CMER

Centre for Mobile Education and Research

Considerations, Practices, and Guidelines for Mobile Devices (BlackBerry)

Overview

- Designing Mobile Applications
- Design Considerations & Guidelines
- Writing Efficient Blackberry Code
- Best Practices for Blackberry Developers
- Programming using the Blackberry JDE
- Blackberry MDS
- Push Technology
- Application Models

Designing Mobile Applications

- Developers need to keep some things in mind when designing applications for mobile devices
  - Display size
  - Processor speed
  - Network
  - Memory
  - Battery Life

Design Consideration

- Display size
  - Limited screen
  - Ability to only display a limited number of characters
- Processor Speed
  - Slow processor
  - Unable to upgrade

Design Consideration (Cont.)

- Network
  - Slow transfer rate
  - Cost (WIFI vs. Data Plan)
- Memory
  - Limited amount of memory
  - Not upgradable
- Battery Life
  - Restricted battery life

Guidelines for Developing BlackBerry Applications

- Purpose
  - Device Specifications
  - Write efficient BlackBerry code
  - Reduce the size of compiled code

Device Specifications

- Mobile devices are constantly coming to market with leading-edge technology
- Must know your target devices
  - Device capabilities
  - Specifications
    - Hardware
    - Software
- http://na.blackberry.com/eng/devices/

Writing Efficient BlackBerry Code

- Use local variables
  - More efficient to access then class members
- Use shorthand for evaluating boolean conditions
  - Results in shorter compiling code

```
// Avoid this
if( boolean_expression == true ) {
   return true;
} else {
   return false;
}
// Do this
return( boolean_expression );
```

Writing Efficient BlackBerry Code (Cont.)

- Use int instead of long
  - A long is a 64 bit int. Since BlackBerry Devices use a 32 bit processor, operations run two to four times faster if you use an int instead of a long
- Avoid garbage collection
  - Avoid calling System.gc() to perform a garbage collection operation because it might take too much time on the devices with limited available memory. Rather, let the BlackBerry JVM collect garbage.

Writing Efficient BlackBerry Code (Cont.)

- Avoid using java.util.Enumeration objects
  - Only use to hide data
  - Calling these objects is slow and creates unnecessary garbage
- Write efficient loops
  - Factor loop-invariant code out of a loop.

```
// Avoid.
for( int i = 0; i < vector.size(); i++ ) {
.... }
```

  - This avoids extra local on the stack

Writing Efficient BlackBerry Code (Cont.)

- Perform casts using instanceof
  - Faster then using try/catch
  - Only use try/catch when a cast failure is an exceptional circumstance
- Evaluate conditions using instanceof
  - If evaluating a condition using instanceof, do not evaluate explicitly whether the variable is null.
  - Code will be smaller and faster

Reduce the Size of Compiled Code

- Set appropriate access
  - Declare fields as private whenever possible which lets the compiler optimize the .cod file
  - Use default package whenever possible
- Avoid creating interfaces
  - Interfaces produce larger and slower code
- Use static inner classes
  - Declare classes as static when possible
  - Prevents the creation of a reference to the outer class

Reduce the Size of Compiled Code (Cont.)

- Avoid unnecessary field initialization
  - Avoid unnecessarily initializing fields in classes, where fields have default values. If you do not initialize a field in a class, the field initializes automatically using the following default values:
- Import individual classes
  - Applications that use only a small number of classes from a package should import the individual classes rather than the entire library

Best Practices for BlackBerry Developers

- Use multithreading
  - BlackBerry devices have multithreading capabilities thus the following apply:
    - Always create a new thread for network connections or other lengthy operations (more than one-tenth of a second).
    - Use background threads for listeners or other processes that run in the background when the application starts.

Best Practices for BlackBerry Developers (Cont.)

- Minimize memory use
  - Use primitive types instead of objects
  - Do not depend entirely on the garbage collector
  - Avoid creating many objects quickly
  - Reuse objects as much as possible
  - Move heavy processing to the server rather then have it processed on the device

Best Practices for BlackBerry Developers (Cont.)

- Avoid returning null
  - If your application does not expect a null return value, the method should throw an appropriate exception. This forces the caller of the method to deal explicitly with the problem.
- Avoid passing null into methods
  - Do not pass null parameters into an API method unless the API Reference states explicitly that the method supports them.

