**Програмиране за мобилни устройствa**

**1. Особености при създаване на приложения за мобилни устройства.**

Всеки мобилен браузър поддържа някои форми на HTML.Много от тях,особено от висок клас устройства като iPhone и Windows Phone 7, поддържат последния HTML5,CSS и JavaScript стандарти и правят перфектни копия на това което ще видите в традиционен компютърен браузър.Вашият най-евтин вариант за подкрепа на мобилни браузъри е да не правите нищо.Изборът на тази опция води до много лошо мобилно браузване, но няколко причини:

1. Най-очевидната разлика между компютър и мобилен телефон е размера на дисплея.Средният размер на компютърен монитор е 21 инча с резолюция на дисплея не по малка от 1024 х 768.Стандартният смарт телефон е с дисплеи от 3,9 инча и резолюция по-малка от 800 х 480.Мобилният телфон не ви дава много място с което да работите.Необходимо е да използва по-малко текст и по добри икони.

Някои мобилни баузъри като Опера мини се справят с десктоп-ширината на страниците като динамично преформатират оформлението настраницата и нейните стилове.Полученият вид рядко е това, което дизайнерът е имал в предвид.Други мобилни браузъри като Сафари за iPhone или Internet Explorer за Windows Phone 7,рендват страниците с нормалните им размери и след това принуждават потребителя да увеличава на определени места и да движи уголемената страница за да прочете текста.

Ето какво трябва да запомните за големината на дисплея:

* Не разполагате с много място за работа, така че намалете екрана и опциите, също така минимизирайте размера на текста.
* Текстът който използвате трябва да бъде четлив и разбираем
* Заменете текста с икони там където е възможно
* Иконите ви трябва да изглеждат колкото се може по реалистични
* Графиките трябва да съответстват на задачите и дизайнът трябва да е ясен

Стандартният компютър има няколко начина за въвеждане на информция, но основно се използват клавиатурата и мишката.Мобилното у-во разчита главно на пръстите,като те се ползват за въвеждане на екранната клавиатура,докосване на икони и бутони, или за жестове като приплъзване или прищипване.За да компенсирате тези недостатъци, моете да предоставите звук и вибрации при натисканията на клавишите и минимизация при въвеждане на текст.Пръстите могат да са всякакви размери бутоните трябва да са достатъчно големи (35 пикселов квадрат) и визуално отделени от другите бутони или обекти за да се намелят грешките.Намерете нови начини за въвеждане на данни, като например използването на снимки.Също така трябва да се възползвате от хардуера.Windows Phone има 3 специални бутона – Back, Start, Search може да ги използвате за да поддържате реда в интерфейса и да намалите възможностите за грешка.

**2. Разпознаване на browser. Управление на viewport.**

**2.1.**Разпознаване на браузъра представлява процес, който най-често има за цел установяване на типа и/или версията на потребителския браузър.Осъществява се чрез код изпълняван от сървъра или от самия браузър.В по-широк смисъл този процес има за цел да установи наличието или липсата на определени(от програмиста) възможности на браузъра.

**2.2.** Можете да разберете дали някой посетител ползва мобилен браузър със с-вото на ASP.NET – Request.Browser.IsMobileDevice. ASP.NET определя какъв браузър прави заявка и какви са неговите възможности (големина на екрана, подръжка на JavaScript и т.н. ) като сравнява входящата заявка от главните низове userAgent с поредица от регулярни изрази в XML файлове които описват общи браузъри.Информацията за съответните възможности на у-вото се съхранява в директорията : C:\Windows\Microsoft.NET\Framework\v4.0.30319\Config\Browsers

(или еквивалентна в зависимост от инсталацията ви).Например стандартния iPhone браузър файл включва кода:

*<browsers>*

*<!-- Mozilla/5.0 (iPhone; U; CPU like Mac OS X; en) AppleWebKit/420+ (KHTML, like Gecko) Version/3.0 Mobile/1A543a Safari/419.3 -->*

*<gateway id="IPhone" parentID="Safari">*

*<identification>*

*<userAgent match="iPhone" />*

*</identification>*

*<capabilities>*

*<capability name="mobileDeviceModel" value="IPhone" />*

*<capability name="mobileDeviceManufacturer" value="Apple" />*

*<capability name="isMobileDevice" value="true" />*

*<capability name="canInitiateVoiceCall" value="true" />*

*</capabilities>*

*</gateway> ... </browsers>*

The following element defines the expression to be matched against incoming userAgent header strings: <userAgent match="iPhone" /> Once the system finds a matching userAgent expression, the remainder of the XML data specifies the type and capabilities of that device. *( Unfortunately, this does not include common modern browsers such as Opera Mobile or the default browser for Google Android. Request.Browser.IsMobileDevice will incorrectly be set to false )*

Със следните 2 възможности можете да подобрите разпознаването на браузър:

1.Можете да предоставяте свой собствени .browser файлове да представляват по-новите у-ва

2.Можете да използвате чужда библиотека за разпознаване на браузъри

**Viewport**

Много мобилни браузъри включително Safari за iOS и Internet Explorer за Windows Phone 7, се опитват да направят заредените страници да изглеждат точно както го правят на настолен браузър.Те знаят че повечето страници са предназначени за екрани около 1000 пиксела широки и дизайнерът най-вероятно не е пригодил страницата за по малки екрани.За да разрешат този проблем обикновено те зареждат страницата на виртуално платно познато като „viewport“ обикновено около 1000 виртуални пиксела.Браузърът може да мащабира визуалния дисплей на виртуалното платно произволно, което позволява на потребителя да увеличава и движи изобразеното.

Много от най-популярните мобилни браузъри поддържат нестандартен „viewport“ мета таг, който позволява да се управлява широчината на виртуалния viewport.Например ако се добави следното към <head> отделението на страница браузърът ще разположи страницата на viewport който е 320 пиксела широк:

<meta name="viewport" content="width=320"/>

Това обикновено е много по-добър избор за мобилни телефони.Трябва да се има в предвид че има телефони с много по-висока хоризонтална резолюция.Например iPhone 4 има 640 физически пиксела на ред.Въпреки това все още има смисъл да се използва виртуален viewport от около 320 пиксела, в противен случай полученият текст ще бъде твърде малък за да се чете добре без приближаване на страницата.Може да се позволи на виртуалния viewport да варира в размер подходящ за устройството което се изполва, като добавим следния синтаксис:

<meta name="viewport" content="width=device-width"/>

Note that some mobile devices won’t give you a literal device width. They interpret “device-width” as meaning “the virtual viewport width that the manufacturer thinks gives the most pleasing result.” So, for example, iPhone 4 defines device-width as 320 pixels, despite its higher physical resolution.

**3. Препоръки по структуриране на markup текста. Софтуерна архитектура на мобилни приложения.**

Препоръки по структуриране на markup текста

Всеки път, когато проектирате страници за мобилни браузъри:

1. Използвайте viewport мета таг, за да направите така че viewport да пасне на хоризонталната ширина на екрана.

2. Настройте оформлението на страниците, CSS стиловете за ограничената ширина.Ако посетителите не трябва да увеличават или да превъртат хоризонтално, страницата ви ще бъде по скоро като приложение създадено специално за тяхното устройство което гарантира по-добро впечатление.

3. Уверете се че връзките и бутоните са достатъчно големи, за да бъдат натискани без усилия. Върховете на пръстите са много по-големи отколкото върха на курсора на мишката.

4. Намаляване на изискванията към пропускателната способност, като не се използват много изображения с висока резолюция или огромни JavaScript файлове.

**4. Създаване на приложения за Windows Mobile с Visual Studio Window. История. Основни tools. Емулация. GPS емулация.**

История

Visual Studio поддържа Windows Mobile разработка на приложения от Visual Studio 5.0.

Visual Studio 2005 поддържа разработката на приложения за Smartphone 2003, Pocket PC 2003 SE, и Windows CE.

Лесно можете да добавите поддръжка за Windows Mobile 5.0 и Windows Mobile 6(Standard, Classic and Professional) като инсталирате SDKs(Software development kits) за тези платформи.SDKs включват различни инструменти, които значително подобряват процеса на разработка, емулатори, GPS, мобилен оператор емулатор и др.Visual Studio 2008 донесе няколко нови възможности и функции, както в IDE така и във функционалността.

