Проучване и избор на подход по повод нова версия на софтуер ProsTE

Версия 1.0

|  |  |  |  |
| --- | --- | --- | --- |
| Версия | Дата | Автор | Описание |
| 1.0 | 14.03.2022 | Александър Константинов | Създаване |

Съдържание

[0. Нови моменти в подхода на работа на ProsTE 2](#_Toc98188087)

[1. Динамичен интерфейс 2](#_Toc98188088)

[2. Изработване на нов начин на конфигуриране на входно изходните процеси в система ПРОСТЕ, така че да се минимизира техния обем и да се създаде възможност за цикъл 5](#_Toc98188089)

[3. Диагностициране и определяне наживо на устройствата закачени към компютъра на инженера 5](#_Toc98188090)

[4. Възможност за сложни валидации, чрез конкретен код 6](#_Toc98188091)

[5. Ъпдейт без тотална прекомпилация 6](#_Toc98188092)

[6. Интерфейсът на системата да е от типа white label за да може да се конфигурира 6](#_Toc98188093)

[7. Преводите да се изнесат в нов конфигурационен файл, който подлежи на ъпдейт без прекомпилиране 6](#_Toc98188094)

# 0. Нови моменти в подхода на работа на ProsTE

Въвеждаме външна библиотека за работа с всяко едно от устройствата, която наричаме библиотека DEV последвана от име на устройството

Всяка една от тези библиотеки е външен файл за ProsTe и подлежи на индивидуален ъпдейт. Неизчерпателно тя предоставя следните функции:

* Канали за връзка с устройството
* Тестова функция, която по зададените канали установява дали и къде има закачено устройство
* Набор от специфични функции – например възможност за сложни валидации, чрез конкретен код

Въвеждаме следната структура на приложението

* Система ProsTE - напълно независима от устройства, езици и външен вид. Неговата функция е интерпретатор на външния вид, интерфесен интерпретатор и интерпретатор на входно изходни операции
  + Елементи за външния види сървър за ъпдейт за конкретната реализация на системата
  + Файл с налични пакети от DEV библиотеки, конфигурационни файлове, дизайн файлове и преводи за всяко устройство и техните версии
  + Набор от DEV библиотеки, конфигурационни файлове, дизайн файлове и преводи за всяко устройство
* Изпълним файл за конфигуруем ъпдейт на системата

# 1. Динамичен интерфейс

Към настоящия документ са приложени 2 c# проекта които нагледно демонстрират 2 подхода за динамичен интерфейс на системата.

**Подход 1**

Интерфейсът за визуализация на конкретно меню или редакция на конкретна част от паметта на произволно устройство се реализира с помощта на eXtensible Application Markup Language – XAML. <https://docs.microsoft.com/en-us/dotnet/desktop/wpf/xaml/?view=netdesktop-6.0>

В примерното приложение **WpfAppProtot2** са демонстрирани следните възможности:

Бутон **Browse** инициира зареждане на външен XAML файл. Примерен файл се съдържа в директория **html and xaml files**. Този файл е в комплект с cs файл със същото име. В него има за нагледност:

* Текстово поле
* Падащо меню
* Чекбокс
* Бутон

Зареждането става в празен до момента панел **XAMLContainer.** На този принцип могат да се зареждяат множество екрани, които да представят по красив и модерен начин отделните менюта и части от паметта на конкретно устройство. XAML файлът може да бъде стилизиран от дизайнер с умения за работа с XAML.

Събитието **LoadCompleted** е прихванато в системата като в него се намира динамично създадения бутон и неговото натискане води до изпълнение на реален метод от системата. В реалната експлоатация, този метод следва да бъде обозначен от специален таг на елемента и да се изпълнява динамично, а не статично

Бутон **Load** извършва инициализация на визуалните елементи от XAML файла с данни. Инициализацията се извършва чрез итериране по всеки един от елементите, проверка дали за него има зададена стойност и инициализацията на тази стойност по подходящ за елемента начин:

* За Текстово поле – като текст
* За Падащо меню – като стойност
* За Чекбокс – като отметната или не

На практика този подход позволява разширяване на визуалните елементи при необходимост

Бутон **Save** показва зареждане на въведените стойности в динамичните елементи. Зареждането се извършва чрез итериране по всеки един от елементите и зареждане на неговата текуща стойност по подходящ за елемента начин:

* За Текстово поле – като текст
* За Падащо меню – като стойност
* За Чекбокс – като отметната или не

Отново този подход позволява разширяване на визуалните елементи при необходимост

**Подход 2**

Интерфейсът за визуализация на конкретно меню или редакция на конкретна част от паметта на произволно устройство се реализира с помощта на HyperText Markup Language - HTML

В примерното приложение **WpfAppProtot** са демонстрирани следните възможности:

Бутон **Browse** инициира зареждане на външен HTML файл. Примерен файл се съдържа в директория **html and xaml files**. В него има за нагледност:

* Текстово поле
* Падащо меню
* Чекбокс
* Бутон

Зареждането става в празен до момента панел **WebContainer**. На този принцип могат да се зареждяат множество екрани, които да представят по красив и модерен начин отделните менюта и части от паметта на конкретно устройство. HTML файлът може да бъде стилизиран от web дизайнер.

