Praekelt Patterns

Release 1

Praekelt Foundation devs and individual contributors

Jul 12, 2017

Contents

Naming things is hard, naming things consistently is even harder. The goal of this patterns repository is to help all stake holders in a project to know and understand what we call the individual parts that together make up the things we build.

Patterns described in this repository should:

1. Have a short name.
2. Have a paragraph (with or without illustration) describing the pattern. The paragraph should be suitable for non-technical people to understand and re-use when describing possible solutions to client problems.
3. A link to a sample implementation of the pattern. The sample implementation should have tests and be ready for a developer to look at understand and apply.

Note: You are free to contribute to this repository, it is automatically published from changes made in the the GitHub patterns repository. The full backlog of things that may possibly need to be described is listed in the patterns-backlog. Feel free to grab any of those and start work on it. Remember to remove it from the backlog when finished.

CHAPTER 1

Mobi & Web Patterns

These are a collection of patterns that mostly apply to Mobi & Web sites we develop.

Age Gateways

Introduction

In campaigns where alcohol brands are involved, an Age Gateway is required. This document covers the Age Gateway requirements of the Diageo sites (Smirnoff, Guinness, etc). An age gateway prompts the visitor for country and date of birth, in order to confirm that the visitor is of legal drinking age in their region. Visitors who are not of drinking age will receive an explanation and will be denied access to the rest of the site.

Types of Age Gateways

- Diageo's "The Perfect Age Gateway", Web - This is a javascript-based age gateway hosted on the Neo platform. It provides an overlay over the current site, allowing visitors a sneek peek which results in more follow-through. This age gateway doesn't work on devices that don't have javascript support or have it disabled - so it is not suitable for feature phone mobi sites.
- Jmbo Age Gateway - An age gateway implemented by Praekelt for the Jmbo CMS. This works for both web and mobi. While it is integrated with Jmbo, it can be used as inspiration for a Django age gateway for sites not using Jmbo. See the GFM project for an example of this used outside of Jmbo.
- USSD Age Gateway - An age gateway developed for the USSD platform.

Scope of Work

Considerations:

- Which front-ends will the project support? An age gateway needs to be implemented for every required front-end.

- Does the project require a mobi-site frontend? Remember that the Diageo Perfect Age Gateway only supports javascript enabled devices, which doesn't include any low-end devices.
- The age gateway will be the first page that visitors interact with. It needs to attract users to the rest of the site without violating Diageo content policies. (see links below for more information)

Limitations

- Passing links through the gateway was a limitation for the Jmbo age gateway but this has been implemented in December 2013.

Code

- Documentation and code for the Diageo Perfect Age Gateway documentation should be obtained from the client's Basecamp to ensure the project uses the latest version.
- An implementation of a USSD age gateway has been implemented in Figo for the GFM project, somewhere in *https://github.com/praekelt/gfm-ussd-service/tree/develop/figo*
- Jmbo age gateway: *https://github.com/praekelt/jmbo-foundry/blob/develop/foundry/middleware.py#L42*

Screenshots

- The USSD Age Gateway is accessible on *120*8864*1139#

Lingo

- USSD - Unstructured Service Supplementary Data - see *http://www.truteq.co.za/tips_ussd/*
- AG - Age Gateway
- Mobi - Low-end device web platform
- Web - Modern web platform which utilizes xhtml, css, javascript, html5, css3 or a combination thereof
- Jmbo - A mobile-friendly CMS developed by Praekelt

See Also

- Diageo Marketing Code: http://www.diageo.com/en-row/newsmedia/Pages/resource.aspx?resourceid=1287

Responsive Web Design

Introduction

When web browsers were first introduced on mobile phones, it had very limited rendering functionality. Stylesheet support was poor, if even supported at all. There was also no font support, amongst many other differences compared to desktop web browsers.

On smartphones, this has changed completely. The rendering engines used on smartphones are nearly identical to those found on the desktop web browsers.

The result of this recent development in mobile web browsing means that the biggest differences between a desktop version of a website and a mobile version are: - Screen size - Touch-friendlyness on mobile devices

Because of the small amount of differences possible on a mobile and desktop version of a web site, the concept of a responsive site is born.

