User interface design practices in simple single page web applications

Jiří Tesařík, Ladislav Doležal and Christian Kollmann
Institute of Medical Biophysics, Palacký University in Olomouc, Czech Republic
Center for Biomedical Engineering and Physics, Medical University Vienna, Austria
tesarik@inf.upol.cz, ladol@tunw.upol.cz, christian.kollmann@meduniwien.ac.at

Abstract

The single page web design is an extreme approach of designing the web applications. It limits the structure of the application to fit into only one single physical web page and suppresses the further navigational capabilities. Such limitations strongly affect the design process of the application development, putting more emphasis on the design of the user interface components. We present and discuss the list of five higher level user interface design principles with implementation examples that make the development process easier and bring additional application usability.

1. Introduction

Common web applications consist of dynamically created mutually interlinked but separated web pages. It is natural, because of web infrastructure based on hyperlinks and its navigational nature. Unfortunately this concept also affects the web user interfaces.

On the contrary, the majority of desktop applications contain their user interface elements integrated inside a single parent window (form) and their event driven user interfaces are more compact and do not exceed the application frame. Application data are displayed and edited at the same location inside the containing form. Applications are highly interactive and user interface elements provide immediate feedback to user inputs.

Modern web design, on the other hand, tries to bring the look and feel of web applications closer to classical desktop applications, since the inter-page navigation is suppressed or missing at all and replaced by other user control and page infrastructure techniques (asynchronous remote calls, partial post backs) which provide more desktop application look and feel. One of such technologies is the Ajax.

The benefit of this approach lies especially in better user experience, enabling well-known desktop application features: rich user interface elements, event handling, reduced response times and efficiently put down drawbacks of web design, e.g. tiresome loading times and screen flickering while navigation and post backs, awkward user interface limited to just basic set of input and interactive elements.

The organization of this paper is as follows. Section 2 presents short overview of main principles and technological coverage of given area. Section 3 discusses the current state of the art applications. Section 4 describes the case study application design and implementation overview. Section 5 shows a short dictionary of user interface design principles with corresponding examples. Section 6 concludes our article.

2. Single page web application design

Single page design is an extreme way to build up a web application in *one single physical web page* as the all-in-one page solution with no navigation, just based on already mentioned dynamic and asynchronous techniques. However this approach puts the developer before some substantial challenges.

It is demanded to design user interface which would show all important information and controls to manipulate application data in one single page. Page design has to be elaborated to maximize page decomposition into distinct components encapsulating main use cases.

Additionally, most of these richer features cannot be implemented without using additional technologies such as dynamic HTML, JavaScript, Ajax, Macromedia Flash or another. These technologies of course require additional knowledge to be learned.

3. Recent works

Though the Ajax technology is not entirely new – the first application of Outlook Web Access comes from the year 2000, the majority of recent e-commerce web applications, like e-shops, stay a bit conservative to the employment of the new rich client user interface paradigm and keep their classical navigation style and structure with just a few cosmetic improvements enabling just a limited number of user friendly features [12].