Best Practices for BlackBerry Developers (Cont.)

- Use caution when passing null into a constructor
  - By casting null to the appropriate object, you indicate precisely which constructor the compiler should use.
- Use longs for unique identifier

Best Practices for BlackBerry Developers (Cont.)

- Exit applications correctly
  - Before invoking System.exit(int status), perform any necessary cleanup, such as removing objects from the runtime store that applications no longer require.
- Print the stack trace
  - Don't catch the exception, but rather catch a Throwable instance

PROGRAMMING USING THE BLACKBERRY JAVA DEVELOPMENT ENVIRONMENT (JDE)

Developing for the BlackBerry

- Unlike MIDlets, BlackBerry applications do not have a lifecycle
- Applications are compiled into .cod and .alx files rather than .JAD and .JAR
- In addition to full support of standard CLDC and MIDP APIs, RIM provides BlackBerry-specific extensions that enable you to develop applications with the look and feel of native applications

Developing for the BlackBerry (Cont.)

- The BlackBerry APIs provide tighter integration for BlackBerry devices, and access to BlackBerry features for user interface, networking, and other capabilities
- CLDC, MIDP, and BlackBerry APIs may be used together in the same application – with the notable exception of user-interface APIs
- RIM's UI APIs provide greater functionality and more control over the layout of your screens and fields, but at a cost: Resulting MIDlets will be non-standard, so porting to other platforms will require more effort

Getting Started

- Create a workspace
  - Where all your projects are stored
- Create a project
- Create new source files
- Building applications
  - Applications must be built before they can be executed
- Running applications
- Debugging applications

Creating a Workspace

- Create a workspace
  - On the File menu, click New Workspace.
  - In the Workspace name field, type a workspace name.
  - In the Create in this directory field, specify a directory in which to save the workspace. If you specify a folder that does not exist, the BlackBerry® IDE creates it for you.
  - Click OK.

Creating a Project

- Create a project
  - On the Project menu, click Create New Project.
  - In the Project name field, type a project name without a file name extension.
  - In the Create project in this directory field, type the folder in which to create the project file.
  - Click OK.

Creating Project Source Files

- Create a source file within a project
  - On the Project menu, click Create New File in Project.
  - In the Source file name field, type a file name without the .java file name extension.
  - In the Create source file in this directory field, type a folder name.
  - Click OK

Building Projects

- Projects may be built one at a time or all at once.
- Build a project.
  - In the Workspace files window, click the project folder.
  - On the Build menu, click Build Selected.
- Build all projects.
  - On the Build menu, click Build All.
- Note: To exclude projects from Build All, click the General tab on the project Properties window. Select Excluded from Build all.

Running Applications

- Projects and applications may be executed and tested using the built-in simulator
  - On the Debug menu, click Go.
  - The BlackBerry IDE builds all active projects in the workspace and loads the applications in the BlackBerry Device Simulator.
- Application will be loaded onto Simulator and all events are printed in the Debug window.

Debugging Applications

- Start by setting a few breakpoints at critical sections of your code.
  - On the Debug menu, select Breakpoint, click Set Breakpoint at cursor.
- Gradually set more breakpoints at shorter intervals to identify the problem.
- To identify the problem, after the program has paused at a breakpoint, use debugging tools to view various application processes and statistics.
- To debug your application during run time, in the main window, on the Debug menu, click Break Now.

HelloWorld Example

HelloWorld.java

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;

public class HelloWorld extends UiApplication {
    public static void main(String[] args) {
                        HelloWorld hello = new HelloWorld();
            hello.enterEventDispatcher();
    }

    public HelloWorld() {
            pushScreen(new MScreen());
    }
}
```

HelloWorld Example (Cont.)

```
final class MScreen extends MainScreen {

    public MScreen() {
            setTitle(new LabelField("Hello World!", LabelField.ELLIPSIS |
    LabelField.USE_ALL_WIDTH));
            add(new RichTextField("Hello World!" ,Field.NON_FOCUSABLE));
    }