A responsive utilises java-script that runs on the device that makes minor modifications to the page depending on screen-width.

Typically, if the screen width is found to be less than a certain value of pixels, the menu will be collapsed and be replaced by a menu icon that contains the menu entries. This reduces the amount of space the menus utilize, creating more space for content while also being more touch-friendly.

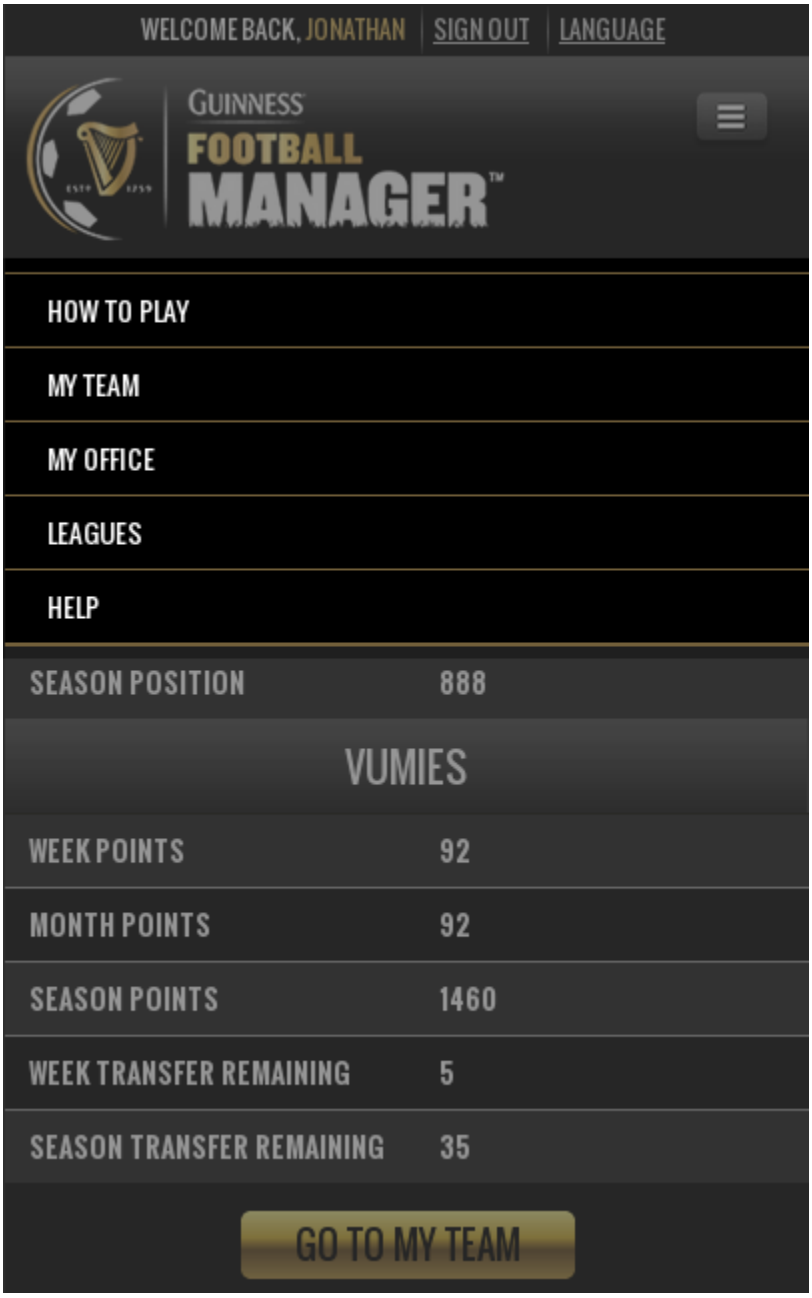
In some cases, sections of pages and tables are also collapsed to allow for more content. Pressing an icon would typically expand these to their full size.

Screenshots

The screenshot below depicts how the Guinness Football Manager responsive site renders on when there is enough screen-width available for a full view. This is typically what it would look like on a desktop or tablet computer:

The screenshots below depict what the exact same webpage looks like when the width is reduced. The client-side javascript will automatically apply the required changes. A red arrow is added to The screenshot on the left to indicate the location of the menu button, while the screenshot on the right has been modified to highlight the resulting menu when the button is pressed.