978-1-4244-2624-9/08/$25.00 ©2008 IEEE

However there are few superior rich web client applications which allow the possibilities of desktop applications inside the web browser environment.
The great inspiration comes especially from the most advanced Ajax application Microsoft Outlook Web Access. This state of the art feature-rich web client employs the most important functions of desktop version of the product. The application uses the most advanced Ajax techniques to access the Microsoft Exchange Server's user content providing mailing, time, task and contact management capabilities in the way that almost exactly mimics the look and feel of the desktop version.
The online mapping software is one of the most suitable applications to be developed as according to Single page web application design. Though there are more competing developer companies (Google [8], Yahoo [10], and Microsoft [9]) their applications provide almost exactly the same set of functions and features. The user interface design is undeniably inspired by their desktop counterparts. For example Google Maps is the web complement to their desktop rich client application Google Earth.
Another inspiration is the field of the office web applications, which provide the functionality known from desktop application packs such as the Microsoft Office or open source project Open Office. Google Docs [11] enables viewing and editing tools for creating text documents, spreadsheets and presentations in one single page application mimicking the look and feel of well known desktop office applications.
It is not without a remark, that all of the successful feature-rich client web applications have their strong inspiration and counterparts in desktop applications, whether provided by the same developer company or not. It seems that the design of the single page user interface shares the same design principles with the desktop application development. What differs is the actual implementation technology. If we understand the similarities in these concepts, we can find them and describe them in the pattern language.
Patterns and the pattern language were introduced by [3] as a named problem – solution documentations used in the field of architecture. An opening and fundamental work that has introduced the concept of patterns to the field of computer science and software engineering was the book on design patterns [4]. Since then a couple of other books have been written using the pattern language and principles describing pattern in particular areas.
One of the works that brought the inspiration to our research is the book on patterns for user interfaces [5]. The design decisions presented later in our article are trying to bring some additional discussion to the field of the user interface patterns.

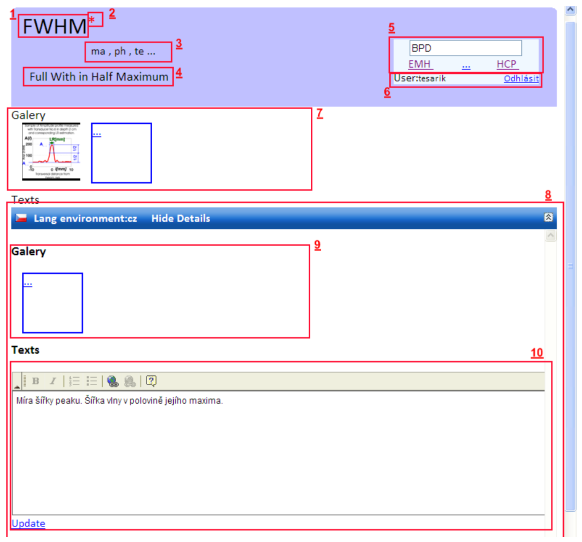
4. Case Study Application

4.1 Application overview

Our team research aims primarily at the work on quality measurement of medical diagnostic and therapeutic ultrasonographic equipment – transducers and scanners [1], [2]. One part of our research also covers the area of custom software applications and tools development. The applications vary from the measuring apparatus control system, the analytic software tools to the database systems. These applications help us to achieve the main goal – ensuring that the quality of healthcare equipment is adequate to be safely used in the medical practice.
One of our minor tasks is to collect and bring together terminology we use and create a dictionary of acronyms, including their meanings and additional information, such as descriptive images. The dictionary is filled in by members of two distinct teams – one at the Palacký University in Olomouc, Czech Republic and the other at the Medical University in Vienna, Austria. Because of the team distance and mutual independence, it is more important to offer an application based on client-server solution which is easy to use and distribute. The web client application with underlying database is the reasonable option, since the runtime environment is a web browser and application data and code are stored and maintained at one place.
The user requirements for the application were quite straightforward: to make a dictionary to add and edit acronym entries. Every dictionary entry should have possibility to attach more than one comment and images. Every user should be able to edit all of the items. Although there are several requirements that go a bit over common tasks to implement using just three tables.
Some of application points of interest are

- **Ambiguous dictionary entries**. Dictionary abbreviation entries may not be unique; there may be more meanings for each entry.
- **Multilingual support.** The descriptions and comments are provided in more languages (currently supports English, Czech, German) **Tags to entries.** Each entry may have category tags attached.
- **Language-dependent image galleries.** Entries have the main illustrative graphics gallery and image galleries for each of the supported languages, in case that the images may differ according to used language.

Though the application business logic is a simple one, the single page design and implementation show that there are more use cases to care about than it was expected. In case the applications are well designed, the use cases manifest themselves as distinct web page components. As seen on Figure 1, the main application components are emphasized.



