerty_keyboard: false               uaprof2:
pointing_method: touchscreen             http://nds1.nds.nokia.com/uaprof/N
nokia_edition: 5                         5800XpressMusicr100-2G.xml
uaprof:                                  is_wireless_device: true
http://nds1.nds.nokia.com/uaprof/Nokia   brand_name: Nokia
5800d-1r100-3G.xml                       release_date: 2008_november
```

```
Group: display
physical_screen_height: 62
columns: 17
dual_orientation: true
physical_screen_width: 53
rows: 13
max_image_width: 360
resolution_height: 640
resolution_width: 360
max_image_height: 640
```

Conclusion

From the methods for mobile device recognition, WURFL turns out to be the most suitable, since it makes efficient use of all methods described above and combines it into a production ready resource that currently contains information of about 13.000 devices.

It also offers a Java API to query information and follows the open-source spirit that also applies to Europeana. There is an easy way to integrate WURFL via Spring as a framework for dependency injection and aspect oriented programming, which is a constraint requirement for the development in EuropeanaConnect. As shown in the examples above, the WURFL-data is structured in a very convenient way that simplifies querying certain capabilities.

4.2 Implementation and integration of device integration

Integration into Europeana framework

A result from the first EuropeanaConnect Developer's meeting (November, 2009, Vienna) was to aim for a tight integration with the original Europeana Portal on the top-most level.

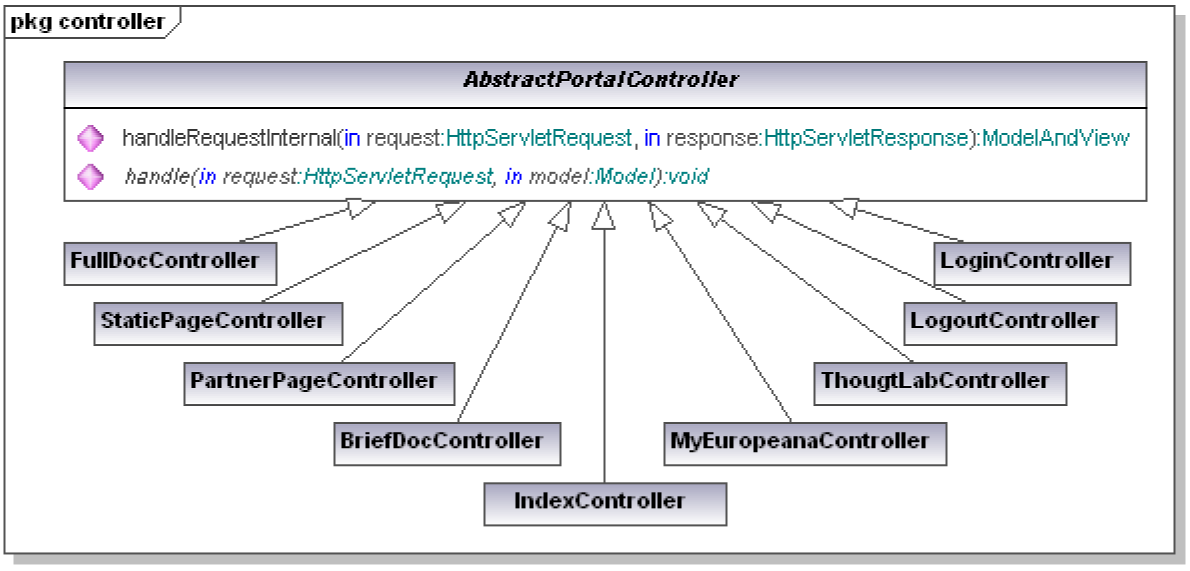


Figure 6. Simplified class diagram of controller package

Our initial approach was the integration of the device in the AbstractPortalController of the Europeana portal (Figure 6). This class serves a base class for all controllers relevant for the mobile client and is called on each request (Siebinga, 2009). When the user connects to Europeana, the device recognition is initiated and returns a path to a template folder which contains the corresponding (mobile) templates. If no mobile device is recognized, this will point to the default template for non-mobile devices.

During the evolvement of the Europeana framework the AbstractPortalController mentioned above became obsolete and was replaced by an even more spring-based approach. As a result of this development we had to rework the integration of our mobile device recognition accordingly.

The new Spring based implementation allows the usage of so-called interceptors, which we decided to use for device recognition. Figure 7 shows a simplified sequence diagram of a typical spring-based web application request.

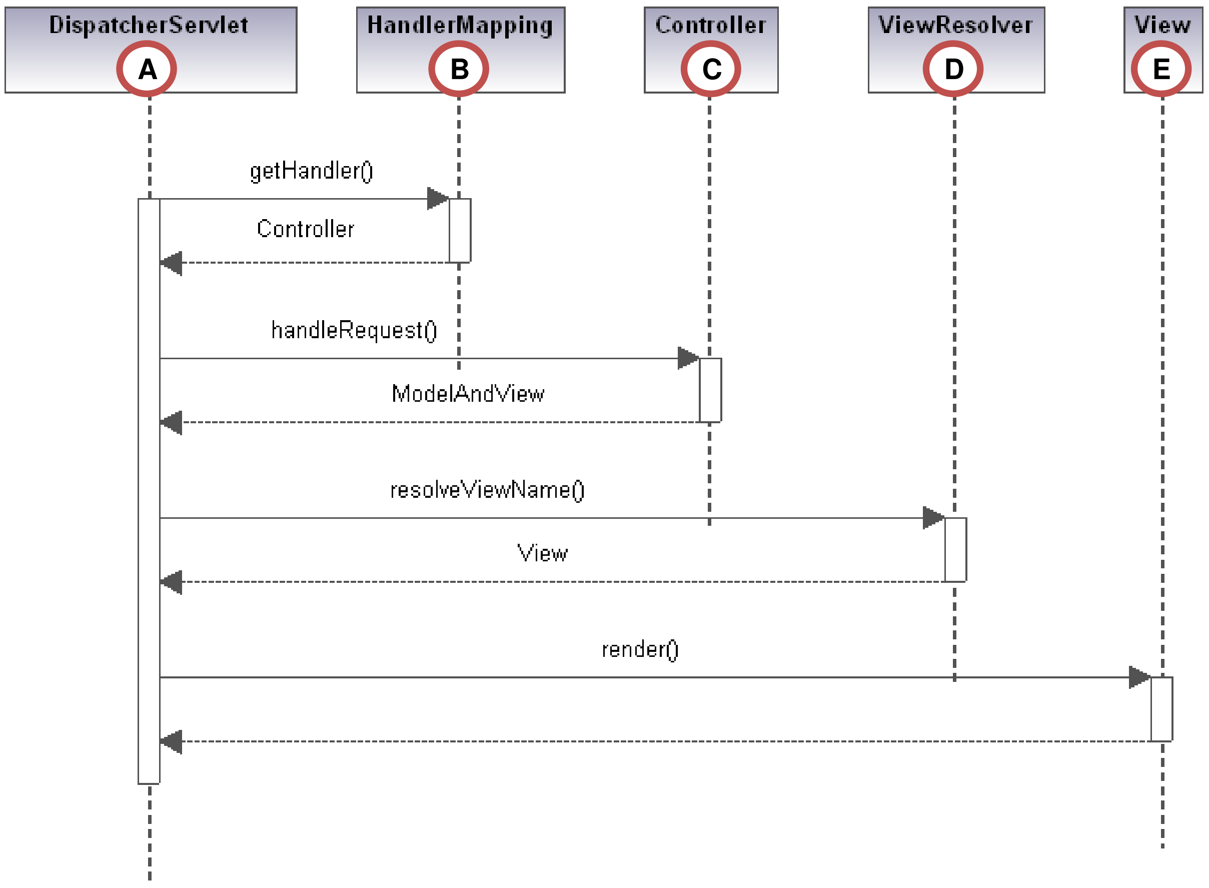


Figure 7. Web application request, simplified sequence diagram (diagram created by Platinum Solutions, 2008)

The DispatcherServlet (A) accepts the request and uses the HandlerMapping (B) to select the configured Handler for the current Request. The returned Controller (C) handles the Request and returns a ModelAndView object that shall used to populate the page that shall be presented to the User. In the next step, the ViewResolver (D) returns the View (E) for this request that finally is rendered and transferred to the User.

A common way to add new features to a web service in Spring's AOP-approach is the use of interceptors. Spring Interceptors have the ability to pre-handle and post-handle web server requests. Figure 8 shows the same sequence diagram with an additional interceptor configured:

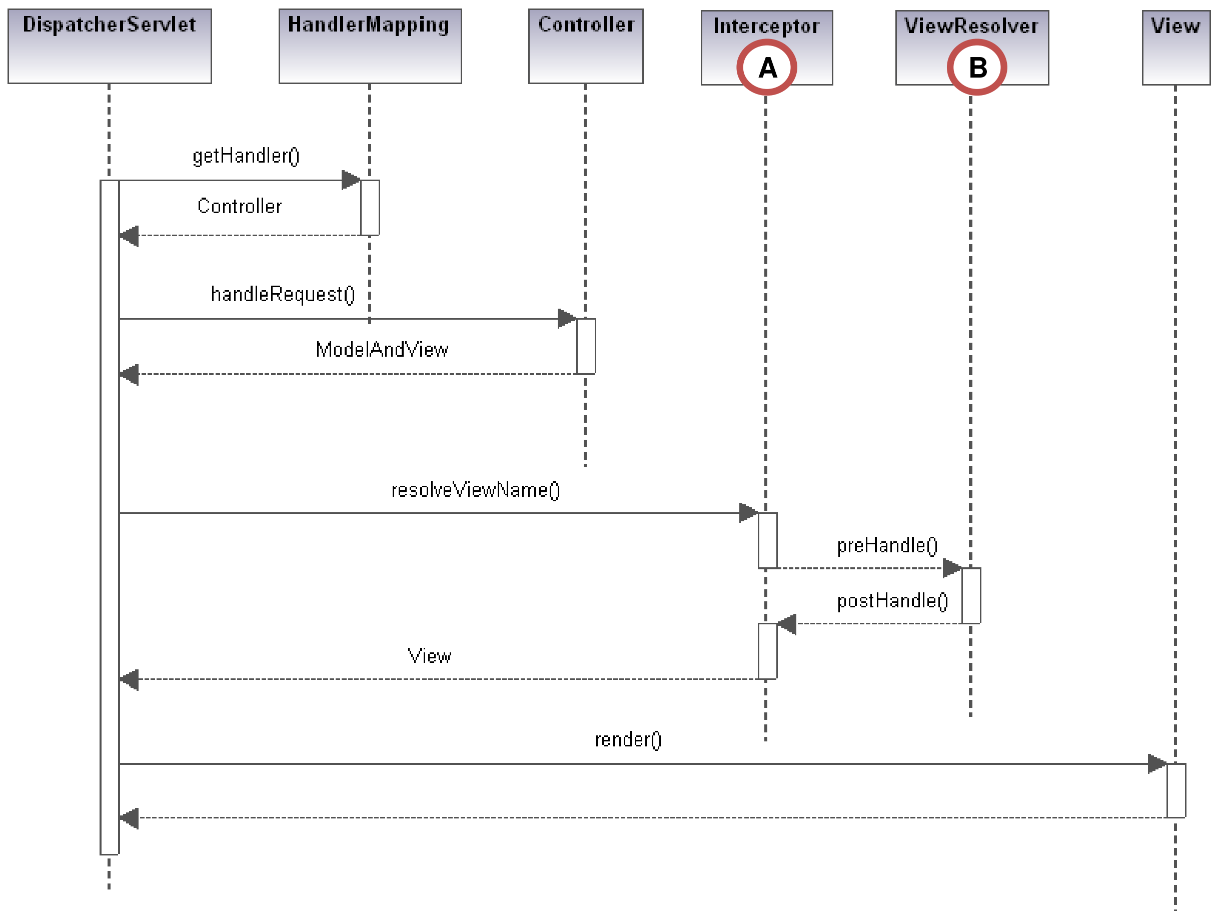


Figure 8. Web application request with interceptor (extended version of diagram created by Platinum Solutions, 2008)

By using an interceptor, it is possible to execute additional code before and after a specific step in the lifecycle of a request. Since it is only a simple configuration via XML, Interceptors can easily be added to a web-service and developed in a decoupled environment. The resulting code may be incorporated into other projects as well.

For mobile device recognition in Europeana we need the ViewResolver (B) to determine the right view name for the page the user requested. After that is done, our Interceptor (A) post-handles the request, performs the device recognition and – if the page was requested from a mobile device – returns the appropriate “mobile” view instead of the regular view.