WELCOME BACK, JONATHAN
SIGN OUT
LANGUAGE
GUINNESS
FOOTBALL
MANAGER™
MY OFFICE
MY RANKING
WEEK POSITION 391
MONTH POSITION 391
SEASON POSITION 888
VUMIES
WEEK POINTS 92
MONTH POINTS 92
SEASON POINTS 1460
WEEK TRANSFER REMAINING 5
SEASON TRANSFER REMAINING 35
GO TO MY TEAM



Scoping Considerations

Low-end Devices:

Responsive sites are extremely effective on desktops, tablets and smartphones, but render very poorly (if at all) on low-end devices - especially devices that don't support javascript.

If a project requires support for low-end (feature) phones, then a server-side implementation is required that will provide an entirely diffrent page which is more suited for basic devices.

Supporting low-end devices as well as the standard devices listed above requires a significant amount of work. Please refer to the Feature Phone Template pattern for more information.

Code

- Twitter Bootstrap provides an example of responsive code plus many elements:
- Github: https://github.com/twbs/bootstrap
- Demo: http://getbootstrap.com/

See Also:

- Examples of responsive sites:
- KCB Bank Group: http://www.kcbbankgroup.com/
- Guinness Football Manager: http://gfm.guinnessvip.com/
- MPowering Health: http://mpoweringhealth.org/
- Related patterns:
- Feature Phone Template

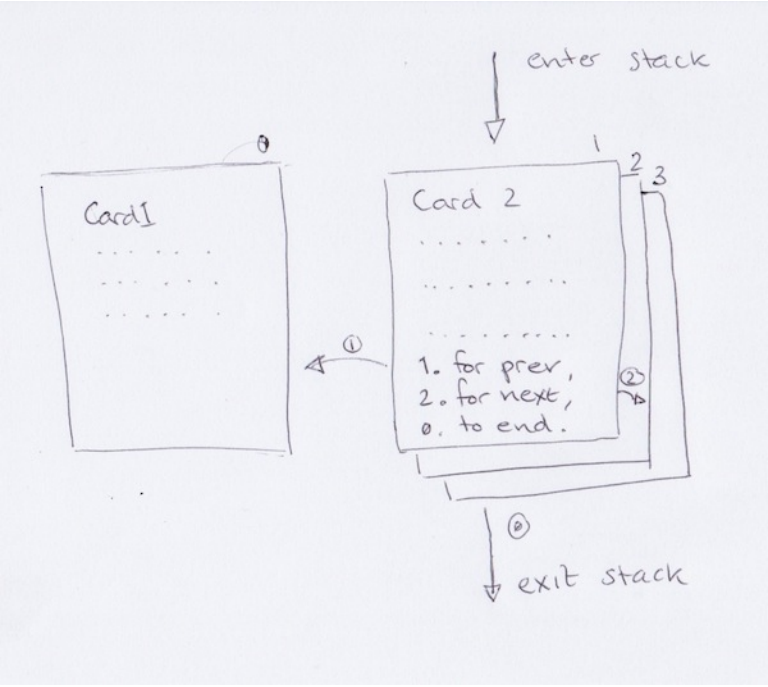
CHAPTER 2

Messaging & USSD Patterns

These are a collection of patterns that apply to applications designed for messaging. The code examples mostly all apply for Vumi.

Card Stack

A card stack allows an end-user to read small bits of content in chunks. The content can be dynamically loaded from an external API. The end-user can navigate between them using next, previous & exit controls.



Code

The state that implements this pattern is called the BookletState and is documented at the Go JSBox documentation.