Списъкът по долу показва някои нови функции във Visual Studio 2008:

-Поддържа тестване на елемент във Visual Studio Developer Edition и Visual Studio Team Suite – Remote Performance Monitor

-Диспечер на устройства за сигурност, част от Visual Studio 2008 IDE

-Нови Device Emulators и Device Emulator Manager

-Нов дизайн на New Project Wizard за създаване на нови Windows Mobile проекти

Основни инструменти

.NET CF (compact framework) е по-малък от.NET Framework, и предоставя подмножество на .NET Framework функционалността. Ето защо,повечето приложения, които се разработват за .NET CF би трябвало да работят без никакви модификации на работния плот на.NET Framework платформа.

Има две основни причини за намалената функционалност при .NET CF:

- Неподдържани десктоп функции

- Ограничена памет на мобилните устройства

Някои нови контроли са добавени и подобрени:

- Месечен календар.

Значително подобрен. - DateTimePicker.

New in .NET CF 2.0:

- DataGrid.

- LinkLabel.

- Splitter.

- WebBrowser.

Емулации

-Device Emulator Manager – управлява емулаторите на устройството, които са инсталирани на работната станция на разработчика..

**Мрежова връзка в Device Emulator**

За да се симулира реална среда, понякога трябва да има връзка с мрежата в Device Emulator.За да се направи това, трябва да се създаде приемаща мрежа, към която искате да се свържете в емулатор.

- Cellular Emulator е част от Windows Mobile SDK Tool.Той симулира присъствието на мрежа на мобилен оператор, която дава възможност на разработчиците да тестват функционалността на телефона чрез техните приложения от Device Emulator.

Fake GPS

Fake GPS позволява използването на GPS функционалностите от Device Emulators или от физическите у-ва, които не разполагат с GPS, или за тестване на приложения при които не е нужно GPS покритие.

**5. Особености при създаване на потребителския интерфейс. Работа с данни.**

Работа с данни

.NET CF поддържа работа с различни източници на данни.Концепцията е същата като при .NET Framework и най-важното е че ADO.NET се поддържа в по-голямата си функционалност.Windows Mobile приложенията могат да използват SQL Server CE, XML, SQL Server, уеб услуги като източници на данни. XML е общ формат, при който повечето източници на данни могат да внасят и изнасят данни.Въпреки че може да се използва от повечето източници на данни, не се препоръчва като средство за съхраняване на данни, защото има голям излишък, не е оптимизиран и е много по-бавен от релационна база данни, защото няма индекси.Въпреки това, ако няма друга възможност XML може да се ползва за съхранение на данни на у-вото.SQL Server може директно да бъде достъпен като средство за съхраняване на данни.Уеб услугите са добър избор за прехвърляне на данни в n-редна среда, когато няма директен достъп до SQL Server (за директен достъп до данни или за копиране).

**6. Пакетиране на готово приложение и инсталация.**

Последният етап от цикъла на развитие на проекта е да се разположи цялостния код на у-вото и да бъде оформен като цялостен пакет за дистрибуция и масово разпространение.Има няколко начина по които приложението може да бъде разположено на у-вото или емулатора:

**Development IDE installer;**

Най-лесният начин за разполагане на приложение е когато у-вото е свързано с разработващата работна станция чрез ActiveSync или WMDC.Първата стъпка е да се избере на устройство или емулатор ще се разполага даденото приложение.Когато направите избор може да свържете избраното у-во ръчно от лентата с инструменти или процеса на разполагане ще го направи автоматично.

**Device Installer;**

Не е много вероятно всяко у-во което се използва в производството да пристига при разработчика за разработка на приложение.Device Installer е метод който позволява приложенията да бъдат лесно разположени директно на у-вото, без да е необходима връзка м-у у-вото и работната станция на разработчика.Всички възли и зависимости са опаковани в един (или повече) CAB файл който се копира на у-вото и се изпълнява там.CAB изпълнението на у-вото се инсталира с предварително зададени параметри.

**7. Някои основни услуги (софтуерен аспект): обектен модел на outlook (POOM), изпращане на e-mail, SMS, телефония.**

**Pocket outlook object model (POOM)**

The POOM API е част от библиотеката Microsoft.WindowsMobile.PocketOutlook.Чрез POOM, можете да получите достъп до календара, контактите и задачите на у-вото.

Всички дейности, които използват POOM функционалност са базирани на класа OutlookSession.

Заради това винаги трябва да имате копие на този обект във вашето приложение.

**E-mail**

E-mail функционалността е достъпна през обекта EmailMessage.Той съдържа всички стандартни с-ва които e-mail съобщенията обикновено притежават: Подател, Получател, Прикачени файлове, Тема, Важност, Текст и т.н. Със следващият пример ще покажем как да създадете ново съобщение,да нагласите с-вата му и да изпратите мейла на предварително избран контакт.

**EmailMessage email = new EmailMessage();**

**email.To.Add(new Recipient(dialog.SelectedContact.Email1Address));**

**email.Subject = textBox1.text;**

**email.BodyText = textBox2.text;**

**email.Send("ActiveSync");**

**SMS**

Работата с SMS прилича на работата с e-mail.Освен че притежава по-малко с-ва, главната разлика е, че когато се извика Send метод съобщението се изпраща веднага, докато e-mail съобщението изчаква до следващата синхронизация.Следващия пример показва как се създава и изпраща нов SMS

**SmsMessage sms = new SmsMessage();**

**sms.To.Add(new Recipient("123456"));**

**sms.Body = "WP\_FirstApp SMS message";**

**sms.Send();**

**Телефония**

Осъществяването на гласови повиквания от приложения позволява да се използва функционалността на телефона директно от приложението, без да се налага да се използва Device Dialer. Класът Phone е в библиотеката Microsoft.WindowsMobile.Telephony.За установяване на гласови повиквания се използва класа Phone.Следния примерен код показва как да се постигнат гласови повиквания чрез контакта който е избран предварително.

**using Microsoft.WindowsMobile.Telephony;**

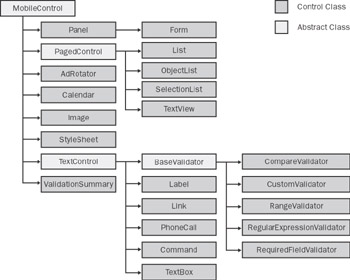
**private void MakeVoiceCall()**

**{**

**Phone phone = new Phone();**

**phone.Talk(dialog.SelectedContact.BusinessTelephoneNumber);**

**}**

**8. Mobile Internet Toolkit (.NET) стандартни контроли и йерархия.   
**

**9. Структура на приложение. Основни файлове и изчислителен процес.**

Разработчик, работещ с обектно-ориентиран език като C# или C++ дефинира програмните класове с модули. .aspx файлът е мястото за дефиницията на клас, който е специфичен за ASP.NET. В мобилните уеб приложения този клас трябва да е наследил(директно или индиректно) System.Web.UI.MobileControls.MobilePage класа. Страницата на една мобилна уеб форма се декларира като наследник на MobilePage класа чрез @Page директивите в началото на .aspx файла, който може да изглежда по следния начин:

<%@ Page Inherits="System.Web.UI.MobileControls.MobilePage" Language="c#"%>

<%@ Register TagPrefix="mobile" Namespace="System.Web.UI.MobileControls"

Assembly="System.Web.Mobile" %>

Ако има code-behind(?) модул, тази страница трябва да наследи клас от този модул, който от своя страна трябва да е наследник на MobilePage класа(sry). В такива случаи @Page директивата трябва да уточни името на code-behind модула и на неговия клас: <%@ Page Codebehind="Default.aspx.cs" Language="c#" Inherits="MyMobileWebForm%>

**10. Контейнери. Пример на приложение с контейнери.**

****

Всички мобилни уеб форми( страници )произтичат от MobilePage клас, който от своя страна произлиза от Page клас ASP.NET. Ето защо, всяка мобилина уеб форма (страница) е валиден MobilePage обект. Всеки MobilePage обект трябва да съдържа една или повече контроли на форми.

**11. Управление на състоянието при мобилни приложения с разкъсване на сесия и stateless обмен. Методи за запазване на сесийното състояние.**

Когато създавате динамични уеб приложения, обикновено се нуждаят от механизъм за съхраняване на информация между клиентски заявки и сървърни отговори. За съжаление, Hypertext Transfer Protocol (HTTP) е ефективно без установено състояние, което означава, че трябва да поддържа състояние по някакъв друг начин.  
  