Identifying desktop browsers

A crucial aspect in mobile device recognition is the correct handling of desktop browsers. If requests are made from a desktop browser, the system needs to ensure that the correct view is presented to the user. By default, WURFL’s API assumes that every request is made by a mobile device. In order to support desktop browsers as well, WURFL supports the usage of so-called “Patch-files”, which is an extension of WURFL that lists all the most common user-agents of web browsers and sets a capability that allows the system to determine that a request was not made by a mobile phone.

During the first tests of WURFL’s device detection it turned out that desktop browsers sometimes were identified wrong as mobile devices, especially when major browser updates were released (e.g. when the latest version number of Firefox 3.5.7 was switched to version 3.6.

http://tech.groups.yahoo.com/group/wmlprogramming/message/32120). Since the system should be able to deal with newly released desktop browser without immediate patching, we decided to integrate a simple desktop detection engine. This engine is a port of an extension to a php-based wurfl-project (Tera-WURFL) to the java programming language that makes it usable within the Europeana framework. The result is depicted in Figure 9.

Implementation

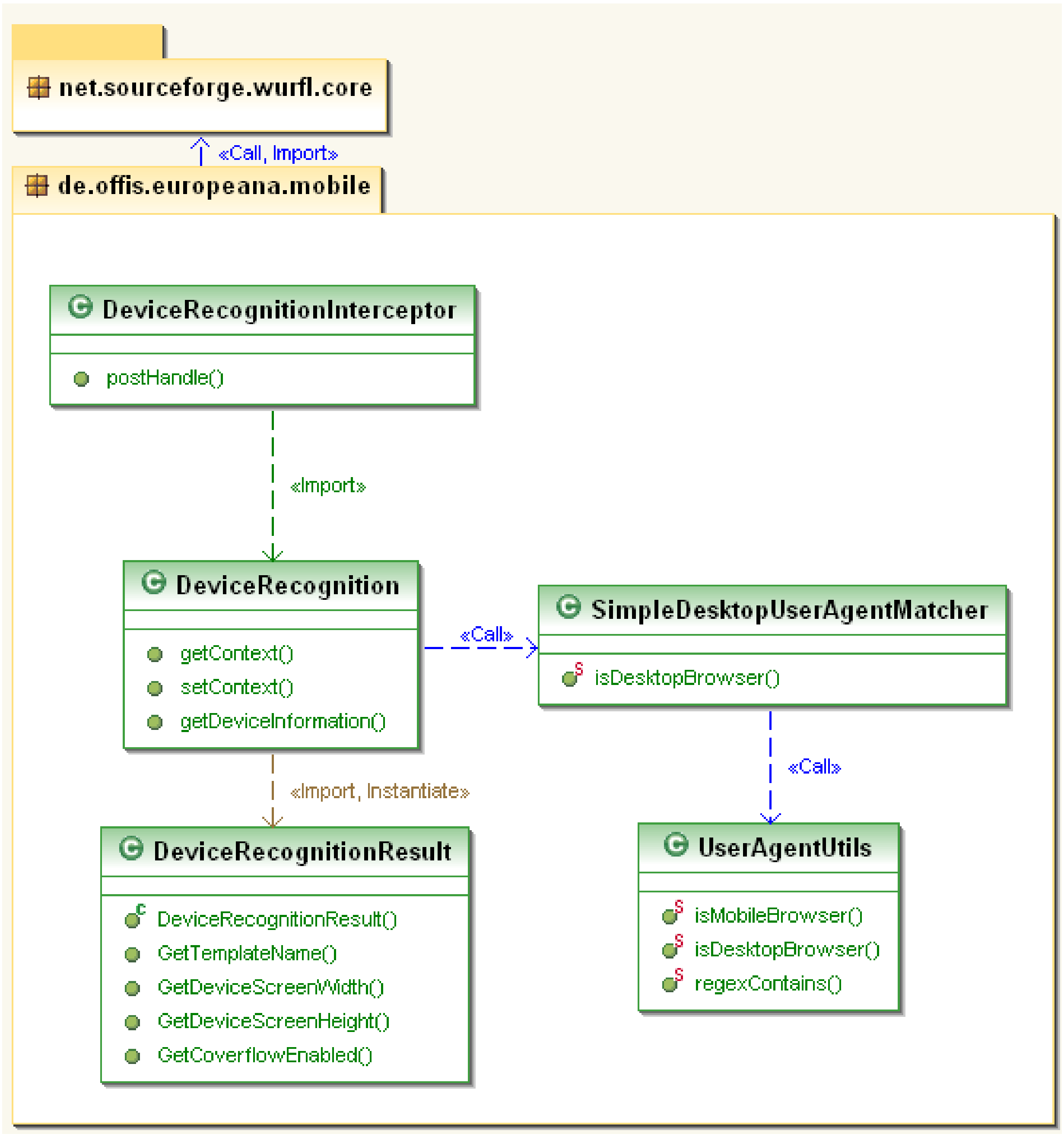


Figure 9. Classes in package de.offis.europeana.mobile

All classes needed for the device recognition are located in a package called `de.offis.europeana.mobile`. This package makes intensive use of the WURFL API for java that is located in the `net.sourceforge.wurfl` package.

`DeviceRecognition` encapsulates the WURFL-based device recognition and allows the detection of a device and the appropriate template selection by calling `getDeviceInformation()`.

An instance of `DeviceRecognitionResult` wraps the result of a device detection to an object that can be used as return-value.

`SimpleDesktopUserAgentMatcher` and `UserAgentUtils` are used to enhance WURFL's desktop browser detection. As described above, these classes try to identify desktop browsers with the help of the UserAgent and some helper methods.

`DeviceRecognitionInterceptor` is meant to be used as an interceptor. It uses an instance of `DeviceRecognition` to modify the `ViewName` to a template that is best suited for a mobile device's capabilities.

Configuration

In order to make use of Wurlf's Java-Api, a new dependency in Europeana's portal.`pom.xml` maven configuration file needs to be added:

```
        <dependency>
            <groupId>net.sourceforge.wurfl</groupId>
            <artifactId>wurfl</artifactId>
            <version>1.0.1</version>
        </dependency>
```

This will download the desired artefact and provide the required libraries for build and deployment tasks.

To add the interceptor, a modification of the `Dispatcher` Configuration is necessary: This configuration is usually found in `/webapp/WEB-INF/dispatcher-servlet.xml`. At first, the interceptor-bean has to be defined:

```
<bean id="deviceRecognitionInterceptor"
     class="de.offis.europeana.mobile.DeviceRecognitionInterceptor" />
```

Afterwards, the bean can be added to the list of already configured interceptors:

```
<bean
    class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotation
    HandlerMapping">
  <property name="interceptors">
    <list>
      <ref bean="localeChangeInterceptor"/>
      <ref bean="formatInterceptor"/>
      <ref bean="configInterceptor"/>
      <ref bean="deviceRecognitionInterceptor"/>
    </list>
  </property>
</bean>
```

Finally, the mobile application context (see `\portal-full\src\main\resources\`) that initializes the WURFL service needs to be imported:

```
      <import resource="classpath:/mobile-application-context.xml"/>
```

The resulting interception of the request is depicted in Figure 31 (Appendix).

Description of the device recognition process

At first, the `DeviceRecognitionInterceptor`, derived from `org.springframework.web.servlet.handler.HandlerInterceptorAdapter`, calls the `postHandle()` method of its base class. The request is handled in 11 steps afterwards:

1. Identify current View name and store it in the variable `currentViewName`.
2. Check if the view name starts with “redirect”, which indicates a view that does not need to be modified to a mobile template. The Europeana framework introduced this mechanism to handle outgoing links and log where users left the Europeana portal.
3. If the view fulfils the criteria above, the actual device recognition is performed and the result is stored in a variable.

   With the help of WURFL’s Java API, the device detection itself is very straightforward:

   At first, the system tries to identify the requesting browser by its User-Agent. If the `SimpleDesktopDetection` Engine is activated, the User-Agent is compared to a list of simple keywords and some regular expressions that match desktop browsers. If the request is not made by a desktop browser, the detection is handed over to wurfl’s detection logic. It returns a `Device`-object that is then used to access the device’s capabilities. By using a list of model names, mobile browsers, operating systems and version information, the system determines a display mode for the list of treasures (see chapter 4.3.1 for explanation of treasures). In addition to that, a folder that contains mobile templates is chosen. Afterwards, this information - together with some basic capabilities (e.g. screen size) - is used to create an instance of the `DeviceRecognitionResult` class that is then used as return-value.
4. The new template name is requested from the `DeviceRecognitionResult` object.
5. The new template name is set as view name in the interceptor.
6. The information about the requesting device’s screen width is accessed.
7. The information about the device’s screen width is added to the `ModelAndView` object.
8. The information about the requesting device’s screen height is accessed.
9. The information about the device’s screen height is added to the `ModelAndView` object.
10. There are two ways to display the list of treasures on the front page. One of these has been picked according to the device’s capabilities. The Interceptor now requests this information.
11. The information about the way the treasures shall be displayed is added to the `ModelAndView` object.

The Interception has now modified the `ModelAndView` object that is returned for the current request and is complete.

4.3 Mobile-optimized presentation

One of the first choices to be made when designing mobile-optimized websites is the decision for a markup-language, being a constraint for what is technically feasible.

When dealing with transmission protocols and markup-languages for mobile devices, one is typically set to one of two options: WAP with WML as markup language, or HTTP with HTML/CSS as markup languages.

WAP with WML

With the release of version 1.1 of the WAP standard in 1999, mobile phone vendors started to add WAP-Browsers to the majority of their devices, while content providers started to offer mobile services like news headlines or sports results. Since data transmission via GPRS or HSCSD was not common, users of WAP-Browsers had to deal with low bandwidths and therefore only limited functionality with these mobile services.

HTTP with HTML/CSS

With the development of efficient layout engines, larger screen resolutions and higher data transfer rates in mobile networks, HTTP with HTML/CSS has taken the place of WAP for mobile internet portals.

It is most likely that the majority of devices used to access EuropeanaConnect will be capable of displaying websites written in HTML via HTTP, allowing the user to access multimedia data like images, movie and sound-clips. Together with David Haskiya, Product Developer for Europeana, we decided to provide a mobile interface for modern smartphones with HTML-support, since this device class is becoming more and more important when it comes to mobile internet access (Haskiya, 2010). By adhering to established web standards (HTML/CSS) it is possible to create a user experience that is independent of the used operating system and mobile browser.

The scope of mobile development for the Rhine release of Europeana was therefore focused on:

A production ready mobile access to Europeana

- That is platform independent;
- automatic device recognition will result in loading a template for either a small or large mobile screen;
- deprecating gracefully, so more advanced mobile devices can utilise and display JavaScript dependent functionalities which simpler devices will simply not display

To avoid developing and maintaining two different versions for the mobile portal and the ordinary website, we decided that new features that are needed for the ordinary Europeana website as well should be integrated and made available there in the first place. We could then check whether or not changes for the mobile interface need to be done.

4.3.1 Features

Traditional user interfaces cannot be ported to mobile devices without adaption, so the interaction with the device has to be considered carefully. Therefore the usage of the mobile device with a touch or keypad interface is an overall goal for the mobile interface that needs to be considered in page layout decisions and icon designs.

In order to create a mobile client, the set of functionalities of the Europeana website has to be compiled to a reasonable list of features for a mobile scenario. In conjunction with the result of our user requirements analysis, we decided to provide a system with the functionality depicted in the mock-ups and described in the following.

Index page

Being the start point and first contact for most Europeana users, the index (or start) page needs to be both, appealing and easy to use. By offering a well-known layout and key elements, its design should allow the user to understand the provided services at first glance.

- In order to be identifiable as part of the Europeana project, the *logo* is a key object on the index page that should be presented in the upper part. (A)
- The *language selection* feature is a crucial feature for the mobile interface as well as for the desktop interface and should be easy to use and presented in way that reduces the number of clicks or interactions for non-English users. It should therefore be shown when the site is loaded and should not require the user to scroll it into view. (B)
    - The auto selection of the interface language based on the user's device settings was discussed, but descoped for the mobile interface. This feature should be integrated in the Europeana framework and implemented for the desktop client first. It will then most likely work for the mobile interface as well without much additional effort.