Míra šířky peaku. Šířka vlny v polovině jejího maxima.

Figure 1. Case study application overview
1- Acronym, 2- Ambiguousness check, 3– tags, 4-meaning, 5- navigation, 6- login, 7- Image gallery, 8-language environment with 9-gallery and 10-descriptions/comments

4.2 Implementation

The web application was developed in Microsoft's *ASP.NET 2.0* and the web pages' code behind is written in C# language. Asynchronous Ajax communication is based on *ASP.NET AJAX*. Certain user interface elements are realized using components from community project *ASP.NET AJAX Control Toolkit*. We also use *FCKEditor* components for writing and editing rich text fields, such as comments. Underlying database runs on the *MS SQL Express* database engine.
Though this article's case study uses enhanced features of ASP.NET AJAX framework with Ajax Control Toolkit component library it is not and was not meant to be a full Ajax web application. It is a mixture of ASP.NET application using a few of Ajax capabilities to reach the main goal – to make application user interface design stay within one single page with minimal effort.
That is why we yield before full Ajax implantation principles according to [6] or [7]. At the current state of available technologies, the full Ajax application development is superfluously expensive. We prefer the server side components architecture instead of client-side JavaScript programming.

5. User interface features and patterns

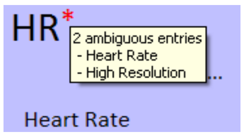
Following high level user interface design principles use similar level of abstraction as the patterns; however they use a loose text structure and do not have the ambition to be handled with such gravity the real user interface patterns have [5].
The next list of patterns describes a few interface design decisions to be made to successfully implement the single web application. Each list item contains the problem-solution description and one or more examples from case study application.
Though the actual implementation of case study application depends on the ASP.NET AJAX environment, the principles themselves are not limited to either single page design or web page design at all. On the contrary, these principles are commonly used in the user interface design of most applications, including desktop or mobile applications via many different implementations.
The main common forces applied to the following patterns are the limited utilizable frame of the single web page and its complement – requesting the rich functionality that keeps or improve application usability to the level known of the desktop solutions.

5.1. Details on demand

Overall small available space of a web page prevents from showing much of detailed information on such a limited area. However, the user must have a chance to obtain desired detailed information without much effort. We need to provide an efficient mechanism to let user to see additional details. A similar pattern is described in [5] as the *Extras on demand*.
Since the single page design prevents to show detailed information on another web page, which is actually very common practice in list–detail control pattern web implementation. It is necessary to use simpler, light weighted components to achieve desired effect. The simplest solution is to use a *Tooltip*, which automatically pops up when the mouse pointer hovers over the target element containing brief information. The tooltip capability is provided by ASP.NET itself. If the detailed information is more complex the more suitable solution is to use a particular component such as *PopupControl* extender (see Ajax Control Toolkit) or another similar component described in [7] as the *Popup* pattern.

5.1.1. Ambiguousness check details

The application needs to store ambiguous dictionary entries, because there may be two or more homographic abbreviations with different meanings. E.g. HR may stand for either Heart Rate or High Resolution. The user needs to be informed whether there is an ambiguous entry. The application displays the big red asterisk at the top right corner of the abbreviation title. The user is also able to check for the different acronym meanings by pointing the mouse cursor over the asterisk and sees popup window containing additional information including the number of ambiguous entries and a list of their meanings.



2 ambiguous entries
- Heart Rate
- High Resolution

Figure 2. Ambiguous entry details.

Actual implementation is to use the *PopupControl* extender, which is part of Ajax control toolkit community project. Popup panel extends the behavior of referenced user interface component (the red asterisk, see Figure 2) and is shown when the referenced control gets the focus.

5.1.2. Tag short forms

Every dictionary entry can be marked with one or many tags. Tags provide additional information on scope of the abbreviation term. A few examples of used tags are: "Medical", "Technical", etc. It's possible to add many of them, so the list could be very long and hard to read. Thus every tag also has its own short form, e.g. "me" for "Medical", "te" for "Technical". However the user might be puzzled by a short version of the tag, detailed information – the full text of a tag is provided as a tooltip, while the mouse cursor hovers over the tag abbreviation.