    public boolean onClose() {
            Dialog.alert("Good bye!");
            System.exit(0);
            return true;
    }
}
```

HelloWorld Explained

- import net.rim.device.api.ui.*;
  import net.rim.device.api.ui.component.*;
  import net.rim.device.api.ui.container.*;
  - Standard import packages required for the application’s user interface

- hello.enterEventDispatcher();
  - To make the application enter the event thread and start processing messages

HelloWorld Explained (Cont.)

- pushScreen(new MScreen());
  - The BlackBerry uses a stack-like process of displaying screens. This will push the MScreen instance onto the UI stack to be displayed.

- setTitle(new LabelField("Hello World!“));
  - Adds a field to the title region of the screen with the text “Hello World!”.

HelloWorld Example (Cont.)

- add(new RichTextField("Hello World!"));
  - Add a read only RichTextField text field to the screen with the message "Hello World!".

- public boolean onClose() {

  Dialog.alert("Good bye!");
  - Overrides the onClose() method from superclass Screen to display the message "Good bye!" when the application closes.

THE BLACKBERRY MOBILE DATA SYSTEM (MDS)

MDS

- What is it?
  - Development framework with tools to allow for the building and deploying of applications for the BES (BlackBerry Enterprise Solution)
- Purpose
  - Allows administrators to deploy, manage, install, upgrade, and remove applications on a BlackBerry wirelessly through the use of BES administration tools

MDS - Benefits

- Makes sure that technologies and systems are reusable
- Reduces development time and complexity
- Optimize wireless data transmission for increased performance and lower operating costs
- Provides centralized application management over the wireless network

MDS – Benefits (Cont.)

- Simplifies development through the use of transparency
- Allows for push-based and pull-based access to systems and databases using existing back-end architectures

MDS - Security

- Triple DES/AES
  - Encrypts communication between the devices and the BES
- RSA /DSA keys
  - Authentication messages between BlackBerry MDS Studio applications and web services
  - Only possible if the web service has access to a Certificate Authority

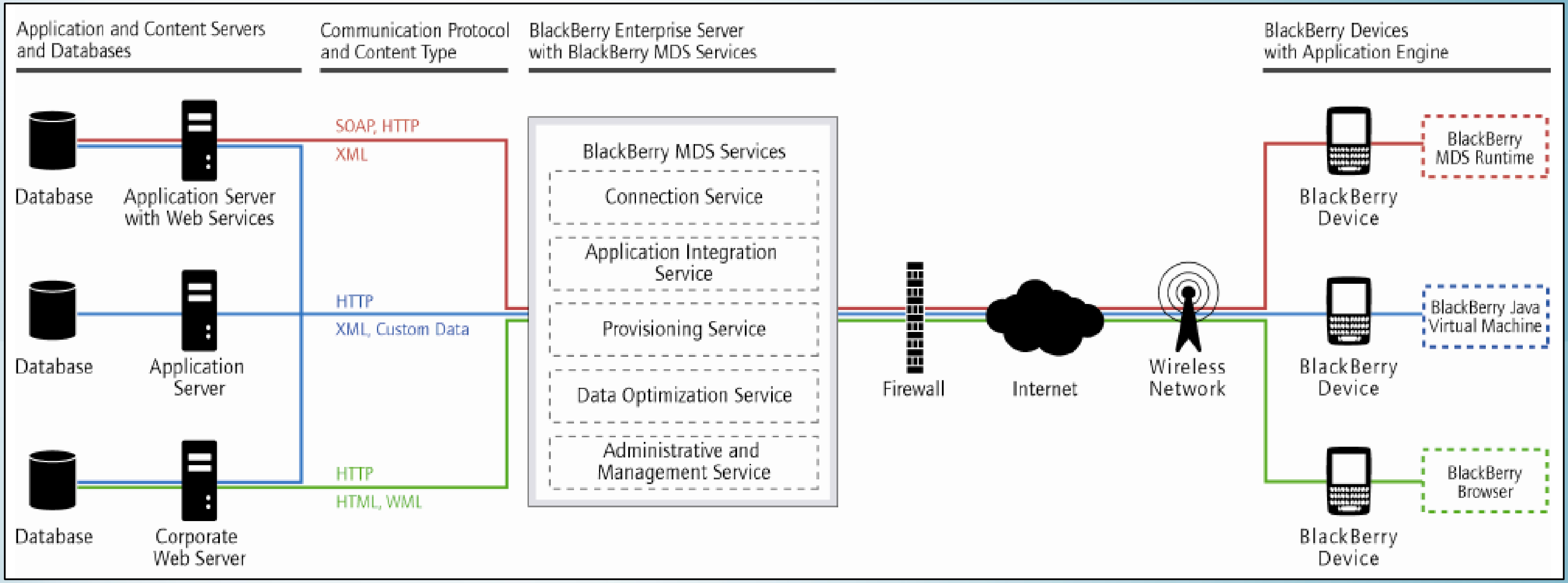
MDS – Security (Cont.)

- Supports various security standards
  - TLS
  - SSL
  - S/MIME
  - IT security policies
  - Code signing
  - Certificates

MDS Runtime Environment

- Components
  - Application/content servers/databases
  - Communication protocol and content types
  - BES with BlackBerry MDS Services
  - BlackBerry MDS Device Software

MDS Runtime Environment (Cont.)



Application and Content Servers and Databases
Communication Protocol and Content Type
BlackBerry Enterprise Server with BlackBerry MDS Services
BlackBerry Devices with Application Engine
Database
Application Server with Web Services
Database
Application Server
Database
Corporate Web Server
SOAP, HTTP
XML
HTTP
XML, Custom Data
HTTP
HTML, WML
BlackBerry MDS Services
Connection Service
Application Integration Service
Provisioning Service
Data Optimization Service
Administrative and Management Service
Firewall
Internet
Wireless Network
BlackBerry Device
BlackBerry Device
BlackBerry Device
BlackBerry MDS Runtime
BlackBerry Java Virtual Machine
BlackBerry Browser

MDS Runtime Environment (Cont.)

- Application and content servers and databases reside on servers behind a firewall