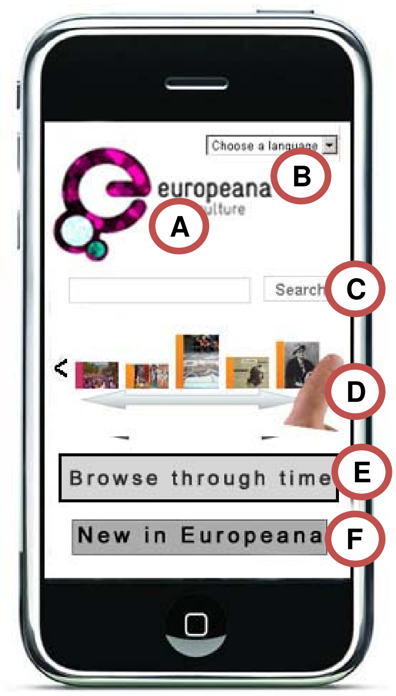


Figure 10. Mock-up of the index page

- *Simple search*: As the most important access to the Europeana database, the search box is a key element on the index page. It should allow the user to perform simple queries. An extended version of the search functionality that allows advanced searches with additional parameters (e.g. operators like "and" or "not") will be available with D3.4.3, the rich mobile client. (C)
    - The automatic suggestion of search keywords after the user has entered the first characters of his query has been discussed, but descoped for the mobile interface development. This feature is under discussion for the desktop interface and will be implemented there first. Depending on the implementation, it is probably easy to activate this feature for the mobile interface as well.
- *Treasures* are famous and visually iconic objects presented as small image (e.g. Mona Lisa, Sunflowers) that directly link to big named works to endorse the Europeana brand. (D)

The following features have been discussed, but descoped for the mobile interface:

- *Timeline browsing*: The current implementation of the timeline was not usable on a mobile device. Due to its beta-state and unfinished reimplementation by the time of code-freeze for eMobile, the timeline browsing feature was disabled. (E)
- *New in Europeana*: A list of recently added items is available on the desktop interface, but has been agreed as not appropriate in its current state for the mobile interface. (F)

The index page covers UC 1.1 (Simple search).

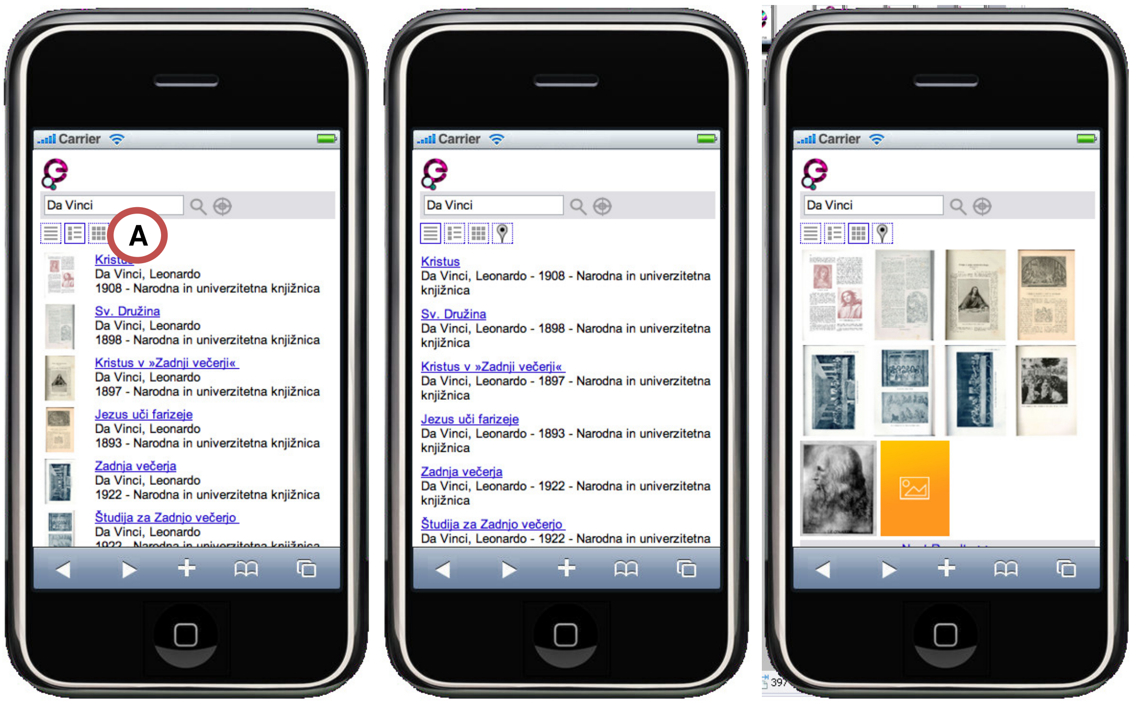
Search results

The search results page is displayed after the user has entered his desired query keywords, and searching by the system is finished. The result page covers three use-cases, the “Visualization of Search Results in text-only List” (UC 2.1), “Visualization of Search Results in gallery list” (UC 2.2) and “Visualization of Search Results in Mixed list” (UC 2.3)

An important requirement for the search result page is the optimal usage of the available screen size. In order to create a convenient user experience, the results should be presented in a way that does not need too much interaction to be viewed, avoiding zooming and scrolling as much as possible.

The Europeana framework returns the results for a search query in chunks of 12 items. Therefore the user may need the possibility to switch between pages of several search results to review all items. Inspired by the visualization of common search engines, we decided to provide three views to display the search results that are shown in Figure 11.

- Result visualization in mixed image/text perspective. This perspective is the default presentation; it combines a thumbnail-sized image preview with the most important item information.
- Result visualization in text-only perspective: This perspective offers a light-weight way to review search results that is useful for lower bandwidth-conditions or older devices.
- Result visualization in image-only gallery perspective: This perspective offers the most graphical representation with a list of images that is displayed in a lightbox way.



Kristus
Da Vinci, Leonardo
1908 - Narodna in univerzitetna knjižnica
Sv. Družina
Da Vinci, Leonardo
1898 - Narodna in univerzitetna knjižnica
Kristus v »Zadnji večerji«
Da Vinci, Leonardo
1897 - Narodna in univerzitetna knjižnica
Jezus uči farizeje
Da Vinci, Leonardo
1893 - Narodna in univerzitetna knjižnica
Zadnja večerja
Da Vinci, Leonardo
1922 - Narodna in univerzitetna knjižnica
Študija za Zadnjo večerjo
Da Vinci, Leonardo
1922 - Narodna in univerzitetna knjižnica

Kristus
Da Vinci, Leonardo - 1908 - Narodna in univerzitetna knjižnica
Sv. Družina
Da Vinci, Leonardo - 1898 - Narodna in univerzitetna knjižnica
Kristus v »Zadnji večerji«
Da Vinci, Leonardo - 1897 - Narodna in univerzitetna knjižnica
Jezus uči farizeje
Da Vinci, Leonardo - 1893 - Narodna in univerzitetna knjižnica
Zadnja večerja
Da Vinci, Leonardo - 1922 - Narodna in univerzitetna knjižnica
Študija za Zadnjo večerjo
Da Vinci, Leonardo - 1922 - Narodna in univerzitetna

Figure 11. Mock-up of search result visualization.

From left to right: Mixed image/text, text-only and image-only perspective

To switch between perspectives the system provides icons that are accessible easily in the upper part of the page. (A)

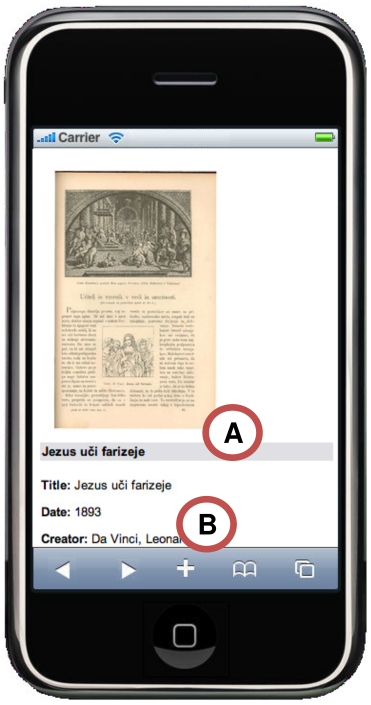
From the search results page, the user can select one of the items or return to the index page to refine his query keywords.

Object presentation

The object presentation page is displayed after the user has selected to view an item from one of the search result perspectives described above. Alternatively, the object presentation is shown after the user has chosen one of the treasure items on the index page.

In the upper part of the page, the title and an image are displayed. (A) Below, a selection of the item's metadata (e.g. creator, year or description) is shown. (B) As in the desktop interface, a link to view the item in its original context is provided, opening a new browser window/tab, if possible.

From the object representation, the user can always return to the search result perspective he came from, as well as back to the index page.



Jezus uči farizeje

Title: Jezus uči farizeje

Date: 1893

Creator: Da Vinci, Leonar

Figure 12. Mock-up of object presentation

Features available on all pages

- Social bookmarking: Each page hosts a button that allows bookmarking and sharing the currently displayed page on the most popular social networks.
- Link to static pages: Each page provides a link to this pages:
  - *About us*: Gives a brief outline on the idea, background and future of Europeana
  - *Contacts*: Offers a contact address as well as a feedback form for comments via email.
  - *Terms and conditions*: Gives information on copyrights, warranty and legal affairs.

This page covers Use Case 3, "Visualize details of an item in the search results".

4.3.2 Device Groups

Although there is a vast variety of browsers on mobile devices, luckily only a few rendering engines are used amongst them. Following the concept behind WURFL, the separation of capabilities into groups, we decided to create clusters of device classes.

- As described in our User Requirements Analysis in the State of art chapter, the WebKit browser engine has the best market share predictions. Being one of the most advanced engines nowadays and used in some of the most popular devices (see Figure 13), we decided to implement an interface that makes use of some of the extensions the WebKit-developers added to their engine. This interface is then shown on devices known to be using this specific engine. Fortunately, two of the most popular device platforms, iPhone and Android are using WebKit, allowing us to create an exciting Europeana user experience for the most common smartphones.

  This set of templates will be referred as "*advanced templates*" or "*advanced interface*" in the following.

- The second set of templates we decided to provide is designed to suit the capabilities of non-WebKit-based browser, offering a simpler layout that only makes use of the most basic browser features.

  This set of templates will be referred as “*basic templates*” or “*basic interface*” in the following.

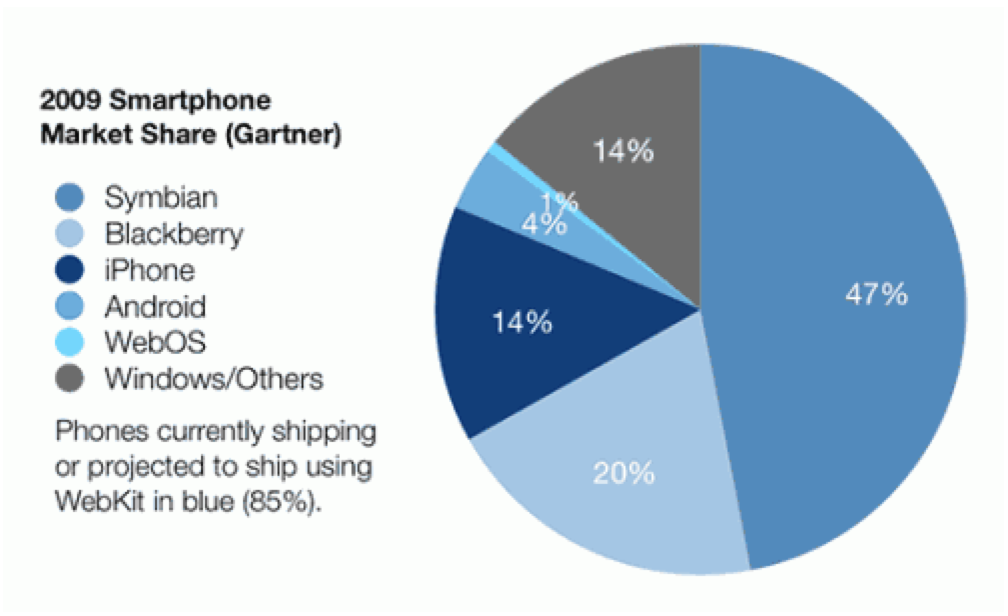


Figure 13. WebKit market share (Gartner, 2009)

Since the Europeana development team has chosen the Freemarker template engine (a JAVA-based engine focusing on the MVC software architecture: http://freemarker.sourceforge.net/) for creating and rendering the web views, the templates that are provided for the mobile interfaces are written using the same technique.

To avoid developing and maintaining two different versions, code from the desktop templates is shared and reused whenever possible. To ease up maintenance, the directory structure and naming conventions for the mobile templates are inspired by the original templates.