Събитието **LoadCompleted** е прихванато в системата. В него се прихваща кликване върху елемент на документа и неговото натискане води до изпълнение на реален метод от системата. В реалната експлоатация, съответния бутон следва да има собствени атрибути, които указват какво точно се случва при натискането му.

Бутон **Load** извършва инициализация на визуалните елементи от ХТМЛ файла с данни. Инициализацията се извършва чрез итериране по всеки един от елементите, проверка дали за него има зададена стойност и инициализацията на тази стойност по подходящ за елемента начин:

* За Текстово поле – като текст
* За Падащо меню – като стойност
* За Чекбокс – като отметната или не

На практика този подход позволява разширяване на визуалните елементи при необходимост

Бутон **Save** показва зареждане на въведените стойности в динамичните елементи. Зареждането се извършва чрез итериране по всеки един от елементите и зареждане на неговата текуща стойност по подходящ за елемента начин:

* За Текстово поле – като текст
* За Падащо меню – като стойност
* За Чекбокс – като отметната или не

Отново този подход позволява разширяване на визуалните елементи при необходимост

Като реализирахме и анализирахме Подход 1 и Подход 2 правим следното обосновано заключение:

Препоръчваме Подход 2 – използване на HTML външни файлове защото:

* На пазара на труда е много по-лесно и евтино да се намери HTML дизайнер отколкото XAML дизайнер
* HTML е описателен език, който може да бъде зареден в много среди и операционни системи, както и чрез много технологии. XAML е много по-ограничен откъм разпространение и може да затрудни евентуално бъдещо развитие на системата в други ОС

# 2. Изработване на нов начин на конфигуриране на входно изходните процеси в система ПРОСТЕ, така че да се минимизира техния обем и да се създаде възможност за цикъл

Както е казано вс заглавието на точката, се създава възможност за цикъл. По този начин се минимизира описанието на входно изходни операции до една вложена в цикъл. Освен това всеки елемент от входно/изходните операции се снабдява с допълнителен елемент описващ за коя версия на устройството е валиден. По този начин се премахва необходимостта от създаване на множество файлове описващи входно/изходните процеси за различните версии на фърмуеъра

# 3. Диагностициране и определяне наживо на устройствата закачени към компютъра на инженера

Диагностициране и живо определяне на закачените устройства се извършва, като за всяка една от библиотеките DEV+ име на устройството се прави проверка и при установяване на свързаност се изчитат входните параметри на устройството

# 4. Възможност за сложни валидации, чрез конкретен код

Валидациите, когато те са сложни, се публикуват от сътоветната библиотека на устройството. Преди запис се извикват всички функции, които са дефинирани като валидиращи за съответната библиотека и се проверява дали те „минават“

# 5. Ъпдейт без тотална прекомпилация

Както бе обяснено в т.0 системата разполага с изпълним файл за ъпдейт. Като негов външен параметър се подава какво точно следва да бъде свалено от сървъра – библиотека(и) и файлове за конкретно устройство или цялата система. Съгласно инструкцията се свалят необходимите елементи и се подменят, като след това се опреснява файла с текущата актуалност на системата

# 6. Интерфейсът на системата да е от типа white label за да може да се конфигурира

На практика системата няма собствен интерфейс, а само разположение на интерфейсни елементи. Всички тези елементи са въшни файлове, които са в комплект със системата. По този начин лесно може да се промени визията ѝ

# 7. Преводите да се изнесат в нов конфигурационен файл, който подлежи на ъпдейт без прекомпилиране

За всяко едно устройство, както и за система ProsTE е наличен json файл със следна структура

{

<име на ключ>:{

<Код на език>:<превод>

}

..

..

..

<име на ключ>:{

<Код на език>:<превод>

}

}

Чрез тази структура се прави превод на всеки интерфейсен елемент. Респективно в описанието на интерфейсът се използват САМО ключове от преводния файл, а не конкретни текстове