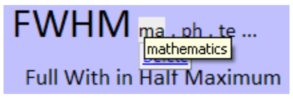


Figure 3. Short forms details using the simple tooltip

5.2. Actions on demand

There is usually more than one item on the page, which can be by some means manipulated – inserted, deleted or edited. For example, tags and gallery images can be added or deleted, the text comments may be added, edited or removed. Having all the available actions for all of the items shown at once could fill the page with too many action buttons making it unable to read.

A possible implementation in single page design is to dynamically show or hide action panels using hover popup action menus. Popup menus add behavior similar to popular context menus known from common windows applications. They act in similar way to previously mentioned Tooltip (5.1.1). The main differences are that they contain interactive UI elements, such as buttons instead of presentation data and the user is able to enter into the popup control's content with mouse cursor, which is not possible in the case of the Tooltip.

5.2.1. Popup Delete action buttons on tags and gallery items

Either gallery images or tags can be simply removed using action button only available on mouse hovering over the selected item (Fig. 4).

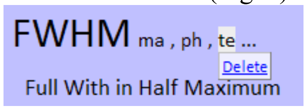


Figure 4. Pop-up delete button on the tag

The technical solution is the same as for the example with ambiguous entry details. *HoverMenu* extender can contain any other ASP.NET elements, in this case a command button.

5.2.2. Adding tags from list of available items

The user is able to add additional tags to decorate a dictionary entry. Holding the mouse cursor over an empty list item (shown as "[…]") brings up a menu containing all tags that can be attached to the current dictionary entry. Each tag can be attached to one dictionary entry only once; hence the menu contains no more than unused tags to select from.

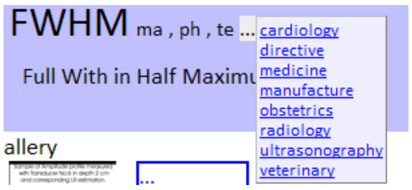


Figure 5. Adding a tag. Dynamic menu contains list of available tags.

The technical solution is similar to the ambiguous entry details example. Hover Menu's content control is dynamically filled from a database with a list of insert command buttons for each available tag (Figure 6).

5.3. User driven page components visibility

Not every piece of information needs to be shown all the time. It is wise to let the users to suppress unwanted page parts and focus their attention just to areas they need to work with. The user have a chance to save page space by hiding non-essential parts leaving the option to show the hidden parts back again. One of this techniques is described in [5] as the *Closable panels*.

The solution is to organize user controls to logical blocks that can be independently switched on or off.

5.3.1. Collapsible language environments

Application allows entering text comments and graphical information that may vary depending on a particular language. Application currently supports up to three (Czech, English, German) language environments. Language environments are containers for language specific pictures and texts. The user may choose which language environment details to show or hide. For example, while entering Czech text comments the user wants to suppress English and German environments to save some page space.

The technical solution is to use specialized component such as the *CollapsiblePanel*. Collapsible panels decorate the target control with appearance and behavior similar to maximization – minimization buttons in frame windows. Such a component allows toggling embedded component's state to full shown or non visible, displaying the frame with state information and toggle action buttons.

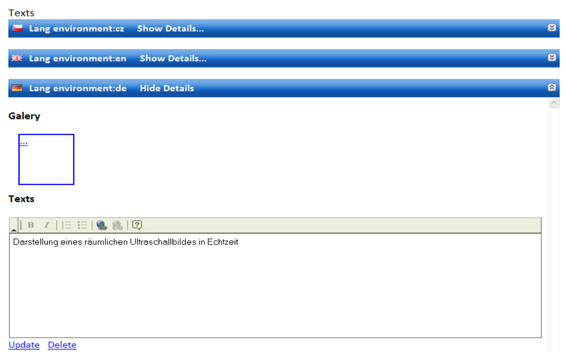

Texts
Lang environment:cz Show Details...
Lang environment:en Show Details...
Lang environment:de Hide Details
Galery
Texts
Update Delete
Darstellung eines räumlichen Ultraschallbildes in Echtzeit

Figure 6. Collapsible language environments. There are two collapsed controls and one opened.