Следват четири значими методи на ASP.NET предлагани за запазване на състояние:  
1. Сесийно състояние - позволява да се поддържа променливи и обекти за клиент през множество заявки и отговори.  
2. Скрити променливи - Позволява да се задържат обекти между сървъра индивидуални пътувания чрез публикуване на данните на клиента, както скрити полета.  
3. View състояние - позволява да се поддържа стойностите на мобилни уеб форми страници на сървъра. По време на работа съхранява тази информация в инстанция на StateBag клас, който от своя страна се съхранява в рамките на сесията.  
4. Състояние на приложението - позволява да се поддържа променливи и обекти на приложението по няколко заявки от множество клиенти.  
  
**Управление на сесийно състояние**  
  
 ASP.NET предлага осъвременена и подобрена версия на сесийния обект. Този обект позволява да изпълнят следните задачи:  
- Съхранява информация специфична за потребителската сесия.  
- Управлява сесийния живот през eventHandler и методи.  
- Освобождаване сесийните данни след определено изчакване.  
  
Сесииното свойство на System.Web.HttpApplication класа (родителския клас  
на Global.asax страница) и сесийното свойство на MobilePage клас  
(родителския клас на мобилния си уеб форми( страница))  
така ще се даде достъп до сесийния обект. Обикновено, вие ще манипулирате обекта Session или в:  
1. code-behind module на Global.asax файл вашето приложение или  
2. code-behind module на вашия мобилен уеб форми (страница).

**12. Мобилни прил. и Cookies. Скрити променливи. View State и запазване на състоянието.**

**Работа с Cookies**  
Cookies предоставят безценен начин за уеб сървъри, за да идентифицира свързани клиенти, като десктоп браузъри HTML. Въпреки това, потенциалът на Cookies е ограничен по отношение на заявленията за безжични клиенти. Това е така, защото много от безжични устройства, включително и някои Wireless Application Protocol (WAP) и I-режим устройства, не поддържат бисквитки.  
Ако знаете, че вашите целеви устройства поддържат бисквитки или прокси ги поддържа от името на клиента, какъвто е случаят с някои портали WAP, бисквитките предоставят отличен начин за проследяване и идентифициране на сесии.  
  
**Скрити Променливи**  
 Понякога може да искате да подадете малки количества информация между уеб страници, без да се използва сесийно състояние. Да предположим например, трябва да съберете информация от вид, който потребителят попълва иначе в HTML, щеше да предаде информацията от една страница към друга използвайки входни тагове с типа стойността на скрит. HiddenVariables свойството на MobilePage предоставя този вид функционалност. Това свойство ви позволява да съхранявате име на променлива-стойност, които по време на работа след това преминава назад и напред между сървъра и клиента като скрити полета. Вие трябва да използвате HiddenVariables само за малки количества информация. Пример: ... HiddenVariables.Add (TextBoxName.ID, TextBoxName.Text);  
  
**View състояние**  
 ASP.NET дава на потребителя впечатление, че по време на работа поддържа страници в над няколко сървъра „индивидуални пътувания”. Страниците наистина не съществуват над множество запитвания и отговори, вместо това, по време на работа се запазват свойствата на страница в сървър контрол view състояние(инстанция на StateBag клас). Когато потребителят прави искане, по време на работа автоматично възстановява страницата с помощта на стойността на свойството се пази в StateBag инстанция.

**13. Ниво-приложение и методи за запазване на състоянието.**

Application State

В ASP.NET, заявление е общата сума на всички файлове, на които по време на работа може да извика или изпълни в обхвата на виртуална директория и всички нейни поддиректории. От време на време, може да искате да инстанцирате променливи и обекти, които имат обхват на ниво приложения, а не на сесиино ниво. Класът HttpApplicationState ви позволява да направите това. Обектът на приложението е общ термин за инстанцията на този клас на вашето приложение, който е разкрит от Application свйството на System.Web.HttpApplication(родителския клас на Global.asax страница) класа и Application свойството на MobilePage(родителския клас на мобилния Web Forms страница) класа.  
  
 Нещата който трябва да обмислите, когато използвате метода за запазване на състоянието на ниво приложение.  
  - На първо място, информация, съхранявана в application state е обемна. Приложението притежава цялата информация за application state в паметта и не освобождава памет, дори когато потребителят излезе от приложението.  
  - Второ, всички нишки в многонишковите приложения може да достъпват до данни, едновременно, тъй като ASP.NET не заключва автоматично ресурсите.

**14. Програмиране за Windows Phone . Silverlight и XNA архитектура. Терминология. Апаратна поддръжка. Помощни tools.**

Silverlight  
Silverlight framework поддържа управление от събития, XAML-базирано развитие на приложението.  
XNA  
XNA framework поддържа циклично-базирани игри.  
Следната таблица изброява някои от критериите, които можете да използвате, за да определите дали трябва да използвате Silverlight или XNA framework за вашeто приложение за Windows телефон.

|  |  |
| --- | --- |
| Text-based controls and menus | Silverlight |
| Event-driven application | Silverlight |
| Interaction with Windows Phone controls such as Panorama | Silverlight |
| Embedded video | Silverlight |
| Hosted HTML | Silverlight |
| Web browser compatibility | Silverlight |
| Vector graphics | Silverlight |
| Looping game framework | XNA |
| Visually complex applications | XNA |
| 3D games | XNA |
| Advanced art assets such as textures, effects, and terrains | XNAX |

Windows Phone 7 има минимално изискване за хардуер, което прави по-лесно за разработчиците да пишат приложения. Всеки телефон с Windows 7 съдържа следните елементи хардуер:  
WVGA (800 х 480) дисплей.  
Капацитивен мултитъч екран.  
DirectX 9 хардуерно ускорение.  
Датчици за A-GPS, акселерометър, компас, светлина и близост.  
Дигитална камера. Start, Search, и Back копчета.  
Поддръжка на данни за свързване чрез клетъчните мрежи и Wi-Fi.  
256 MB (или повече) RAM и 8 GB (или повече) флаш памет.  
  
**Терминология  
Кодово име Metro дизайн:** Потребителският интерфейс (UI), използван в Windows phone. Трябва да следвате този дизайн в своите приложения, така че да се интегрира с операционната система и други приложения. Дизайнът осигурява модерен потребителски интерфейс, който е лесен за използване, като същевременно минимизира консумацията на енергия от телефона.  
 **Плочка(Tile):** представяне на приложението, което се появява в началния екран. Плочката може да бъде проектирана да бъде динамична и да показва информация на потребителя.  
 **Status Bar:** Показва статуса на телефонни операции, като силата на сигнала. Не е задължително за специфично приложение.  
**Помощни инструменти**Expression Blend for Windows Phone   
Visual Studio 2010 Express for Windows Phone   
Windows Phone emulator   
Zune software   
XNA Game Studio 4.0   
Silverlight   
.NET Framework 4   
Ако имате предварително инсталиран Visual Studio 2010 (Professional или Ultimate), можете да го използвате за мобилни приложения, след като инсталирате Windows Phone Developer Tools.

**15. Създаване на приложение за Windows Phone 7 (Silverlight структура).**Start->Microsoft Visual Studio 2010 Express for Windows Phone->New Project(Windows Phone Application template)->Finish  
Приложенията на Silverlight са комбинация от markup(за интерфейса) и код. Използва се Extensible Application Markup Language или XAML, като се произнася „замел“. XAML предимно дефинира оформлението на потребителския интерфейс, неговите контроли и панели. Code-behind файловете също могат да извършват някаква инициализация и логика, но предимно се използват за поддръжката на събитията, свързани с контролите.  
Файловете, който се използват при разработването на стандартно приложение са MainPage.xaml и MainPage.xaml.cs. MainPage.xaml дефинира потребителския интерфейс на приложението. XAML е XML-базиран декларитивен език, използван за създаването и оформлението на елементите на потребителския интерфейс. При разгръщането на MainPage.xaml се вижда C# code-behind файл на име MainPage.xaml.cs. Този файл е съединен с XAML файла чрез частичен клас, който съдържа логиката за XAML файла. Разделянето на потребителския интерфейс от кода позволява разделното създаване на интерфейса и логиката зад него, правейки работата на дизайнерите и програмистите по-лесна.