Implementations

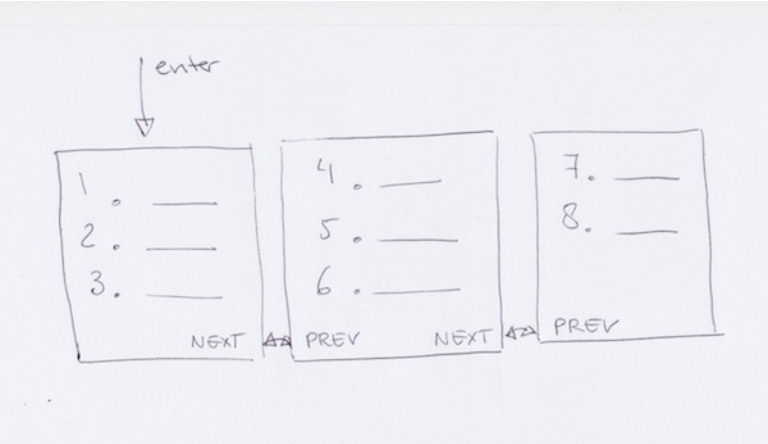
- Girl Hub Rwanda uses a card stack to display content over USSD which is loaded from a Django based TastyPie API.
- Wikipedia Text uses a card stack to allow a user to retrieve chunks of a Wikipedia article's content via SMS.

Notes

- For USSD there is generally a maximum window of 30 seconds within which the content needs to be displayed, read and be responded to before a timeout is reached.
- There is no limit on how many pages a Card Stack can contain. If this uses a session based transport like USSD the limits are all determined by the total session length that the mobile network operator imposes. In South Africa this is 3 minutes but this varies per country.

Paged Result Set

The paged result set allows an end-user to page through a set of results that is larger than the communication channel is able to display fully.



Code

The state that implements this pattern is called the PaginatedChoiceState and is documented at the Go JSBox documentation.

Implementations

- RTS Zambia uses a paged result set to allow an end user to page through a list of districts.
- Switchboard uses a paged result set to allow a community health worker to select a cadre from a long list of cadres.

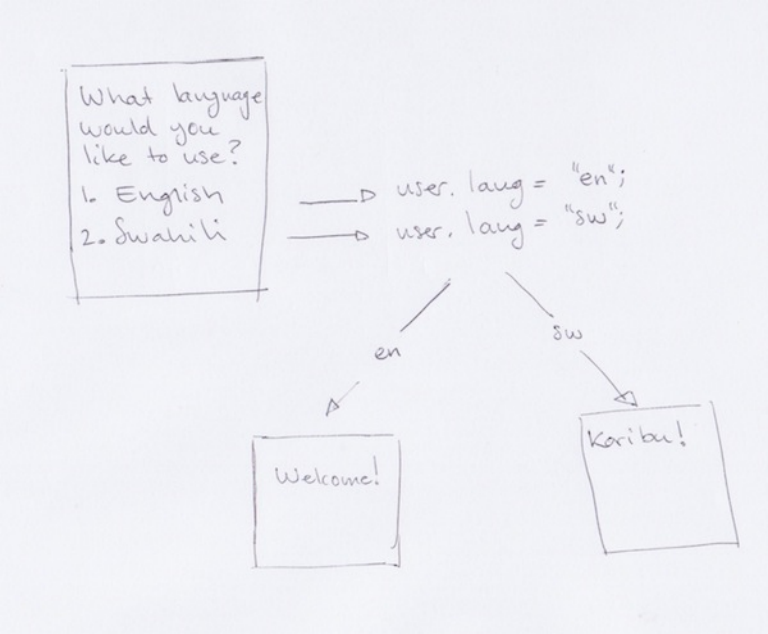
- mAgriculture used a paged result set to allow a farmer to select crops from a long list of known crops.
- NewsWorld uses a paged result set to display a list of news headlines.

Notes

- the PaginatedChoiceState is a variation of the ChoiceState but adds the ability to page through a larger set.

Language Choice

The language choice pattern allows a user to select their language choice of preference. The language choice is only sticky for the duration of the session. If language choice preference is to be maintained across sessions it needs to be stored somewhere more permanent than a temporary session store.



Code

The state that implements the language choice is called the LanguageChoice and is documented at the Go JSBox documentation.

Implementations

- Switchboard uses the LanguageChoice to allow the user to select a language of preference for the duration of a USSD session.

Notes

- Vumi Go's `i18n` machinery works in tandem with the InteractionMachine's utilities for fetching of translations for the selected language.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search