5.4. Dedicated mode

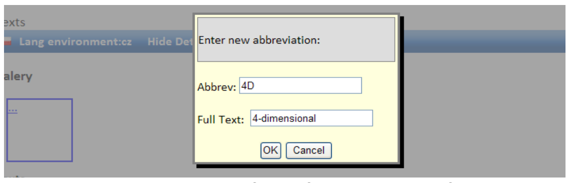
Though the single page application is designed to be easy and effective to use, there are several actions that require more user attention or inputs, which cannot be entered just by simple clicking on one of action buttons. Adding a new item is generally one of these actions that require more user textual input and more complicated interface.

The multi page application solution is to create an input form on a separate web page. Single page design replaces this technique by emulating page navigation to a new page by showing modal form. Modal forms are commonly used in desktop applications to show current important messages requiring user attention such as message boxes or forms for user inputs (modal dialogs).

Another common task is to ask the user for confirmation of a destructive action such as massive change of data or deleting items. A simple solution is to interlink action button with the extender to show confirmation dialog.

5.4.1. New dictionary entry form

Adding a new dictionary entry requires entering two texts, abbreviation and its meaning. The insert form containing two text boxes is not present inside the main page all the time, but it is shown only on demand after processing the insert item action. Modal form makes the parent page disabled to user interaction, showing only the necessary insert form elements. User fills in form data and the modal form hides itself and returns focus back to the parent page after the user input confirmation or cancellation.



Enter new abbreviation:

Abbrev: 4D

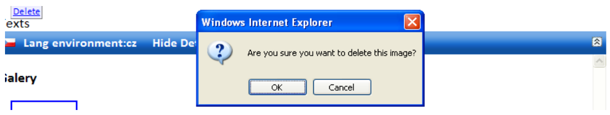
Full Text: 4-dimensional

Figure 7. The modal form for insertion of a new dictionary entry.

Technical solution is to use the functionality of *ModalPopup* component. The Modal Popup works in similar fashion as the Collapsible Panel (5.3.1). Target user input form is extended by the additional behavior to be shown as a modal form or hidden.

5.4.2. Delete confirmation dialog

It is a good manner to show user confirmation dialog on changing or deleting items which require some user effort to enter and forbid unwanted deletion (e.g. a miss-click on delete action button). Deleting the gallery items or text comments is such a case.



Are you sure you want to delete this image?

Figure 8. Browser driven message box waits for the user input.

Use the *ConfirmButton* extender or just the simple client side JavaScript command to show the operation system's message box containing OK - Cancel options (Figure 9).

It is also possible to create our own confirmation dialog using Modal Popup (see 5.4.1) in case that we need to show more detailed information about the confirmation, like the deleted item details or a disclaimer. In this case there is no need to show such a detailed text we can use the built in solution.

5.5. Multifunction controls

The design and development process is a dynamic one, there are still things to change or add, new requirements or ideas come and we have to handle them, choose the right ones and put them into our design.

Due to the lack of free space and restricted component sizes it is hard to add more of the originally planned or newly requested control functionalities into the limited boundaries of the control. The conflict between simple design and rich user interface capabilities is omnipresent and there is not a simple solution to find the right balance.

While the functionalities stay simple, disjoint and their added value is bigger than the cost of their integration into the current design we may just simply add them to the application. But we might also consider the fact if there is a possibility to not just simply add the functionalities but to blend them together. We may use the overlapping between given functionalities and mix them together into a more sophisticated result.