Когато се пусне програма **App Class** декларира обект от типа **PhoneApplicationFrame.**

Тази рамка е широка 480 пиксела и 800 пиксела висока и заема цялата повърхност на дисплея на телефона. Обектът от тип **PhoneApplicationFrame** после се държи малко като уеб браузър като навигира до обект наричан **MainPage** (главната страница).

**16. Добавяне на TextBlock към приложение. Добавяне на графичен обект.**

Добавяне на Блок с текст (**TextBlock**)

* Ако **MainPage.xaml** не е вече отворен, с двойно натискане на **MainPage.xaml** в

**Solution Explorer** прозореца.

* На менюто **View**  натиснете **Other Windows,** натискате **Toolbox** . Прозореца с

Инструменти **Toolbox** се показва.

* Преоразмерете или закопчайте кутията за да може да я виждате и да виждате целия телефон едновременно.
* От **Toolbox** кутията издърпаите **TextBlock** на главния панел на телефона.

В **XAML** изглед, забележете, че ***TextBlock*** е добавен в панела **Grid content** (съдържание на мрежата)

**MainPage.xaml** сега включва:

<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0"> <TextBlock Text="Hello, World" HorizontalAlignment="Center" VerticalAlignment="Center" /> </Grid>

* На менюто изглед(**View**) , натиснете други прозорци(**Other Windows**), след това натиснете прозорец с характеристики(**Properties Window**). Прозорецът с карактеристики се показва.

Добавяне на **Graphics** или графики

В Silverlight може да добавяте графики използвайки класовете за форми (**Shape classes).**

Вие може да създавате прости форми, като правоъгалници или по комплексни фигури като полигони. Четки(**Brushes**) се използват за оцветяване или изрисуване на обекти в Silverlight. Вие ще започнете като добавите **StackPanel** около **TextBlock**-а. **Panel** –ът е контейнер, който се използва за групиране и полагане на UI елементи. Всяко приложение трябва да има поне един панел. **StackPanel** полага всеки елемент един след друг, или вертикални или хоризонтално в зависимост от орентацията. **Grid**(мрежата) и **Canvas** (платното) панелите позволяват за по-точно позициониране на елементите. Формата която ще създадете е елипса. Елипсата ще се появи в **TextBlock** в **StackPanel –**лът. Вие ще зададете височина и ширина на елипсата също така и запълването (**Fill**). За запълването вие трябва да зададете четка(**Brush**) за да нарисувате елипсата.

**17. Добавяне на бутон и обработчик. Публикуване и marketplace.**

Silverlight има богата библиотека с контроли, която включва **Button**(бутон), **TextBox**(текстова кутия), **ListBox**(списъчна кутия)и много други. Съществуват две стъпки за добавяне на бутон.Първата стъпка е да добавим **Button** елемент в **XAML**. Втората стъпка е да се добави логика за обработване на събитие генериран от взаимодействието на ползвателя, като натискате на бутон.

* В **XAML** изглед добавете следния **XAML**  код.

<BUTTON HEIGHT="150" Width="300"

* Name="FirstBUTTON" Content="Tap" /> Студиото може да създаде обработчик на събития за вас.

Публикуване в Marketplace

Когато сте завършили приложението и желаете да го дистрибутирате публично за безплатно сваляне или да го продавате, можете да го направите по следния начин. За да направите това като подавате вашето приложение на пазара на Windows Phone. Вие изпращате вашето приложение за публикуване през **App Hub** , където то отива през процес на сертификация за да се осигури, че то изпълнява всички изисквания. Когато приложелието е сертифицирано, се създава страница в Marketplace-а за да се намира от телефоните в Zune софтуерът.

**18. Особености при изобразяване на малък дисплей. Видове и решения.**

При първоначалното пускане на Windows Phone 7, устройствата трябва са с размер на дисплея 480 × 800 пиксела. В бъдеще, е възможна и поддръжката на екрани с размер 320 × 480.

Тези размери са познати като „голям“ и „малък“. Най-големият общ знаменател на хоризонталните и вертикални размери на двата дисплея е 160, поради което могат да се представят като сбор от множество квадрати с размер 160х160 пиксела. Разбира се, телефоните могат да се завъртат в панорамен режим, като някои приложения могат да изискват определена ориентация.  
 Всички размери в Silverlight са в пиксели. При размер на шрифта 36, се оказва, че реалният размер е 48 пиксела, поради разстоянието, необходимо за предотвратяване на сливането на текста. Обикновено размерите на шрифта се изразяват в точки, като една точка е близо 1/72-ра от инча. Преобразуването от пиксели в точки става по следния начин:  
 При принтери с 600 точки за инч(DPI), 72-точков шрифт ще е 600 пиксела! Обикновено, дисплеите са с около 100 DPI(най-често 96), което определя размера на точките- 1 точка= ¾\*пиксел, а 1 пиксел= 4/3\*точка.  
 При мобилните устройства с високи резолюции се стига до DPI с размери 264(за 480х800).

**19. Работа с данни. Източници на данни. Терминология.**

****

Файлове с ресурси **Resource Files**

Файловете с ресурси са вградени в пакета на проекта (.xap). Преимуществото на ресурс файловете е, че файла винаги ще бъде на разположение на приложението. Макар , че на вашето приложение може да му отнеме повече време да стартира, когато използвате ресурс файлове. Вие може да доступите ресурс файловете като използвате Аpplication.GetResourceStream().

Файлове със съдържание **Content Files**

Най- вече за причини свързани с производителност, файловете със съдържание са предпочитани пред ресурс файловете за Windows Phone 7 приложения. **Content** файловете са включени в пакета на приложението (.xap) без да има нужда от вграждане в сглобяването на проекта. Макар , че те не са компилирани в сглобката (**assembly**), сглобките са компилирани с **metadata** която установява асоциация с всеки фаел от тип съдържание.

Isolated Storage изолирано съхранение

Ако се искате да запазите или да получите специфична потребителска информация вие може да използвате Isolated Storage(Изолираното съхранение). Silverlight за Windows Phone приложения не съществува директен достъп до файловата система на операционната система. Макар , че вие може да използвате Isolated Storage за да съхраните или получите данни локално на устройството. Съществуват 2 начина за използване на isolated storage:

* Да запазваме или получаваме данни като двойки от ключ/стойност(**key/value**) които използват класа **IsolatedStorageSettings**
* Да запазване или получаваме файлове като използвале класа **IsolatedStorageFile**

**20. Web услуги и протоколи.**

**От презентациите на Наков:**

Web Service терминология

Работата с различни уеб услуги може да бъде малко объркващо заради различните типове услуги, формати и технологии. Описаните по доло са някои термини свързани с уеб услугите.

Термин. Описание

**web service** (уеб услуги) – Дялове от логиката на приложението която предоставя данни и услуги до други приложения. Приложенията достъпват уеб услуги като използват стандартни уеб протоколи и формати за данни като HTTP, XML, и SOAP, независимо как всеки елемент е имплементиран.

**REST** - (Representational State Transfer Protocol) . протокол за излагане на ресурси в уеб пространството за достъп от клиенти.

**POX** - (Plain Old XML) – термин използван за описание на базовия XML

**JSON** - (JavaScript Object Notation)-Много лек формат за обмен на данни. Създаден е за да бъде четим от хората, но също и независим от компютъра.

**OData** -(Open Data Protocol)- уеб протокол за заявки и обновяване на данни.

**SOAP**(сапун :D)- (Simple Object Access Protocol)-Лек протокол предвиден за обмен на структорирана информация в децентрализирана, дистрибутирана среда.

**От wikipedia:**

**20.1.**Уеб услугата представлява софтуерна истема, която предоставя комуникация м-у взаимносъвместими системи по мрежи според определението на консорциума W3C.Напрактика това означава стандартни интерфейси м-у приложенията.Тези стандарти се основават на технологии, които са общи за всички платформи мрежов софтуер, каквито са например XML и HTTP.

**20.2.**Протоколи за уеб услуги – в основата на уеб услугите са 3 стандарта, базирани на XML:

* SOAP (Simple Object Access Protocol) – стандартен протокол, който се използва за обмен на данни м-у приложения.Този стандарт е стандарт на W3C.
* WSDL (Web Service Description Language) – програмен език, който позволява да се опише интерфейса на уеб услуга.Този стандарт също е стандарт на W3C.
* UDDI (Universal Description, Discovery and Integration) – регистър

Стандартите при уеб услугите се определят от 2 организации:

* World Wide Web Consortium (W3C).
* Organization for the Advancement of Structured Information Standards (OASIS).