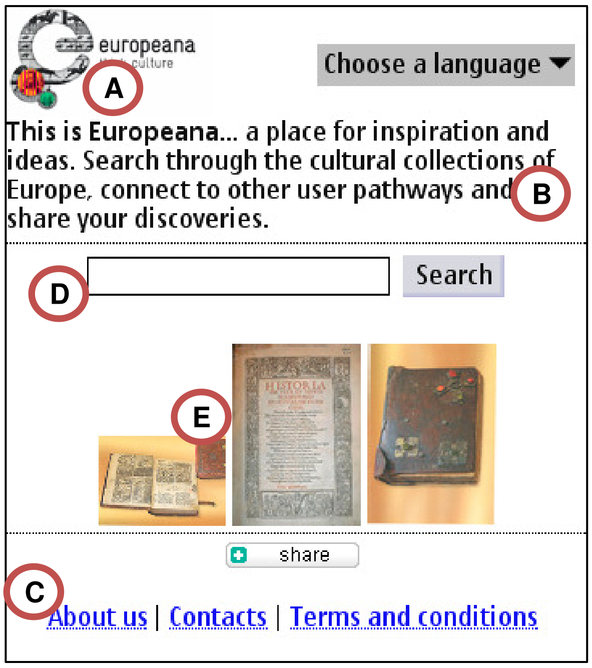
In the following chapters we will present each page of the two sets of templates, compare them and delve into the implementation of some highlights worth mentioning.

4.4 Implementation of the advanced and basic interfaces

This chapter shows the implementation result of the features identified in Chapter 4.3.1. The images below show a screenshot of each page in both templates, the advanced on the right and the basic interface on the left side.

4.4.1 Index page

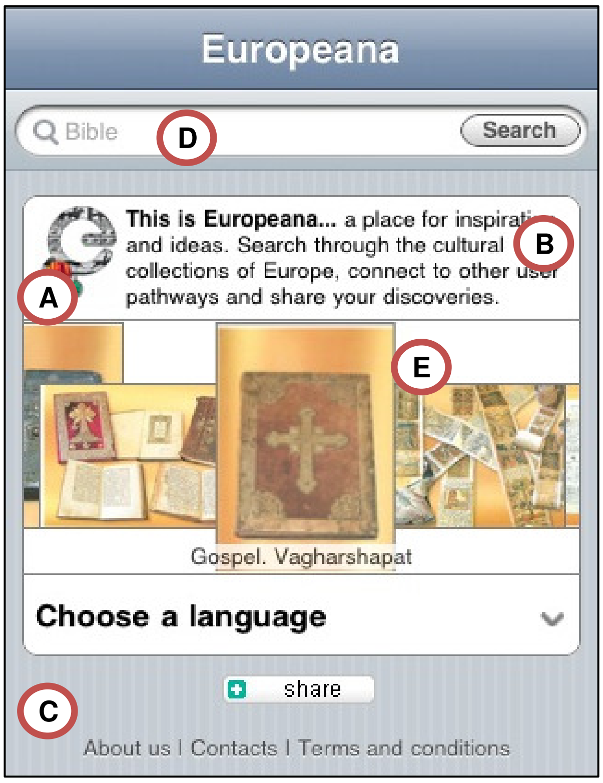
While the advanced interface makes use of some WebKit-browser extensions to mimic the look and feel of a native application (see Chapter 4.5.3 for more information) and provides a dynamic presentation of the treasures, the basic interface offers a more static view of the index page.



This is Europeana... a place for inspiration and ideas. Search through the cultural collections of Europe, connect to other user pathways and share your discoveries.

About us | Contacts | Terms and conditions

Figure 15. Basic Interface: Index page



This is Europeana... a place for inspiration and ideas. Search through the cultural collections of Europe, connect to other user pathways and share your discoveries.

About us | Contacts | Terms and conditions

Figure 14. Advanced Interface: Index page

However, the logo (A) and introductory statement (B) are available in both interfaces to support the Europeana brand and offer a guideline on what to expect from the Europeana project.

The index page usually fits on a single screen and is shown entirely without the need for scrolling or zooming. Only on devices with a very small screen, the footer (containing the links to static pages and share-button) may be out of view (C).

On the advanced template, the search box (D) already shows an example of a search keyword when the page is loaded. It is using a random entry from the list of popular searches in the Europeana database.

Treasures

The list of treasures (E) is presented differently, according to the capabilities of a user's device. While the basic interface displays a static set of images, the advanced interface shows an interactive presentation.

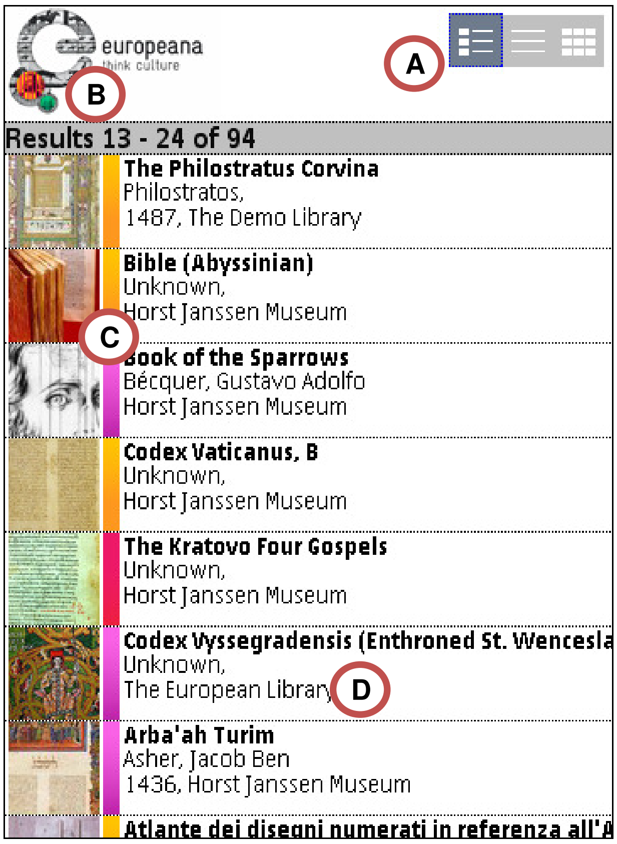
For the basic interface the number of shown images is calculated dynamically to ensure that there is only one "row" of images and no line break that would increase the amount of scrolling, needed to capture the entire page. Each thumbnail-sized treasure image is shown with a width of 80 pixels and since the system "knows" the width of a mobile device, this number can be calculated easily.

For the advanced templates we offer a more interactive presentation of the treasures. Based on a collection of JavaScript and CSS that showcases WebKit's CSS animation effects (http://code.google.com/p/css-vfx/), we were able to create a gallery of images (see Figure 14 (E)) that can be browsed via gestures on touch screen-based devices.

To keep the loading time of the index page to a reasonable time, the system will load a maximum of 12 images from the `carouselitem` database table for the treasures presentation.

4.4.2 Search results

All three perspectives of the search results (Figures 16-21) offer icons in the upper area to switch to a different perspective (A). To return to the index page, the icon (on the advanced template) or Europeana logo (on the basic template) in the upper left can be used. (B)



Results 13 - 24 of 94

The Philostratus Corvina
Philostratos,
1487, The Demo Library

Bible (Abyssinian)
Unknown,
Horst Janssen Museum

Book of the Sparrows
Bécquer, Gustavo Adolfo
Horst Janssen Museum

Codex Vaticanus, B
Unknown,
Horst Janssen Museum

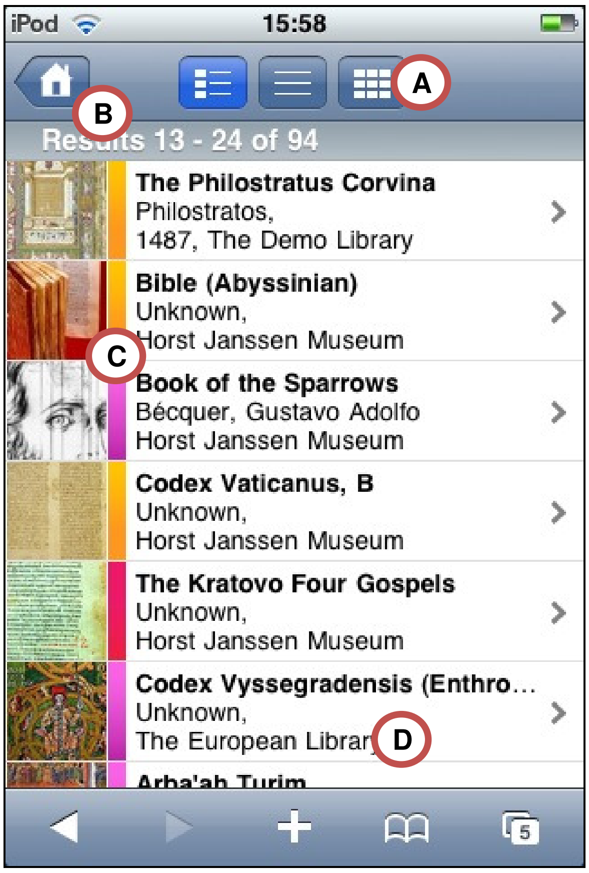
The Kratovo Four Gospels
Unknown,
Horst Janssen Museum

Codex Vyssegradensis (Enthroned St. Wencesla
Unknown,
The European Library

Arba'ah Turim
Asher, Jacob Ben
1436, Horst Janssen Museum

Atlante dei disegni numerati in referenza all'A

Figure 16. Basic Interface: Search results in mixed image/text perspective



Results 13 - 24 of 94

The Philostratus Corvina
Philostratos,
1487, The Demo Library

Bible (Abyssinian)
Unknown,
Horst Janssen Museum

Book of the Sparrows
Bécquer, Gustavo Adolfo
Horst Janssen Museum

Codex Vaticanus, B
Unknown,
Horst Janssen Museum

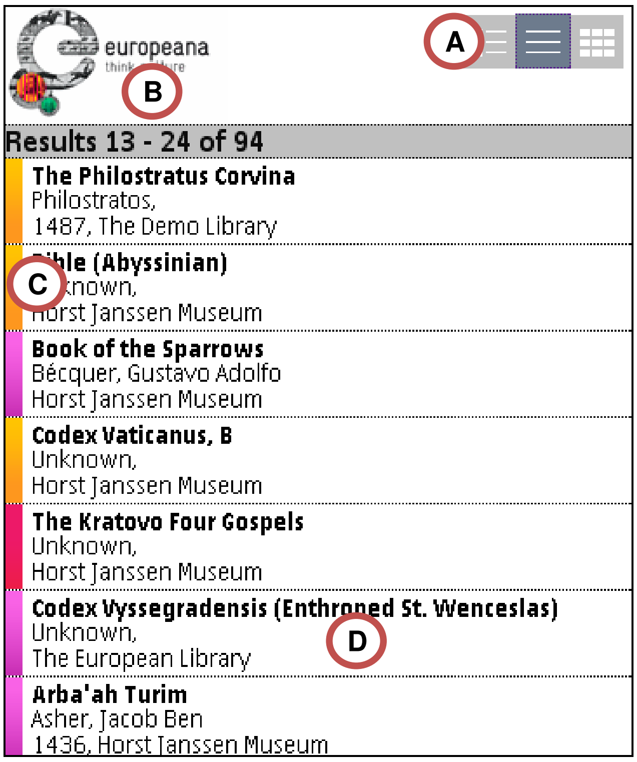
The Kratovo Four Gospels
Unknown,
Horst Janssen Museum

Codex Vyssegradensis (Enthro…
Unknown,
The European Library

Arba'ah Turim

Figure 17. Adv. interface: Search results in mixed image/text perspective

This perspective (Figures 16 and 17) is the default presentation; it combines a thumbnail-sized image preview with the most important item information.