5.5.1. Previous – Next item navigation

The simplest and commonly practiced way to blend two functionalities is presented on the basic navigation control. There are two action buttons to navigate to previous and next directory item. The added value is that the action buttons also show the names of these items (Figure 10).

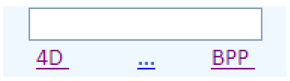


Figure 9. Previous and next navigation links show the dictionary entry names.

This practice of providing the additional useful information whenever possible is a big helper to the user and its usability should be considered in the design of all user interface control elements. On the other hand, it is also good to decide whether the additional information is actually helpful or if it is not just placeholder which could be used better. In such a case there is always the choice to combine this principle with the previously mentioned "Details on demand" pattern (5.1).

5.5.2 Auto-complete, search and index

Since there is only one abbreviation shown at time, the user needs to navigate to other ones. The simplest navigation to the previous and the next dictionary item is a necessary but not sufficient way, because of the dictionary is growing and may contain many entries. We have to provide a more efficient way of navigation between entries (e.g. let the user start navigating from selected entry or at least starting character). It is also important to add some search capabilities and help user to choose the right item from an index.

We can have all of these multiple features in one solution using the AutoComplete functionality. The AutoComplete control extends the behavior of simple text inputs and enhances them with a possibility to choose the text from a list of items, giving them the desktop combo-box look and feel. The list of items is dynamically created using Ajax communication with the specified server side web service method. The web method accepts the partially written text and returns a collection of texts, which usually complete the unfinished word.

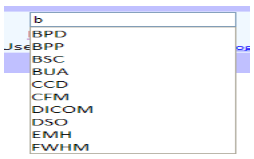


Figure 10. Auto complete capability serving also as the dictionary index and helping the navigation

As seen on the Figure 11, the implemented auto-complete control works as both the combo box and dictionary index giving the user the possibility to jump directly to the given entry. We use the web method that has a bit uncommon but efficient implementation that enables not just the auto-complete capability, but also shows the following extra index items to choose from. This implementation allows blending the auto-complete functionality, dictionary index and search into one single place.

6. Conclusion

In our article we listed several web page design level solutions leading to the successful implementation of single page Ajax web application. The case study of simple yet not trivial application shows some of common features that make it possible to effectively design a feature-rich single page web application.

The main force applied to the patterns was the limiting space of the single web page. Presented patterns allow achieving the requested level of user interface functionality according to the application usability of the desktop solutions.

While the necessary development tools are not yet optimal and the development of such an application is not problem-proof, following the above mentioned design principles, with a help of modern web infrastructure components, make the creation of such a simple but feature-rich application possible with reasonable amount of effort.

7. References

[1] Doležal, L., Kollmann, Ch., Hálek, J., Smolan, S., "An Image Quality Parameter Measured in Whole Scanned Area", *Ultraschall in der Medizin 23*, Thieme, 2002, p. 69

[2] Doležal, L., Mazura, J., Tesařík, J., Kolářová, H., Hálek, J., "Derivation of sonograph quality parameters by the use of Point Spread Function analysis.", *Physiological Research 56 (Suppl. 1)*, Prague, 2007

[3] Alexander, Ch. et al., *A Pattern Language: Towns, Buildings, Construction*, Oxford Press, USA, 1977

[4] Gamma, E. et al., *Design Patterns*, Adison – Wesley, 1995

[5] Tidwell, J., *Designing Interfaces: Patterns for Effective Interaction Design*, O'Reilly, 2005

[6] Crane, D., et al., *Ajax in Action*, Manning, 2006

[7] Mahemoff, M. *Ajax Design Patterns*, O'Reilly, 2006

[8] Google Maps, http://maps.google.com, [4-15-2008]

[9] Live Search Maps, http://maps.live.com, [4-15-2008]

[10] Yahoo! Maps, Driving Directions, And Trafic, http://maps.yahoo.com, [4-15-2008]

[11] Google Docs, http://docs.google.com, [4-15-2008]

[12] Amazon.com, http://www.amazon.com, [4-15-2008]