**21. Пример за привързване на данни към контрол.**

Data Binding на контрол на елемент. Пример. Следният код показва пример за свързване на контрола на една еденица данни. Целта е свойството на Text на контрола textBox. Източникът е от класа на информация за музика:Recording.  
**XAML**  
<GRID X:NAME="CONTENTPANEL" GRID.ROW="1" MARGIN="12,0,12,0">  
<TextBox VerticalAlignment="Top" IsReadOnly="True" Margin="5" TextWrapping="Wrap" Height="120" Width="400" Text="{Binding}" x:Name="textBox1" />  
</ Grid>  
**Code-behind**  
public MainPage() {

InitializeComponent();

// Set the data context to a new recording

textBox1.DataContext = new Recording("Chris Sells", "Chris Sells Live",

new DateTime(2008, 2, 5)); }

// A SIMPLE BUSINESS OBJECT

public class Recording

{

{ public Recording() { }

public Recording(string artistName, string cdName, DateTime release) {

Artist = artistName;

}

Name = cdName;

ReleaseDate = release;

public string Artist { get; set; }

public string Name { get; set; }

public DateTime ReleaseDate { get; set; }

// Override the ToString method.

public override string ToString()

{ return Name + " by " + Artist + ", Released: " + ReleaseDate.ToSortDateString(); }

}

**22. Програмиране с Windows Phone SDK 7.1 (Mango). Привързване към данни.**

“Mango” вътрешното кодово име на излезлия Windows Phone SDK 7.1.

Някои от новите преимущества на 7.1 версията са:

-Локални бази от данни и LINQ до SQL

-второстепенни плочни и дълбоко свързване

**Създаване база с данни и клас от тип DataContext**

Windows Phone SDK 7.1 поддържа локални бази данни.Това означава ,че приложението може да съхранява данни в локалната база от данни в файла (.sdf) на телефона.Ние препоръчваме да създадете базата данни в кода, или като част от самото приложение или чрез отделно помощно приложение. За да създадете базата данни в кода ние ще се нуждаем от клас извлечен от стандартния клас на DataContext. Същия DataContext клас може да бъде използван както и за помощното приложение което създава базата данни така и за главното приложение което използва базата данни. В помощното приложение ние трябва да специфицираме мястото на базата данни което трябва да е в isolated storage, защото това е единственото място където ние можем да записваме от приложение на телефона.Този клас също съдържа набор от Table fields(таблични полета)за всяка таблица от базата данни:

public class MangoDataContext : DataContext

{

public MangoDataContext():

base("Data Source=isostore:/Mangolicious.sdf") { }

public Table<Recipe> Recipes;

public Table<Fact> Facts;

public Table<Cocktail> Cocktails;

}

Нуждаем се от ViewModel class в помощното приложение за да посредничи между View (the UI) и Model (the data) използвайки класът DataContext. ViewModel има DataContext поле и набор от колекции за данните на таблицата(рецепти, факти и коктейли).Вида на данните са статични, така че прости List<T> колекции са достатъчни.За същата причина ние само се нуждаем от оценители на свойства, не набор от модификатори.

public class MainViewModel

{

private MangoDataContext mangoDb;

private List<Recipe> recipes;

public List<Recipe> Recipes

{

get

{

if (recipes == null)

{ recipes = new List<Recipe>(); }

return recipes;

}

}

... additional table collections omitted for brevity

}

В подходящо място в помощното приложение може би в обработчик на (button click handler) ние можем да извикаме CreateDatabase метода. Когато ние пуснем помощника(или в емулатора или на физическото устройство) файла с базата данни ще бъде създаден в isolated storage на устройстовто. Финалната задача е да се извлече този файл до десктопа, за да можем да го използваме в главното приложение. За да направим това ние ще използваме Isolated Storage Explorer инструментът, инструмент от тип командна линия (command-line) който се предоставя заедно с Windows Phone SDK 7.1. Като извлякохме SDF файла на десктопа, ние завършихме с помощното приложение и може да отдадем нашето внимание на главното приложение, което ползва тази база данни.  
 Главното приложението също се нуждае клас от тип ViewModel и ние може да ползваме ViewModel класът от помощното приложение. Нуждаем се от DataContext полето и набор от List<T> свойства колекции за таблиците с данни.   
 Критичния метод в ViewModel е LoadData метода. Тук ние инициализираме базата данни и създаваме LINQ-to-SQL заявки, за да заредим данните чрез DataContext в колекция във вътрешната памет. Можем да презаредим всички три таблици до този момент, но искаме да оптимизираме стартирането като забавяме зареждането да данните ако, или докато съответната страница не се посети.  
 Ние може да използваме подобни LINQ-to-SQL заявки за да създадем отделни методи които можем да използваме след пускането за зареждане на съответните им данни при поискване.

**23. Създаване на UI. Нива на вложеност и преходи. Пример.**

Създаване на UI

Главната страница се състои от една панорама Panorama с три PanoramaItems( панорамни елемента).Първия елемент се състои от ListBox които предоставя главно меню за приложението. Когато ползвателя избере един от елементите в ListBox-а ние навигираме до кореспондиращата страница- това е страницата с колекцията за или Рецепти или Факти или Коктейли.Точно преди да навигирате ние трябва да се подсигурим, че сме заредили кореспондиращите данни в Рецепти,Факти и Коктейли(части от приложението за мангото):

switch (CategoryList.SelectedIndex)

{

case 0:

App.ViewModel.LoadRecipes();

NavigationService.Navigate( new Uri("/RecipesPage.xaml", UriKind.Relative));

break;

... additional cases omitted for brevity

}

Когато ползвателя избере един елемент от списъка с сезони акценти (Seasonal Highlights) в UI,ние разглеждаме избрания елемент за да видим дали е рецепта или коктейл и да се насочим до индивидуалната страница за Коктейл или Рецепта, преминавайки през ИД то на елемента като част от заявката за навигиране:

SeasonalHighlight selectedItem = (SeasonalHighlight)SeasonalList.SelectedItem;

String navigationString = String.Empty;

if (selectedItem.SourceTable == "Recipes")

{

App.ViewModel.LoadRecipes();

navigationString = String.Format("/RecipePage.xaml?ID={0}", selectedItem.ID);

}

else if (selectedItem.SourceTable == "Cocktails")

{

App.ViewModel.LoadCocktails();

navigationString = String.Format("/CocktailPage.xaml?ID={0}", selectedItem.ID);

}

NavigationService.Navigate(new System.Uri(navigationString, UriKind.Relative));

Потребителя може да навигира от менюто на главната страница до един от трите страници със списъци. Всяка от тези страници се свързва по данни с една от колекцията в ViewMode за да покаже списък от елементи: Рецепти,Факти и Коктейли. Всяка от тези страници предлага прост ListBox където всеки елемент от списъка съдържа конторил за изображение за снимката и TextBlock за име на елемента Фигурата показва FactsPage:

Когато потребителят избере индивидуален елемент от списъка с Рецепти,Факти или Коктейли, ние го пращаме до идивидуалната страница за Рецепти,Факти или Коктейли

минавайки по ИДто на индивидуалния елементи и заявката за навигация. Отново тези страници са почти еднакви, като предлагат в себе си Изображение и някъв текст доло.

Кодът зад всяка страница е почти идентичен и прост. В OnNavigatedTo() отмяната ние извличаме индивидуалния ИД на елемента от низа на заявката, намираме този елемент от ViewModel колекцията и го правим свързан по данни.

Кодът за страницата рецепти RecipePage е малко по сложен от другите- добавения код в тази страница изцяло е свързан с HyperlinkButton бутонът позициониран на горния десен ъгъл на страницата:

Когато потребителя натисне копчето закрепи HyperlinkButton на индивидуалната страница на рецептата ние закачаме този елемент като плочка към стартовата страница на телефона. Действието закачане изпраща потребителя към началната страница и деактивира приложението. Когато плочка е закрепена по този начин тя се анимира периодично обръщайки се между отпред и отзад както е показано:

Потребителят може да натисне на закрепената плочка която навигира директно до елемента в приложението. Когато достигне страницата бутона“ прикрепи“ ще има изображение „откачи“ . Ако той окача страницата той ще бъде премахнат от стартовата страница, и приложението ще продължи.