Results 13 - 24 of 94

The Philostratus Corvina
Philostratos,
1487, The Demo Library

Bible (Abyssinian)
Unknown,
Horst Janssen Museum

Book of the Sparrows
Bécquer, Gustavo Adolfo
Horst Janssen Museum

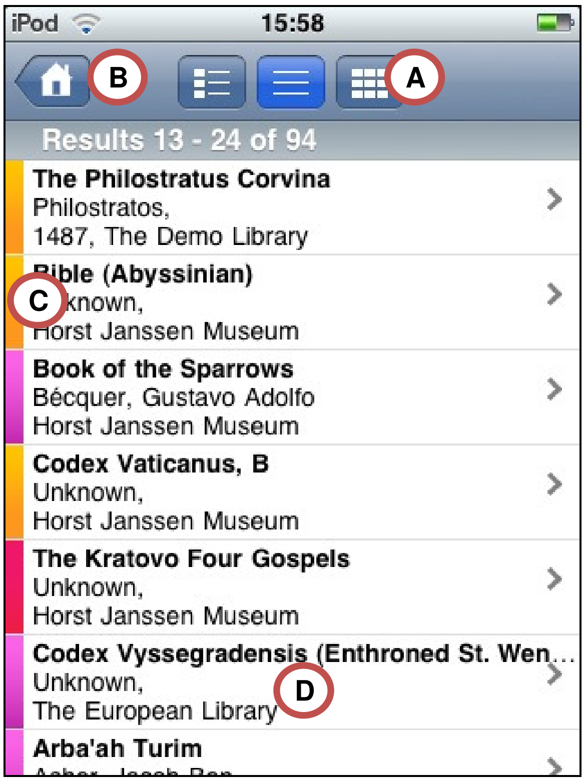
Codex Vaticanus, B
Unknown,
Horst Janssen Museum

The Kratovo Four Gospels
Unknown,
Horst Janssen Museum

Codex Vyssegradensis (Enthroned St. Wenceslas)
Unknown,
The European Library

Arba'ah Turim
Asher, Jacob Ben
1436, Horst Janssen Museum

Figure 18. Basic Interface: Search results in text-only perspective



Results 13 - 24 of 94

The Philostratus Corvina
Philostratos,
1487, The Demo Library

Bible (Abyssinian)
Unknown,
Horst Janssen Museum

Book of the Sparrows
Bécquer, Gustavo Adolfo
Horst Janssen Museum

Codex Vaticanus, B
Unknown,
Horst Janssen Museum

The Kratovo Four Gospels
Unknown,
Horst Janssen Museum

Codex Vyssegradensis (Enthroned St. Wen…
Unknown,
The European Library

Arba'ah Turim

Figure 19. Adv. Interface: Search results in text-only perspective

This perspective (Figures 18 and 19) offers a light-weight way to review search results that is useful for lower bandwidth-conditions or older devices.

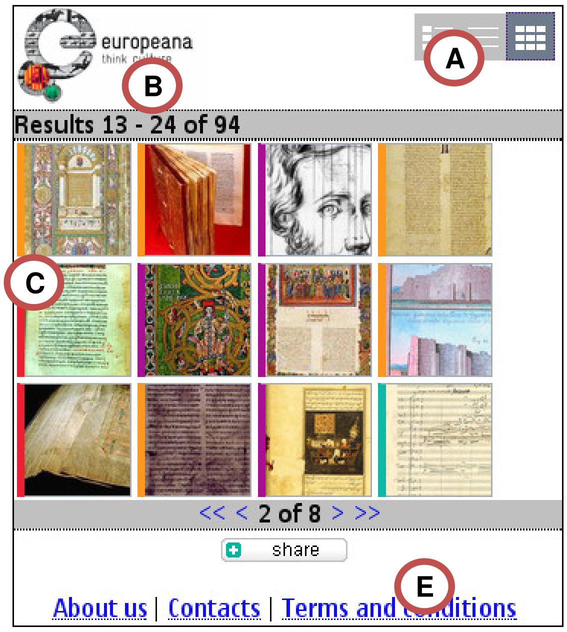


Figure 20. Basic Interface: Search results in image-only gallery perspective

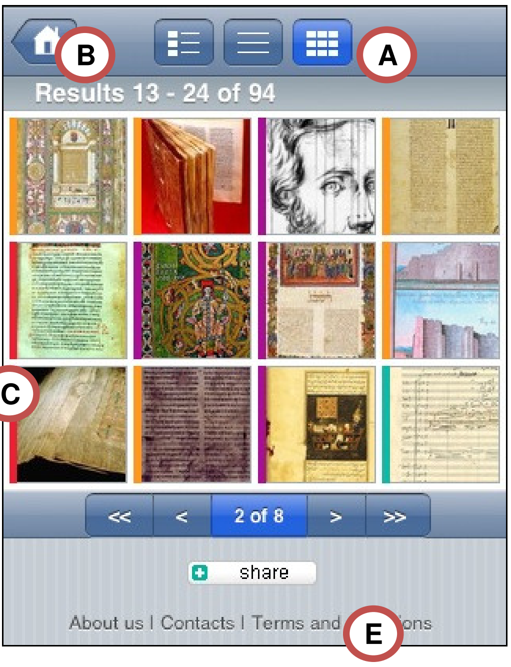


Figure 21. Adv. Interface: Search results in image-only gallery perspective

The last perspective (Figures 20 and 21) offers the most “visual” presentation with a list of images that is displayed in a gallery-like way.

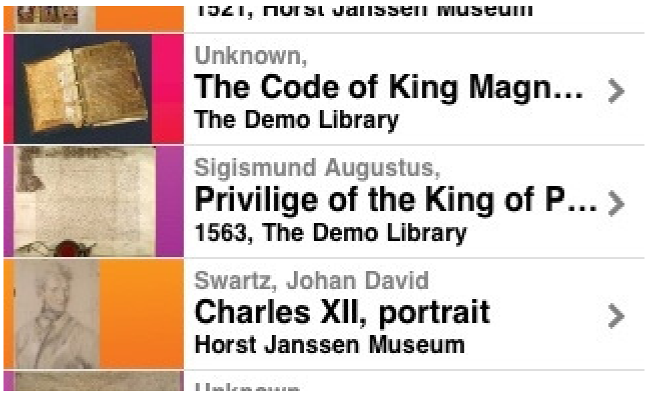
For each result item, the type (image, text, sound or video) is indicated visually (C), either by a coloured border around the thumbnail-sized image preview or a border separating this image from the item information (title, creator, provider and year, if available) (D).

To navigate between different pages of results, the buttons and links in the lower area on each page can be used. They are shown automatically in all perspectives, whenever more than 12 items match the searched keywords (E).

Thumbnails

According to the thumbnail specification for the Europeana prototype, thumbnails for search result presentations are specified with a height of 110 pixels (Verleyen, 2009).

In our first version, thumbnail height was reduced to 55 pixels to make optimal usage of the limited screen size. Since the system provides thumbnails in portrait and landscape orientation and we wanted to keep the aspect-ratio, the resulting preview images often were very small and left a large amount of unused space between image and text, as seen in Figure 22.


1521, Horst Janssen Museum

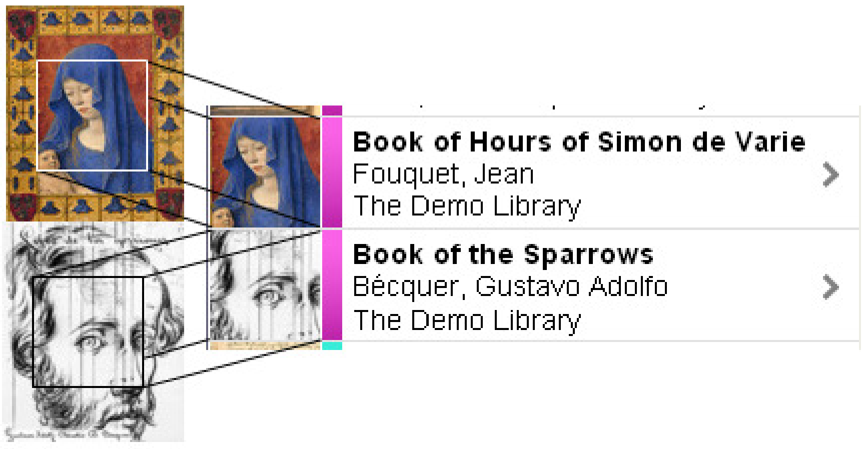
Unknown,
The Code of King Magn...
The Demo Library

Sigismund Augustus,
Privilige of the King of P...
1563, The Demo Library

Swartz, Johan David
Charles XII, portrait
Horst Janssen Museum

Figure 22. Initial presentation of thumbnails in search results

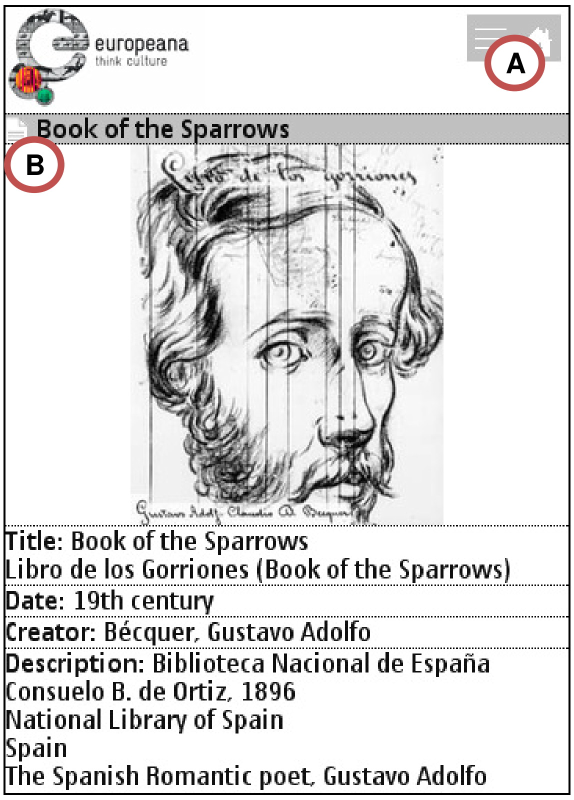
In Figure 23, the result of our reworked thumbnail presentation is shown. By using standard CSS-formatting on the client, the regular image is cropped to a 55x55 pixel image that shows a center region of the original image without increasing load on the server. This way, the available screen size is used in an efficient way, the result page shows a better symmetry in layout and the explorative character of Europeana is emphasized, encouraging users to browse the database.


Book of Hours of Simon de Varie
Fouquet, Jean
The Demo Library

Book of the Sparrows
Bécquer, Gustavo Adolfo
The Demo Library

Figure 23. Reworked thumbnail visualization

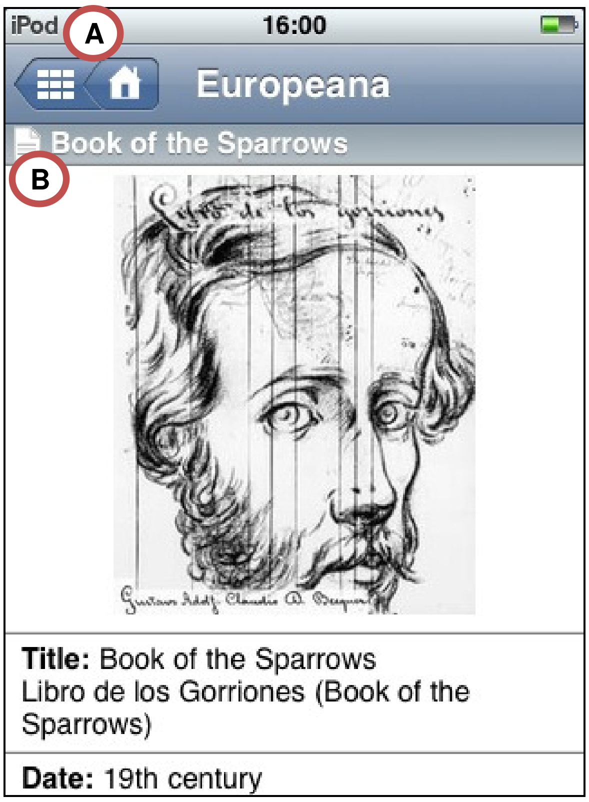
4.4.3 Object presentation



Book of the Sparrows

Title: Book of the Sparrows
Libro de los Gorriones (Book of the Sparrows)
Date: 19th century
Creator: Bécquer, Gustavo Adolfo
Description: Biblioteca Nacional de España
Consuelo B. de Ortiz, 1896
National Library of Spain
Spain
The Spanish Romantic poet, Gustavo Adolfo

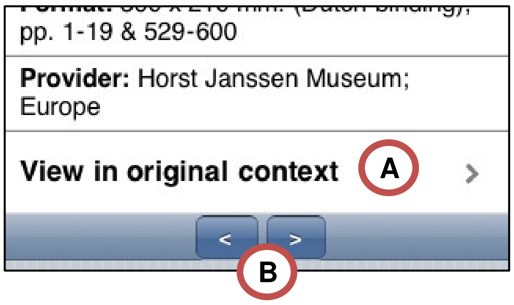
Figure 24. Basic Interface: Object presentation



Book of the Sparrows

Title: Book of the Sparrows
Libro de los Gorriones (Book of the Sparrows)
Date: 19th century

Figure 25. Adv. Interface: Object presentation



pp. 1-19 & 529-600
Provider: Horst Janssen Museum; Europe
View in original context

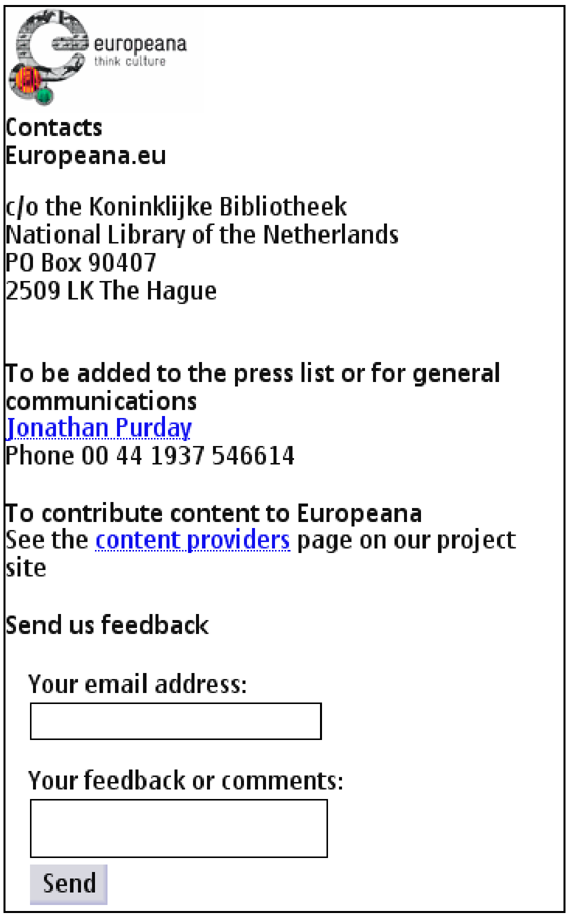
Figure 26. Adv. Interface: Bottom area navigation

To view the object presentation of an item, the user needs to click an item from the search results or select an item from the list of treasures on the index page.

The upper area offers icons to return to the search results or the index page (A). Below, a small icon next to the title indicates the item type (B). By using the links provided in the bottom area (Figure 26), the item can be viewed in its original context (A) or the user can switch to the next/previous item in the result list (B).

4.4.4 Static page example: Contacts

As an example for static page content, the contacts page is depicted in Figure 27 and Figure 28. In order to create this content, we took the “desktop” content for this page and simply changed some of the css-styling to better suit on a mobile device.



Contacts
Europeana.eu

c/o the Koninklijke Bibliotheek
National Library of the Netherlands
PO Box 90407
2509 LK The Hague

To be added to the press list or for general communications
Jonathan Purday
Phone 00 44 1937 546614

To contribute content to Europeana
See the content providers page on our project site

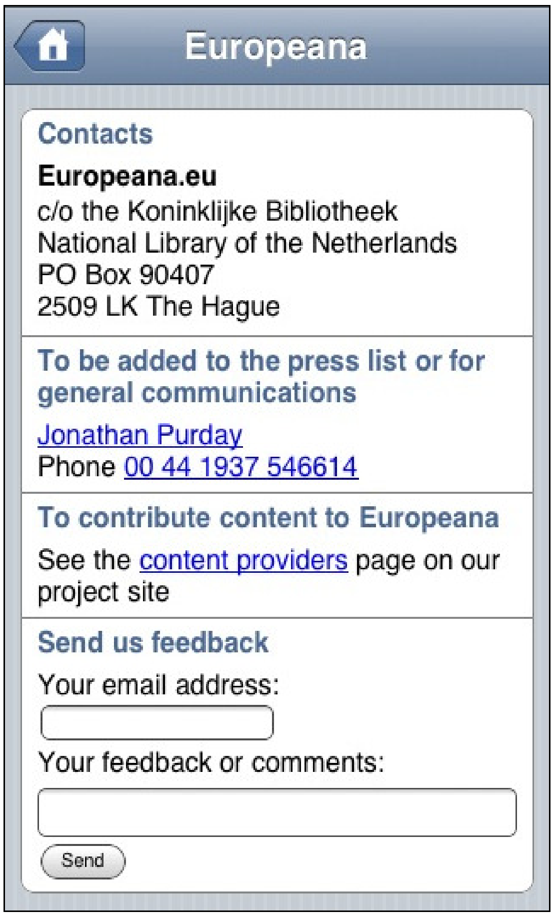
Send us feedback

Your email address:

Your feedback or comments:

Send

Figure 27. Basic Interface: Contact page



Europeana

Contacts
Europeana.eu
c/o the Koninklijke Bibliotheek
National Library of the Netherlands
PO Box 90407
2509 LK The Hague

To be added to the press list or for general communications
Jonathan Purday
Phone 00 44 1937 546614

To contribute content to Europeana
See the content providers page on our project site

Send us feedback
Your email address:

Your feedback or comments:

Send

Figure 28. Adv. Interface: Contact page

4.5 Implementation details

In the following, we will describe the implementation specifics of the mobile Europeana client.

4.5.1 Directory structure

The diagram below shows the directory structure we used to develop the mobile interface. It is located in the `\portal-full\src\main\webapp`-directory.

```
├── mobile
│   ├── css
│   ├── images
│   ├── iwebkit
│   │   ├── css
│   │   ├── images
│   │   ├── javascript
│   │   └── thumbs
│   ├── js
│   └── zflow
└── WEB-INF
    └── templates
        └── mobile
            ├── generic
            └── iphone
```

- `\mobile\css\` contains Stylesheets for both mobile templates
- `\mobile\images\` contains images, logos and icons specially created for the eMobile
- `\mobile\iwebkit\` and subfolders contain files from a 3rd party toolkit used for the layout of the advanced interface
- `\mobile\js\` contains javascript-files
- `\mobile\zflow\` contains the files from a 3rd party toolkit used for the treasure presentation in the advanced index page (see Chapter 4.4.1)

The directory structure and file content of the `WEB-INF\templates`-folder is shown below.

```
WEB-INF
└──── mobile
       │   inc_flow.ftl
       │   inc_footer.ftl
       │   inc_header.ftl
       │   inc_macros.ftl
       │   inc_result_table_brief.ftl
       │   inc_result_table_full.ftl
       │
       ├──── generic
       │        brief-doc-window.ftl
       │        contact.ftl
       │        full-doc.ftl
       │        index_orig.ftl
       │        static-page.ftl
       │
       └──── iphone
                brief-doc-window.ftl
                contact.ftl
                full-doc.ftl
                inc_header.ftl
                index_orig.ftl
                static-page.ftl
```

To ease up maintenance, code between the advanced and basic - and even the desktop - interface is shared, whenever possible. A good example for code sharing between both mobile interfaces and the desktop interface is the language selection: We have been able to completely reuse the code for this feature that was already available as part of the desktop interface. All that was needed was the inclusion of the `language_select` template, the related JavaScript code and some additional styling information.

Freemarker-Files, located directly in the `mobile` directory are implemented to work as includable-files, allowing to be used from any other template:

- `inc_flow` contains the code to create the dynamic treasure-presentation in the advanced index page (see Chapter 4.4.1)
- `inc_footer` is included in every page template to show links to the static pages and the bookmark-button

- `inc_header` provides the basic structure of a webpage (opening tags, meta information, stylesheet-definitions)
- `inc_macros` is used intensively in most of our templates. This file contains helper methods and other useful functions.
- `inc_result_table_brief` contains code to create the search result page's three perspectives.
- `inc_result_table_full` is used to render the content of the object representation page

The names of the files in the `iphone` and `generic`-folder are identical. Each file represents one of the pages. While `generic` is used for the basic interface, `iphone` is used for the advanced interface. The file-naming follows the conventions established by the desktop interface:

- `brief-doc-window` is used for the search result page
- `contact` is used for the contact page
- `full-doc` is the template for the object presentation page
- `index_orig` is used for the index page
- `static-page` is the template for all other static pages

The folder for the advanced interface contains an additional `inc_header` file, since that interface requires more logic and definitions in the header of each page.

4.5.2 Mobile meta tags

Mobile web content is usually identified by using a website's DOCTYPE definition. If the DOCTYPE states that a document is XHTML-MP or WML, then content is declared mobile-optimized, by definition. However, today's advanced mobile devices and modern smartphones are capable of rendering "normal" desktop content, most of them with almost full JavaScript and AJAX support. A lightweight and responsive full-HTML mobile web experience provides the best user experience across a mobile network and on the smartphone browser. Since a smartphone-optimized mobile web document uses the full tag set of HTML, its DOCTYPE is no longer the right criteria to use to decide whether the document is optimized for mobile devices.

With the help of mobile meta tags, documents are identified as mobile optimized, therefore we make use of the following tags:

The HandheldFriendly meta tag

```
<meta name="HandheldFriendly" content="true" />
```

Originally developed for AvantGo browsers on Palm devices, this tag is today interpreted by most mobile browsers as a directive to display a document without scaling.

The MobileOptimized meta tag

```
<meta name="MobileOptimized" content="[width in pixels]" />
```

The MobileOptimized tag was developed by Microsoft to enable developers to enhance their content for Internet Explorer Mobile. The presence of this tag prevents the browser from modifying the layout, assuming a non-optimized page. Instead, this tag forces a single-column layout with the specified width (Microsoft, 2006).

This tag is also used by other mobile browsers to force a single-column layout, as well as mobile search engines to identify mobile-optimized web pages.

The Viewport meta tag

```
<meta name="viewport" content="width=[width in pixels], height=[height in
pixels], user-scalable=no, initial-scale=1.0" />
```

The Viewport tag controls the dimensions and scaling of the browser viewport window. It is widely supported by WebKit-based mobile browsers, including but not limited to Safari Mobile on the iPhone, the Android browser, webOS browser in Palm's newer devices (Pre and Pixi) and Blackberry browsers. The presence of the Viewport tag indicates that the document is optimized for mobile devices. The example shows the Viewport directives we used: width and height of the device's screen; to disable zooming, `user-scalable` is set to no, together with an `initial-scale` of `1.0` (a non-zoomed document).

To set a device's specific screen size in one of the latter two meta tags, the system uses the result of the device recognition that is injected into the ModelAndView object after the requesting device has been identified as mobile device (see Figure 31, Appendix for details).

4.5.3 Web apps

We based the design of the advanced interface on the look and feel of a native application on the iPhone, a so-called "web app". This presentation looks particularly well on the iPhone, but still very good on other WebKit-based mobile browsers of other devices.

There are a few web-app frameworks available as open-source software, e.g. iUI (iPhone User Interface Framework), iWebKit or UiUIKit (Universal iPhone UI Kit) (for a more complete list, see: http://komparable.com/133/mobile-frameworks-for-iPhone-WebApps). By using a combination of standard html markup, CSS, JavaScript and/or Ajax their goal is to mimic the appearance of a native phone application. Unfortunately, most of these projects seem to be inactive and unmaintained. Therefore, we decided to use the iWebKit-framework (http://www.iwebkit.info/), being the most active project that is still maintained and supported by the community.

For iPhone users, the iWebKit framework also offers an interesting "full screen" mode. By adding a new meta tag to the head section of each document, the mobile browser hides the task and navigation bars, like a native application does:

```
<meta name="apple-mobileweb-app-capable" content="yes" />
```

There are two more tags that may be used to support this behavior:

```
<link rel="apple-touch-icon" href="image.png" />

<link rel="apple-touch-startup-image" href="startup.png" />
```

These two items are interpreted, when a webpage is added as a bookmark to the homescreen of an iPhone. It will modify the icon for the bookmark and show a loading image while the phone is loading the index page.

As shown in Chapter 4.5.1, the files of the iWebKit framework are located in the directory `\portal-full\src\main\webapp\mobile\iwebkit\`. Since we needed to add a

Europeana-specific stylesheet, we created an additional file, `iwebkit_addon.css`, where we added new styling declarations as well as overrides for existing styling information. This way, the original files may always be updated with the latest release without the need for additional patching when a newer version is released.

4.5.4 Semantic Layer integration

According to the Description of Work for Task 3.4., the mobile client could make use of the Semantic Layer developed in EuropeanaConnect WP1. In a meeting with the developers of the Semantic Layer it turned out that this feature will be integrated in a transparent way (Hesselmann, 2010): From a user point of view, the original search functionality of the Europeana portal can still be used. Synonyms will be automatically created in the background by the semantic layer and will produce other or more search results.