За да стане това програмистът трябва да :

1. Дефинира таг в настоящата страница маркиран като „ прикрепен“
2. Дефинира код обработчик на събитиети PinUnpin\_Click() дефиниращ начина и заместването на „Откачането“

**24. Работа със сензори (камера) под Windows Phone 7. Пример.** Въпреки че камерата винаги е била в Windows Phone, единствената API, която е налична в оригиналния релийз е CameraCaptureTask. Този клас създава дъщерен процес, който позволява на потребителя да заснеме снимка, след което връща снимката към приложението. Приложението не може да контролира която и да е част от този процес, нито пък да приема видео поток в реално време, който постъпва от обектива. Това се поправя чрез два програмни интерфейса- един за класовете на камерата и фотокамерата, които позволяват на приложението да асемблира потребителския интерфейс за заснемането с камерата, достъп до бутона на затвора, светкавица и други. Тези API-та са наследени от webcam интерфейса в Silverlight 4.  
Webcam интерфейсът се състои от много класове, дефинирани в System.Windows.Media namespace. Винаги ще започвате със статичен CaptureDeviceConfiguration клас. Може проста да се извикат GetDefaultVideoCaptureDevice() и GetDefaultAudioCaptureDevice() методите  
Приложението трябва да създаде и CaptureSource обект, който комбинира видето и от устройството със звука. CaptureSource има две свойства- VideoCaptureDevice и AudioCaptureDevice, които трябва да инстанцират VideoCaptureDevice и AudioCaptureDevice от CaptureDeviceConfiguration. След създаването на CaptureSource обект, могат да се извикат методите му Start() и Stop(). Също такаIn може да се използва и CaptureImageAsync() метода от CaptureSource, за да се извлечат отделни кадри от видеото под формата на WriteableBitmap обекти  
 Най-лесният начин за използване на CaptureSource е VideoBrush. Като всяка четка(brush), може да се използва за „оцветяване“ на фона на елементи.  
**XAML *<VideoBrush x:Name="videoBrush" />*Code-behind *captureSource = new CaptureSource***

***VideoCaptureDevice = CaptureDeviceConfiguration.GetDefaultVideoCaptureDevice() }***

***protected override void OnNavigatedTo(NavigationEventArgs args)***

***{ if (captureSource != null && CaptureDeviceConfiguration.AllowedDeviceAccess)***

***{ videoBrush.SetSource(captureSource);***

***captureSource.Start();***

***startButton.Visibility = Visibility.Collapsed; }***

***base.OnNavigatedTo(args);***

***}***

***protected override void OnNavigatedFrom(NavigationEventArgs args)***

***{ if (captureSource != null && captureSource.State == CaptureState.Started)***

***{ captureSource.Stop();***

***startButton.Visibility = Visibility.Visible; }***

***base.OnNavigatedFrom(args);***

***}***

***void OnStartButtonClick(object sender, RoutedEventArgs args)***

***{ if (captureSource != null && (CaptureDeviceConfiguration.AllowedDeviceAccess ||***

***CaptureDeviceConfiguration.RequestDeviceAccess())***

***{ videoBrush.SetSource(captureSource); captureSource.Start();***

***startButton.Visibility = Visibility.Collapsed; }***

***}***

**25. Sink обекти и работа с тях по обработка на видео или аудио вход.** Алтернативата на използване на VideoBrush е свързването на CaptureSource обект към AudioSink, VideoSink или FileSink обект. “Sink” в тези класове е в смисъла на „приемане“. Класът FileSink е предпочитан метод за запазването на видео или аудио потоци в isolated storage на приложението, без ваша намеса. При нужда от достъп до самото видео или аудио в реално време, се използва VideoSink и AudioSink. Тези два класа са абстрактни. Вашият клас наследява един или двата абстрактни класа, като пренаписва OnCaptureStarted, OnCaptureStopped, OnFormatChange и OnSample методите. Класът, който наследява VideoSink или AudioSink винаги ще извика OnFormatChange преди първото повикване на OnSample. Информацията от OnFormatChange показва как се интерпретират данните. И за двата класа(VideoSink и AudioSink), OnSample повикването предоставя информация за времето и масива от байтове. За AudioSink тези байтове представляват PCM занните, а за VideoSink- байтовете са редове и колони от пиксели за всеки кадър от видеото.

**26. Операционна система Android. Пазарно пространство.**

**Android (транскрипция: Андроид) е операционна система за мобилни устройства. Тя е поддържана и развивана от консорциума Open Handset Alliance, ръководен от Google Inc..**

През 2005 г. Google Inc. закупува операционната система Android от Android Inc. и започва нейното развитие. Пускането и на пазара на 5 ноември 2007 г. е съпроводено с основаването на Open Handset Alliance - консорциум от 84 хардуерни, софтуерни и телекомуникационни компании, решени да развиват отворените стандарти при мобилните устройства. Google пуска по-голяма част от кода под свободен лиценз.

Операционната система Android използва в основата си модифицирана версия на ядрото на Linux.

За развитието на Android се грижат голям брой софтуерни разработчици, които създават така наречените "apps" - малки приложения, които разширяват функционалността на системата. Приложенията могат да бъдат сваляни от различни сайтове в Интернет или от големи он-лайн магазини като Android Market (ново име Google Play) - магазинът на Google. По данни към януари 2011 г. за Android има над 200 000 приложения.[1] Приложенията се пишат предимно на Java, Python или Ruby.

За последното тримесечие на 2010 г. Android е определена като най-продаваната платформа за смартфони в света. [2] 52,5% от всички продадени за третото тримесечие на 2011 г. смартфони са с Android

**27. Архитектура на операционната среда Android. Софтуерни слоеве.**

Основата на Android е [ядрото на Linux](http://bg.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BD%D1%83%D0%BA%D1%81_%28%D1%8F%D0%B4%D1%80%D0%BE%29) (версия 2.6). То е отговорно за управлението на паметта и [процесите](http://bg.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81_%28%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29), както и за [мрежовите връзки](http://bg.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8E%D1%82%D1%8A%D1%80%D0%BD%D0%B0_%D0%BC%D1%80%D0%B5%D0%B6%D0%B0). Тук са разположени и [драйвърите](http://bg.wikipedia.org/w/index.php?title=%D0%94%D1%80%D0%B0%D0%B9%D0%B2%D1%8A%D1%80&action=edit&redlink=1).

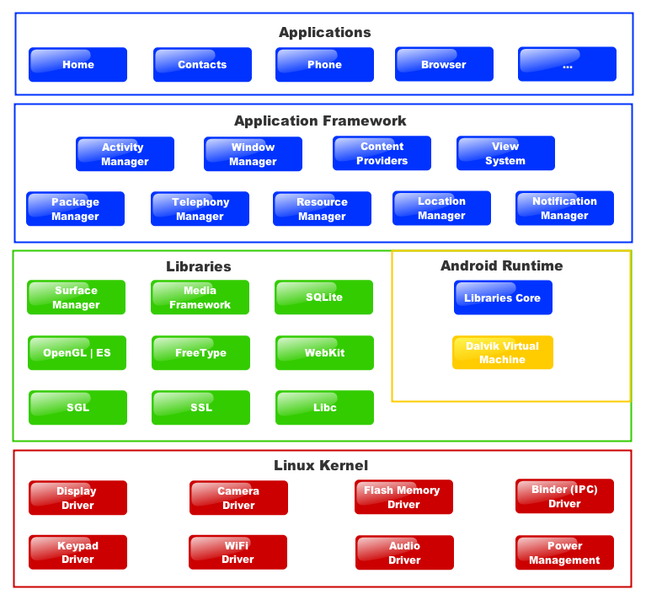
Директно над ядрото се намира т. нар. "Runtime Environment". Тя съдържа най-важните библиотеки по време на изпълнение и най-важната функционалност на езика [Java](http://bg.wikipedia.org/wiki/Java). Тук се съдържа и виртуалната машина Dalvik Virtual Machine (DVM). Тя се различава от класическите виртуални машини на Java (Java Virtual Machine, JVM) по това, че е оптимизирана за мобилни уреди с малко памет. Оптимизацията позволява и едновременното изпълнение на няколко виртуални машини на същия уред.