According to this information, there is currently no need to integrate more sophisticated query mechanisms, e. g. SPARQL, to access the semantic layer. It will most likely be possible to utilize the functions of the semantic layer using the SOLR-based search functions of Europeana.

4.5.5 Testing

Unit Tests

The Unit tests we provided to ensure a proper testability of the code are located in the folder `\portal-full\src\test\java\de\offis\europeana\mobile\`. The file `DeviceRecognitionTest.java` contains three jUnit tests to check the correct detection of mobile devices. Besides, we added three tests to ensure the right identification of desktop browsers.

Simulators

Although a physical device is the best choice when it comes to testing, often a simulator is sufficient to do a quick test of a new feature. The list below covers a wide range of mobile devices and platforms. All simulators are at least available for Microsoft Windows, most of them also for Mac and Linux operating systems. However, the official iPhone simulator is available only for MacOS. For other platforms, Apple's Safari Browser offers a developer mode that can be used for simulating an iPhone-like device.

- Android Emulator
  http://developer.android.com/guide/developing/tools/emulator.html
- Opera Mobile emulator
  http://www.opera.com/developer/tools/
- Symbian, S60 platform
  http://www.forum.nokia.com/info/sw.nokia.com/id/ec866fab-4b76-49f6-b5a5-af0631419e9c/S60_All_in_One_SDKs.html
- Palm webOS SDK
  http://developer.palm.com/index.php?option=com_content&view=article&layout=page&id=1788&Itemid=21
- Blackberry simulators
  http://na.blackberry.com/eng/developers/resources/simulators.jsp

- Windows Mobile
    - Windows Mobile 5.0 Smartphone Emulator Images http://www.microsoft.com/downloads/details.aspx?familyid=52FED581-8F8D-4C46-9966-4832098191B7&displaylang=en
    - Windows Mobile 6.1.4 Emulator Images http://www.microsoft.com/downloads/details.aspx?FamilyID=1a7a6b52-f89e-4354-84ce-5d19c204498a&displaylang=en

  Getting Network Access on the Windows Mobile Emulator: http://betterthaneveryone.com/archive/2008/08/31/getting-network-access-on-the-windows-mobile-emulator.aspx

4.5.6 Code repository

The entire code and all files mentioned throughout this document have been committed to the central Europeana code repository. It is accessible either via subversion (see the Europeana Development Guide for details, http://europeanalabs.eu/wiki/DevelopmentGuide) or browseable online: http://europeanalabs.eu/browser/europeana/trunk.

The integration of the mobile client into the Europeana Framework, Subtask 3.4.5, has been taken into account from the beginning of development:

The developed code was committed to the “contribute” directory of the central Europeana code repository from where it has been picked up and integrated into portal-lite by the core development team. (http://europeanalabs.eu/ticket/607). After some additional testing and bug fixing, this code was finally migrated to portal-full. (http://europeanalabs.eu/changeset/2698)

4.5.7 3rd party libraries

eMobile makes use of the following 3rd party libraries and frameworks:

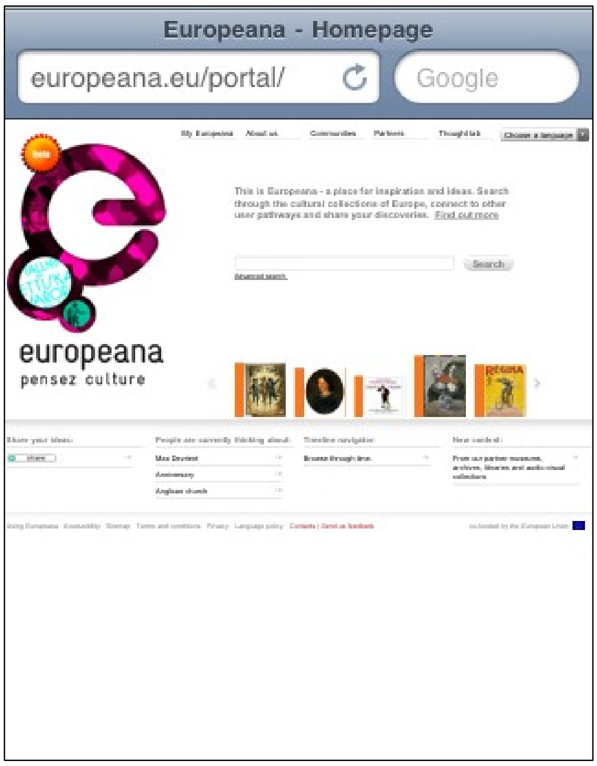
- WURFL device library and Java API. http://wurfl.sourceforge.net/
- SimpleDesktop Matching Engine from Tera-WURFL. http://www.tera-wurfl.com/
- iWebKit webapp framework. http://www.iwebkit.info/
- css-vfx: JavaScript and css effects library. http://code.google.com/p/css-vfx/
- Freemarker template engine. http://freemarker.sourceforge.net/

5 Conclusion

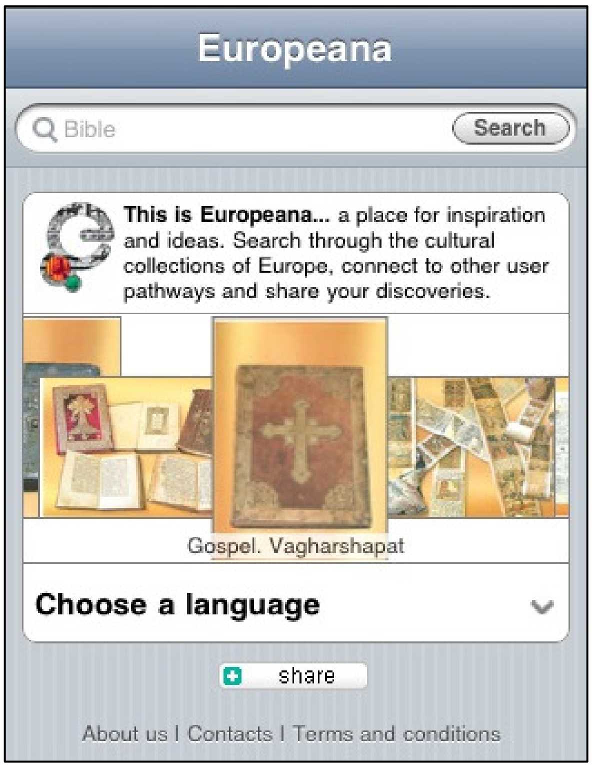
In this document, we have presented the design and implementation of a middleware and web server for accessing Europeana.

We started with a presentation of the underlying design process and a summary of the results from our user requirements analysis. We then presented concepts, a solution for mobile device recognition and its integration into the Europeana Framework. Afterwards, we created mockups for a mobile presentation of Europeana and our implementation of two interfaces, one basic interface and a more advanced interface for modern devices. In the following, we then discussed some of the details and design decisions worth mentioning from the implementation phase.

The mobile client we created has already been integrated into the Europeana portal and is supposed to go live with the Rhine release. It offers an appealing interface adapted to the needs of users in a mobile environment.



This is Europeana - a place for inspiration and ideas. Search through the cultural collections of Europe, connect to other user pathways and share your discoveries. Find out more



This is Europeana... a place for inspiration and ideas. Search through the cultural collections of Europe, connect to other user pathways and share your discoveries.

Figure 29. The index page displayed on an iPhone

Figure 30. The index page adapted for a mobile device

The side-by-side comparison of the index page shows the advantages the mobile interface (Figure 29) has over the desktop interface (Figure 30): Without optimization to the available screen size, the content of the page is hard to capture and it needs extensive zooming and scrolling to navigate between all elements (e.g. language selection and search box). By hiding the address bar when it is not used, the small screen can be used in an efficient way.

Changes in production environment: After we handed our code over to the Europeana Office, some minor changes have been applied to the mobile client. Due to the fact that this Deliverable covers the design of our interface, the actual mobile layout presented on the Europeana portal now differs slightly from the screenshots in this document. For example, the list of treasures, as described in chapter 4.4.1 has been de-scoped while the language selection in the advanced interface is now shown above the logo. However, a test-server – with a limited demo dataset – that may be used for comparison is available as described in the appendix.

5.1 Suggestions

Website statistics with Google analytics: The Europeana portal uses Google Analytics to track users and gain website statistics. Google analytics can be added easily, by adding a small bit of Javascript to each page. However, the lack of JavaScript on older devices is an important aspect that needs to be considered for statistic on mobile pages. To solve this problem, Google created a special implementation for statistics, “Google Analytics for Mobile” (http://code.google.com/intl/en/mobile/analytics/) that may be implemented server-sided.

Validation errors: The mobile interface is valid XHTML 1.0 Transitional code. However, some tests, e.g. the mobiReady test (http://ready.mobi/) show an incorrect score because of errors in the used W3C validator. The validation error is caused by the social bookmarking button that is perfectly valid XHTML 1.0 Transitional, but is not validated correctly due to a limitation of the W3C test (http://www.addthis.com/help/client-api#valid-xhtml). As soon as the button is disabled, both tests pass and the mobile readiness score reaches 4-5 (with 5 being maximum).

Memory Usage for WURFL's device database: Since the library constantly grows – as new devices are added to it – device detection may become slow and more time consuming. To avoid this issue, the usage of the WURFL customizer tool provided by the Tera-WURFL project (http://www.tera-wurfl.com/wiki/index.php/WURFL_Customizer) may be considered: This small utility allows reducing the size of the xml file that contains all devices information, by removing all capabilities that are not needed.

WURFL database updates: It is recommended to regularly update the WURFL library to support future devices. Updates are provided on a monthly base by the maintainers (currently available at http://sourceforge.net/projects/wurfl/files/WURFL/latest/wurfl-latest.zip/download). The official mailing list (http://tech.groups.yahoo.com/group/wmlprogramming/) is also a useful resource for the latest news in development of the WURFL API as well as information on new mobile devices and how to support them best.

Thumbnail generation: In its current implementation the thumbnails for the mobile interface are cropped to a smaller size via CSS on the client-side, as described in chapter 4.4.2. To further reduce loading times and transferred data, an extension of the Europeana thumbnail cache – that then also serves thumbnails already suited for mobile devices – could be implemented.

5.2 Next steps

After designing and implementing the mobile Europeana Clients, the next step is the evaluation of the mobile clients according to the defined requirements in Task 3.4.4. The goal of the evaluation is to investigate if the applications developed in this task satisfy the requirements which have been identified in task 3.4.1. Additionally, the requirements themselves will be subject to evaluation with the goal of revealing potential future improvements and extensions.

When the Europeana Rhine release goes live, the Europeana Office will gain interesting statistics on the usage of the mobile interface. This information will be a useful resource that could - together with user feedback and the results of our evaluation - form the requirements for future developments of the mobile client and establish the mobile client as an important access channel for Europeana.

References

Apache Software Foundation. 2010. *Traffic Server Software Developers Kit. Chapter 10. HTTP Headers*. [Online] 2010. [Cited: 28 07 2010.] http://trafficserver.apache.org/docs/v2/sdk/HTTPHeaders.html.

ISO. 1998. *Ergonomic requirements for office work with visual display terminals - Part 11: Guidance on usability* . s.l. : DIN / ISO, 1998.

ISO. 2006. *Ergonomics of human-system interaction - Part 110: Dialogue principles* . s.l. : ISO, 2006.

ISO. 1999. *Human-centred design processes for interactive systems.* s.l. : ISO, 1999.

Georgieva, E. and Georgiev , T. 2007. *Methodology for mobile devices characteristics recognition*. International Conference on Computer Systems and Technologies : CompSysTech 2007

Gartner. 2009. (via TECH cocktail. 2010.) *Webkit Marketshare* [Online] 2010. [Cited: 28 07 2010] http://techcocktail.com/home/2010/05/19/mobile-os-fragment-as-the-mobile-web-converges/blog-webkit-marketshare/.

Gartner Inc. 2010. Gartner's Top Predictions for IT Organizations and Users, 2010 and Beyond: A New Balance. Gartner Inc., 2010

Haskiya, David. 2010. *Minutes from Skype call between EDLF and OFFIS on Mobile access to Europeana*. [Online] [Cited: 30 07 2010] [only project internally accessible] https://version1.europeana.eu/c/document_library/get_file?p_l_id=16989&folderId=24260&name=DLFE-6041.doc

Hesselmann, Tobias. 2010. *Minutes: Interface from Mobile Client to Semantic Layer*. [Online] [Cited: 30 07 2010] [only project internally accessible] https://version1.europeana.eu/c/document_library/get_file?p_l_id=16989&folderId=24260&name=DLFE-11330.pdf

Microsoft. 2006. *Layout Meta Tag*. [Online] [Cited: 28 07 2010] http://msdn.microsoft.com/en-us/library/ms890014.aspx.

OFFIS Institute for Information Technology. 2009. D3.4.1 Catalogue of User requirements. http://www.europeanaconnect.eu/documents/D3.4.1_eConnect_Catalogue_of_User_Requirements_v1.0_20091222..pdf

Platinum Solutions. 2008. *Spring MVC*. [Online] 2008. [Cited: 28 07 2010.] http://www.platinumsolutions.com/tech_library/Spring%20MVC.ppt.

Siebinga, Sjoerd, Purday, Jon and van der Werf, Bram. 2009. *Guidelines for the use of EuropeanaLabs.* 2009. Version 1.

Siebinga, Sjoerd. 2009. *Portal Modules*. 2009. Version 5. [Online] 2009. [Cited: 28 07 2010]. http://europeanalabs.eu/wiki/DevelopmentPortalModules?version=5.

Verleyen, J. 2009. *Thumbnails in Europeana Prototype*. [Online] 2009. [Cited: 28 07 2010.] http://version1.europeana.eu/c/document_library/get_file?uuid=6b52d4be-6a4d-443a-842a-ab991bca2b1f&groupId=10602.

Description of software developed for Europeana within EuropeanaConnect

This software demonstrates the features of the web server based mobile access to Europeana. Using this software, users are able to access the contents of Europeana from their mobile devices. It contains the following features:

- **Device Recognition.** The capabilities of the accessing mobile devices are automatically recognized by the Europeana web server. The user is served a mobile web site, which is dynamically optimized to the capabilities of his / her smartphone.
- **Web based application.** The mobile client is realized as a web based application, and is thus adaptable to a multitude of available mobile devices. It is accessible via the mobile web using ordinary mobile browsers.
- **Basic search functionality.** The software offers users the possibility to search the contents in Europeana by entering search strings, comparable to the normal Europeana web site.
- **Gesture based browsing.** Users with appropriate mobile devices (iOS, Android) are able to search through popular Europeana “treasures” using a gestural carousel browser on the home screen.
- **Three levels of detail.** Search results are presented in one of three views: A gallery view, a text-only view, and a mixed mode consisting of thumbnails and text. Users can thus adjust the result browser to their current usage scenario.
- **Detailed item view.** Users can select items from the carousel browser or the search results screens in order to view them at full detail, including a bigger thumbnail, as well as a detailed description of the selected item.

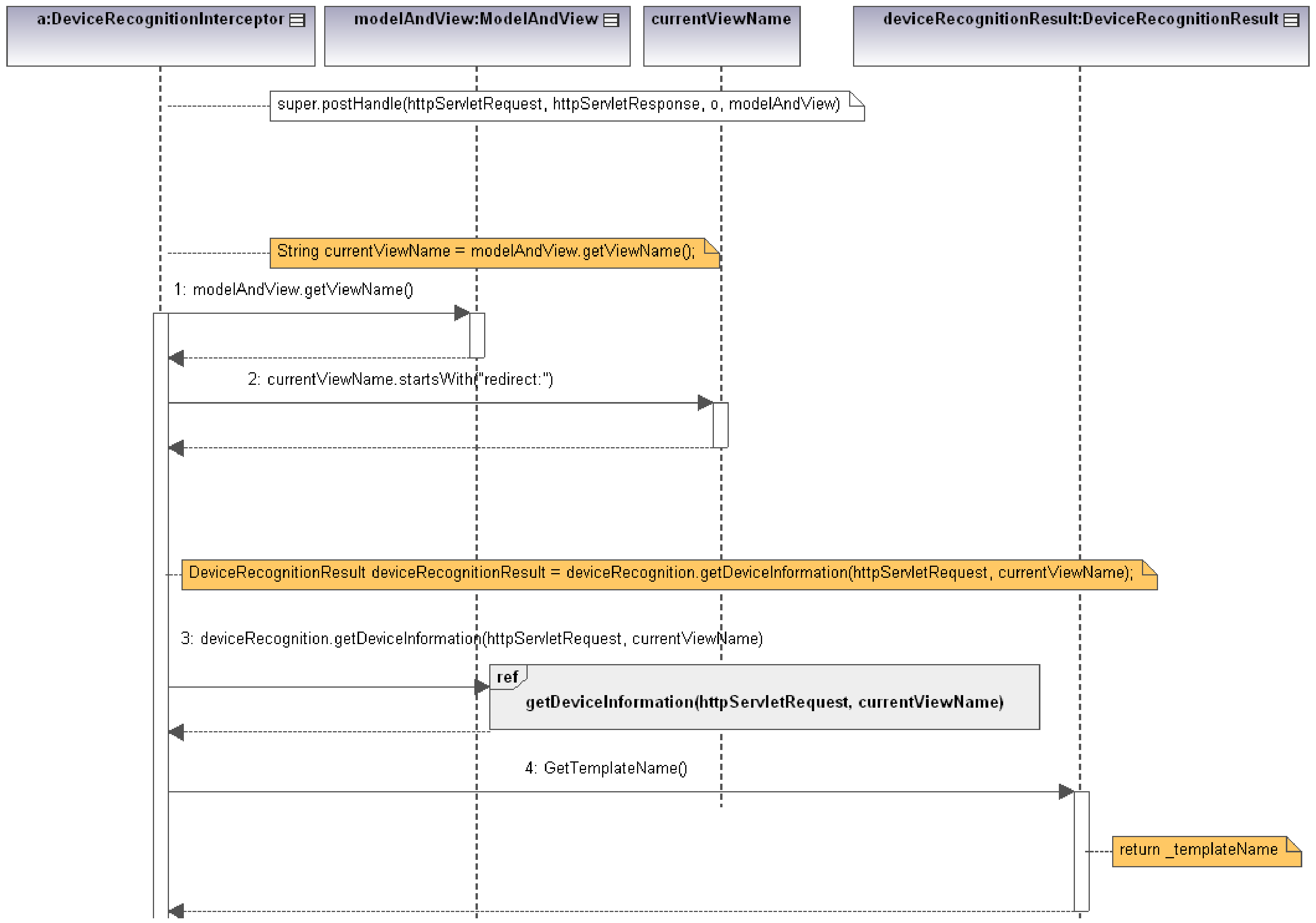
Link to software	http://134.106.50.15:81/portal/
Development environment	IntelliJ IDEA
Programming language used	Java 6.0, JavaScript
Application server used	Jetty 6.1.X
Database requirements	Postgres 8.x
Operating system requirements	Windows XP or higher or Debian Linux
Port requirements / default ports used	HTTP on Port 81
Interface	Web service over HTTP
Licensing conditions	EUPL, GPL, BSD

List of figures

Acronyms

Acronym	Meaning
AJAX	Asynchronous JavaScript and XML
AOP	Aspect-Oriented Programming
API	Application Programming Interface
CSS	Cascading Style Sheets
GPRS	General Packet Radio Service
HCD	Human-Centred Design
HSCSD	High-Speed Circuit-Switched Data
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
ISO	International Organization for Standardization
MVC	Model-View-Controller
UAProf	User Agent Profile
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
WML	Wireless Markup Language
WURFL	Wireless Universal Resource FiLe
WWW	World Wide Web
XML	Extensible Markup Language

Appendix A: Sequence diagram of Interceptor for mobile device detection


a:DeviceRecognitionInterceptor
modelAndView:ModelAndView
currentViewName
deviceRecognitionResult:DeviceRecognitionResult
super.postHandle(httpServletRequest, httpServletResponse, o, modelAndView)
String currentViewName = modelAndView.getViewName();
1: modelAndView.getViewName()
2: currentViewName.startsWith("redirect:")
DeviceRecognitionResult deviceRecognitionResult = deviceRecognition.getDeviceInformation(httpServletRequest, currentViewName);
3: deviceRecognition.getDeviceInformation(httpServletRequest, currentViewName)
ref
getDeviceInformation(httpServletRequest, currentViewName)
4: GetTemplateName()
return _templateName

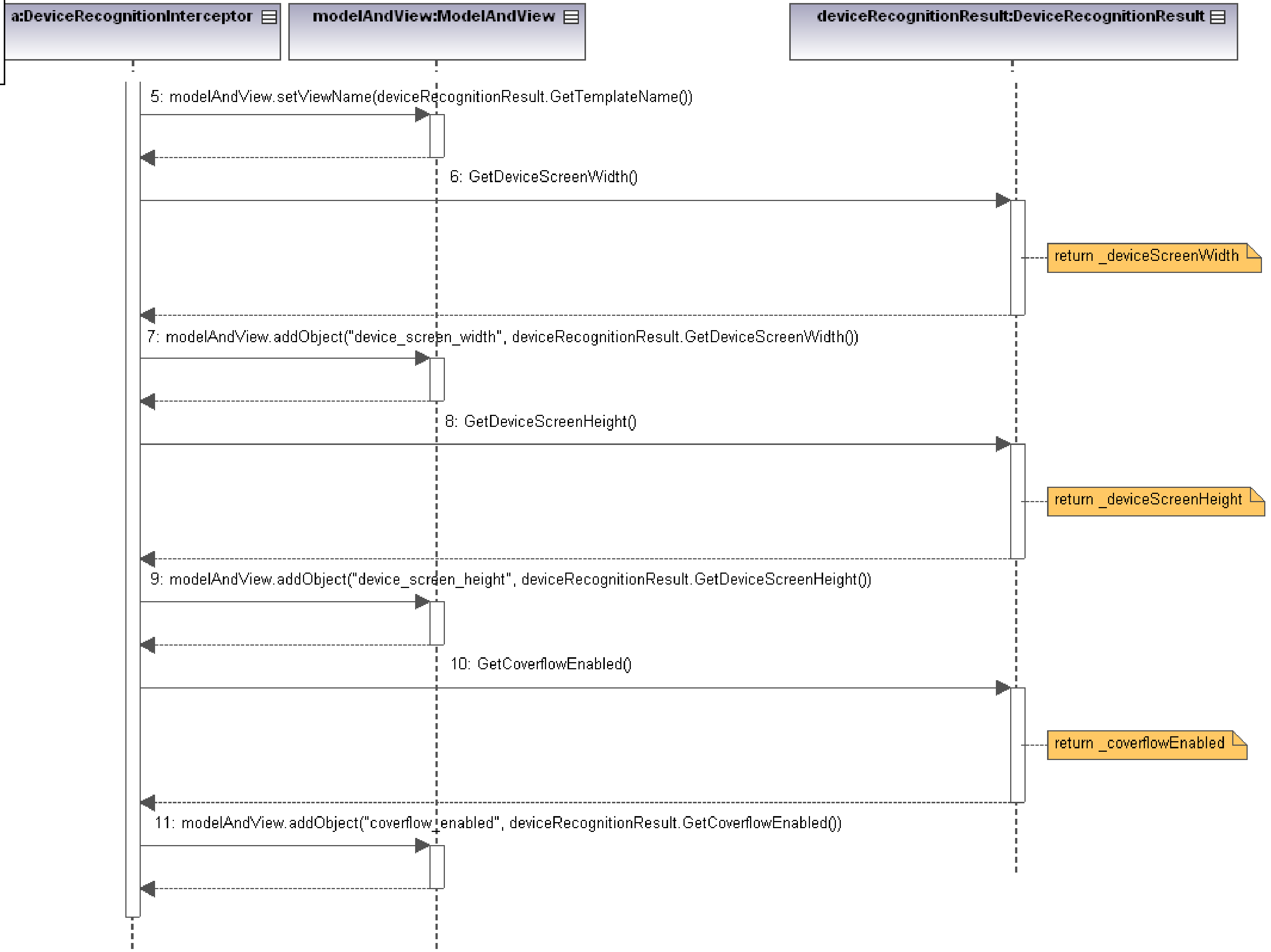

a:DeviceRecognitionInterceptor
modelAndView:ModelAndView
deviceRecognitionResult:DeviceRecognitionResult
5: modelAndView.setViewName(deviceRecognitionResult.GetTemplateName())
6: GetDeviceScreenWidth()
return _deviceScreenWidth
7: modelAndView.addObject("device_screen_width", deviceRecognitionResult.GetDeviceScreenWidth())
8: GetDeviceScreenHeight()
return _deviceScreenHeight
9: modelAndView.addObject("device_screen_height", deviceRecognitionResult.GetDeviceScreenHeight())
10: GetCoverflowEnabled()
return _coverflowEnabled
11: modelAndView.addObject("coverflow_enabled", deviceRecognitionResult.GetCoverflowEnabled())

Figure 31. Sequence diagram of Interceptor for mobile device detection (continued)