Android съдържа няколко библиотеки на [C](http://bg.wikipedia.org/wiki/C)/[C++](http://bg.wikipedia.org/wiki/C%2B%2B):

* Surface Manager (управлява достъпа до дисплея)
* OpenGL ES ([приложно-програмен интерфейс](http://bg.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B5%D0%BD_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) за 3D компютърна графика), SGL (приложно-програмен интерфейс за 2D компютърна графика)
* Media Framework (управление на мултимедийно съдържание, на основата на OpenCORE, поддържа формати като MPEG4, H.264, MP3, AAC, AMR, JPG PNG и др.)
* FreeType (библиотека за рендъринг на пикселни и векторни шрифтове)
* SSL (криптиране)
* SQLite (бази данни)
* [WebKit](http://bg.wikipedia.org/w/index.php?title=WebKit&action=edit&redlink=1) (рендъринг на HTML)
* Libc (версия на стандартната C-библиотека за Android)

Приложният [фреймуърк](http://bg.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D1%83%D1%8A%D1%80%D0%BA) (Application Framework) ползва библиотеките на C/C++ и предлага стандартизиран приложно-програмен интерфейс за програмистите на приложения (Apps).

Android се доставя с няколко приложения, сред които са комуникационните приложения за телефониране, [електронна поща](http://bg.wikipedia.org/wiki/%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0_%D0%BF%D0%BE%D1%89%D0%B0), [SMS](http://bg.wikipedia.org/wiki/SMS) и [браузър](http://bg.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D1%8A%D1%80), както и [Google Maps](http://bg.wikipedia.org/wiki/Google_Maps), календар и приложение за управление на контактите.



**28. Компоненти на приложение в Android. Комуникации между компонентите.**

**Важни Андроид компоненти.**

Ето и някои от по важните компоненти за едно Андроид приложение.

* **Activity** – Представя тип слой от приложението например екран, който потребителя ще вижда. Едно приложение може да има няколко слоя активности и може да превключва между тях по време на работа на проложението. Потребителския интерфейс на тези слоеве е изграден от widget класове които са свързани от "android.view.View" Слоевете view се управляват от  ViewGroups.
* **Services** – Услугите ви осигуряват фонови задачи без наличието на интерфейс. Те могат да известяват потребителя посредством notification framework в Андроид.
* **Content Provider –** Доставя данни от приложението . Благодарение на този компонент можете да споделяте данни с други приложения. Андроид съдържа SQLite база данни, която може да служи като доставчик на тези данни.
* **Intents** – Това са асинхронни съобщения, които позволяват приложението да иска функционалност от други услуги или дейности. Приложението може директно да извика услуга или дейност (explicit intent) или да извести  “системата за регистрация на услуги и приложения на Андроид”  за тези си намерения (implicit intents). Например приложението може да извика чрез Intent приложението за контакти. Приложението се регистрира посредством Intent филтър (IntentFilter).  Компонента Intent е много мощен инструмент благодарение на който можете да създавате свободно свързани приложения.
* **Broadcast Receiver**  - Получава системни съобщения, както и асинхронни съобщения implicit intents (за справка виж малко по на-горе). Може да се използва за реакция при променящи се условия в системата. Приложението може да се регистрира като приемник (**Broadcast Receiver** ) за определени събития, и може да се стартира при наличието на такова (събитие).

**29. Манифест на приложение.**

**30. Android SDK и процес на разработка. Публикуване на приложение**

Разработка на приложения за Android устройства, се улеснява от група инструменти, които се предоставят с SDK. Можете да получите достъп до тези инструменти чрез плъгин Eclipse нарeчeн ADT (Android Development Tools) или от командния ред. Разработка с Eclipse е предпочитаният метод, защото може пряко да се позовете на инструменти,от които се нуждаете, повреме на разработката на приложения.  
  
  
Основните стъпки за разработване на приложения с или без Eclipse са едни и същи:  
1. Настройте Android Virtual Devices или хардуерни устройства. Трябва да създадете Android Virtual Devices (AVD) или да се свържете хардуерни устройства, на които ще инсталирате приложения. Един Android Virtual Device (AVD) е емулатор конфигурация, която ви позволява да симулирате действително устройство чрез определяне на хардуерни и софтуерни опции да бъдат емулирани от Emulator Android. Най-лесният начин за създаване на AVD е да използвате графичния AVD мениджър, който стартира от Eclipse  
2. Създаване на проекта Android. Един проект Android съдържа всичкия сорс код и ресурсни файлове за вашето приложение. Той е построен в apk пакет, които можете да инсталирате на устройства с Android.  
3. Изгради и стартирай вашето приложение.  
4. Debug на вашето приложение с SDK debuging инструменти. инструментитe са осигурени с Android SDK. Eclipse вече идва в комплект със съвместим дебъгер.  
5. Тествайте вашето приложение с тестване и **Instrumentation framework**.  
  
An AVD е конфигурацията на устройството за Android емулатор, който ви позволява да моделирате различни конфигурации на Android-устройства. Когато стартирате AVD мениджъра в Eclipse или стартирате Android инструмент на командния ред.  
  
  
Android SDK  
  
Първо - създадат своя SDK.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | ***Platform*** | | |  | | --- | | ***Package*** | | |  | | --- | | ***Size*** | |
| |  | | --- | | **Windows** | | |  | | --- | | **android-sdk\_r12-windows.zip** | | |  | | --- | | **36486190 bytes** | |
|  | |  | | --- | | **installer\_r12-windows.exe (Recommended)** | | |  | | --- | | **36531492 bytes** | |
| |  | | --- | | **Mac OS X (intel)** | | |  | | --- | | **android-sdk\_r12-mac\_x86.zip** | | |  | | --- | | **30231118 bytes** | |
| |  | | --- | | **Linux (i386)** | | |  | | --- | | **android-sdk\_r12-linux\_x86.tgz** | | |  | | --- | | **30034243 bytes** | |

Eclipse IDE Android Development Tools (ADT) е плъгин за Eclipse IDE, която е предназначена да ви даде мощтна интегрирана среда, в която да се изгради Android приложения.

**31. Android приложение. Файлова структура. Създаване и изпълнение.**

Android API-то включва повече от 70 пакета, с повече от 400 класа  
Първо приложение:Hello,World  
Като разработчик, вие знаете, че първото впечатление за development frameworka-a е колко лесно еда се напише "Hello, World". Е, на Android, това е доста лесно. Това е особено лесно, ако използвате Eclipse IDE, защото този плъгин управлява вашия проект,значително ще ускори вашият цикъл на разработка

1.Вие ще стартирате вашето приложение в Emulator Android. Преди да можете да стартирате емулатора, трябва да създадете Android Virtual Device (AVD). AVD определя системен образ и настройките на устройството, използвани от емулатора  
2. Създаване на нов проект Android  
След като сте създали AVD можете да преминете към следващата стъпка и да започне нов проект Android в Eclipse.  
Вашият Android проект вече е готов. Той трябва да се вижда в пакет Explorer в ляво. Отворете файла HelloAndroid.java, разположени вътре HelloAndroid ◊ ◊ SRC com.example.helloandroid Тя трябва да изглежда така: **package com.example.helloandroid; import android.app.Activity; import android.os.Bundle; public class HelloAndroid extends Activity { /\*\* Called when the activity is first created. \*/ @Override public void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.main); } }**  Забележете, че този клас се базира на Activity класa. Activity е един обект приложение, което се  
използва за извършване на действия. Приложението може да има много отделни дейности, но потребителят взаимодейства  
с тях един по един. The onCreate () метод се извиква от системата Android, когато вашето Activity започва - това е мястото, където трябва да се изпълни всичко инициализация и UI настройка.

**32. Примерно приложение в Android. Добавяне на интерфейс.**

Построява се UI  
Обърнете внимание кода по-долу и след това направете същите промени във вашия HelloAndroid клас.  
Сред отличителните елементи са редове които са били добавени.  
пакет com.example.helloandroid;  
  
 import android.app.Activity;

import android.os.Bundle;

**import android.widget.TextView;**

public class HelloAndroid extends Activity {

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

**TextView tv = new TextView(this);**

**tv.setText("Hello, Android"); setContentView(tv);** } }

Android UI се състои от йерархии от обекти, наречени Views (изгледи). A View е drawable обект използвaн като елемент в UI изглед, например бутон, изображение или (в този случай) TextLabel. Всеки от тези обекти е подклас на View класа и подклас на , който може да обработва текста е TextView. В тази промяна, вие създавате TextView с класа конструктор, който приема за параметър инстанция на Android Context. Контекст е манипулатор към системата, той предлага услуги като разрешаване на ресурси, получаване на достъп до бази данни и предпочитания, и така нататък. Activity класa наследява от контекста, и защото вашият HelloAndroid клас е подклас на Activity, тя е и контекст. Така че, може да го подавате като Context на TextView. След това можете да зададете текстовото съдържание с setText (). И накрая, можете да подадете setContentView (),TextView за да го покажете, като съдържание на Activity UI-a. Ако вашeто Activity не извиква този метод, тогава никакъв UI не се показва и системата ще покаже празен екран. Ето го - "Hello, World" в Android! Следващата стъпка, разбира се, е да видите, че работи.

**33. XML интерфейсно описание. R клас. Примери.**

Упгрейд на UI-a до XML лейаут.

Примерът ”hello world ” който тукощо завършихме използва така нареченият „програмируем” UI лейаут.Това означава че вие можете да констуирате и изградите интерфейса на вашето приложение директно в сорс код.Андроид предоставя алтернатива за изграждане на UI:XML-базирани лейаут файлове.Най-леснич начин да се обясни тази концепция е да се покаже пример.Това е XML layout файл който е идентичен като поведение със програмируемият UI:

<?xml version="1.0" encoding="utf-8"?>

<TextView xmlns:android="http://schemas.android.com/apk/res/android" android:id="@+id/textview" android:layout\_width="fill\_parent" android:layout\_height="fill\_parent" android:text="@string/hello"/>

Общата структура на android XML layout-а е проста:тя е дърво от XML елементи където всеки възел е името на View класа.В горния XML пример има само един елемент от View –TextView който има няколко xml атрибута.Тези XML layout файлове принадлежат на res/layout/ директорията на проекта.Eclipse plugin-a автоматично създава един от тези layout файлове:main.xml трябва почти винаги да се определя оформлението в XML файла вместо в кода.  
Следните процедури ще ви инструктират как да промените съществуващо приложение да използва XML оформление.

1.В пакета Eclipse Explorer, разширете / res / layout / folder и Отвори main.xml. Заменете  
Съдържанието със следния XML:  
<TextView xmlns

<XML версия = "1,0" кодиране = "UTF-8"?>:

Android = "http://schemas.android.com/apk/res/android" Android: ИД = "+ ID / textview "

Android: layout\_width =" fill\_parent "Android: layout\_height =" fill\_parent "

Android: текст =" @ низ / Hello "/>  
Запишете файла.  
2. Вътре в res/values/ folder, отворете strings.xml. Това е мястото, където трябва да запази всички текстови низове по подразбиранеза вашия потребителски интерфейс. Ако използвате Eclipse, тогава ADT ще ви направи с два стринга,  
Hello и APP\_NAME. Преработете Hello на нещо друго. Целият файл трябва да изглежда така:  
<? XML version = "1,0" кодиране = "UTF-8">

<resources>

<string name="hello">

Hello, Android! Аз съм низ ресурс! </ string>

<string name="app\_name"> Здравейте, Android </string>

</ resources >  
  
3. Сега отворете и променяте вашия клас HelloAndroid и да използвате XML оформление. Редактиране на файл, за да изглежда така:  
pacage com.example.helloandroid;

public class HelloAndroid extends Activity {

/ \*\* извиква се, когато activity е за първи път създаден. \* /

@ Override

public onCreate void (Bundle savedInstanceState)

{super.onCreate (savedInstanceState);

setContentView (R.layout.main);}}  
Когато направите тeзи промени, въведете го на ръка да опитате с кодово завършване функцията. Като започнете да въвеждате  
"R.layout.main" плъгина ще ви предложи довършване. Ще откриете, че тя помага в много ситуации.  
Вместо преминаване setContentView () View обекта, му давате препратка към оформлението на ресурсите.  
Ресурсът е идентифициран като R.layout.main, който всъщност е компилиран обект представител на  
оформлението определено в /res/layout/main.xml. Плъгина на Eclipse автоматично създава тази препратка за васв класа на проекта R.java.  
  
R класа  
В Eclipse, отворете файла с име R.java (в gen/ [Generated Java Files] folder). Тя трябва да изглежда така:  
package com.example.helloandroid; public final class R { public static final class attr { } public static final class drawable { public static final int icon=0x7f020000; } public static final class id { public static final int textview=0x7f050000; } public static final class layout { public static final int main=0x7f030000; } public static final class string { public static final int app\_name=0x7f040001; public static final int hello=0x7f040000; } }  
Файлът R.java е индекс на всички ресурси, определени във файла.  
Можете да използвате този клас в изходен код като един по-кратък начин да реферира на ресурсите, които сте включили във вашия проект.  
  
примери ":

Relative Layout  
RelativeLayout е ViewGroup който сочи деца на View да гледа вотносителните позиции A ViewGroup е специален view, която може да съдържа и други view (наречени деца.) ViewGroup е базов клас за оформления и контейнери за изгледи. В този клас се определя също ViewGroup.LayoutParams клас, който служи като базов клас за layout параметри.

RelativeLayout  
Оформление, където позициите на деца мoгат да бъде описан по отношение един към друг или към майка.  
  
  
1. Започване на нов проект, наречен HelloRelativeLayout. 2, отворете res/layout/main.xml файл и го поставете следния <XML версия = "1,0" кодиране = "UTF-8"?> <RelativeLayout xmlns: Android = "http://schemas.android.com/ APK / ВЕИ / Android "Android: layout\_width =" fill\_parent "Android: layout\_height =" fill\_parent "> <TextView Android: ИД =" + ID / етикет "  
Android: layout\_width = "fill\_parent" Android: layout\_height = "wrap\_content" Android: текст = "Тип тук:" /> <EditText Android: ИД = "+ ID / влизане" Android: layout\_width = "fill\_parent" Android: layout\_height = " wrap\_content "Android: фон =" @ Android: рисуваемо / editbox\_background "Android: layout\_below =" @ ID / етикет "/> <Button Android: ИД =" + ID / OK "Android: layout\_width =" wrap\_content "Android: layout\_height = "wrap\_content" Android: layout\_below = "@ ID / влизане" Android: layout\_alignParentRight = "истинска" Android: layout\_marginLeft = "10dip" Android: текст = "OK" /> <Button Android: layout\_width = "wrap\_content" Android: layout\_height = " wrap\_content "Android: layout\_toLeftOf =" @ ID / OK "Android: layout\_alignTop =" @ ID / OK "Android: текст =" Cancel "/> </ RelativeLayout>  
  
Забележете, всяка от Android: layout\_ \* атрибути, като например layout\_below, layout\_alignParentRight и layout\_toLeftOf. При използване на RelativeLayout, можете да използвате тези атрибути да се опише как искате да разположите всеки View. Всеки един от тези атрибути определят различен вид на относителното положение. Някои атрибути използва ресурса ID на View да определят своя собствена относителна позиция. Например, последния бутон се определя да лежи на ляво-на и съобразена-with-the-top-of на View идентифицира с ID (което е предишния Button). Всички налични атрибути на оформлението са определени в RelativeLayout.LayoutParams. 3. Уверете се, че сте поставили този план в onCreate () метод: onCreate() method: public void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.main); ThsetContentView (Int) метод зарежда файла за оформление за Activity, определени от идентификатор на ресурс – ID R.layout.main отнася до res/layout/main.xml layout файл.

4. Изпълнете приложението  
  
Пример на views (2)  
ListView  
ListView е ViewGroup, която създава списък на скролируеми предмети. Елементите на списъка  
автоматично се добавя към списъка с помощта на ListAdapter Google Map View С помощта на Google Maps библиотека, можете да създадете своя собствена map-viewing Activity  
  
Web View  
WebView ви позволява да създадете свой собствен прозорец за разглеждане на уеб страници  
(Или дори да се разработи браузър).  
Grid View GridView е ViewGroup който показва елементите в двумерена, скролируема мрежа. Решетката елементи автоматично се добавя към оформлението с помощта на ListAdapter