- The MDS Services manage connections between the application/content servers/databases and the BlackBerry devices

MDS Runtime Environment (Cont.)

- Uses the protocols to communicate with various servers:
  - Simple Object Access Protocol (SOAP) and HTTP
  - Custom data and HTTP
  - Extensible Markup Language (XML) and HTTP
  - HTML and HTTP
  - Wireless Markup Language (WML) and HTTP

BlackBerry MDS Services

- Purpose of the BlackBerry MDS is to extend enterprise systems to BlackBerry devices thus it contains the following services:
  - BlackBerry MDS Connection Service
    - Provides TCP and HTTP-based connectivity between mobile applications and enterprise applications that reside behind the firewall.
    - Provides an encrypted connection between devices and the corporate network
    - Users browsing web content use the same connection

BlackBerry MDS Services (Cont.)

- BlackBerry MDS Application Integration Service
  - Supports web services and other standard mechanisms for integrating mobile applications with enterprise applications.
  - Manages transmitting application data messages between BlackBerry MDS Studio applications and back-end systems.
- BlackBerry MDS Provisioning Service
  - Controls which applications users can download to their devices and manages the installation of applications on the devices over the wireless network.

BlackBerry MDS Services (Cont.)

- BlackBerry MDS Data Optimization Service
  – Transforms existing server-side content and data into a format that is optimized for efficient wireless transmission and for use on mobile devices.
- BlackBerry MDS Administrative and Management Service
  – Manages policies, such as application availability to users and services availability to applications.
  – Centralizes application lifecycle management.

Required Software

- BlackBerry MDS Runtime
  - Provides runtime services for applications developed with the BlackBerry MDS Studio.
- BlackBerry Java Virtual Machine (BlackBerry JVM)
  - Provides a secure environment on BlackBerry devices for running custom Java applications created with the BlackBerry JDE.

Required Software (Cont.)

- The BlackBerry Browser
  - Enables access to web-based applications and supports Internet standards, such as HTML, WML, Common Gateway Interface, ASP, and JSP.
  - Included with the BlackBerry Enterprise Solution and runs on top of the BlackBerry JVM.

Push Technology

- Developers and administrators can proactively send data to users as it is required
- Server-side applications can send automatic updates and alerts when data changes, without users requesting the new content.
- Used with MDS, developers can design applications that listen silently in the background for push requests

Push Technology (Cont.)

- Allows applications to send web pages or alerts to the device so that they appear as messages in the inbox of the BlackBerry
- Applications can also create and update channels which appear on the home screen as an icon.
  - Enables users to receive updates to certain types of data (acts like a bookmark to important content)

Push Technology (Cont.)

- Web Pages are pushed directly to the browser's cache
  - User receives no indication that the content is updated
  - Next time when the user visits the specified web page, the data is retrieved from the cache
  - Results in faster access to the latest data on the device when not in a wireless coverage area

Advantages of Push Technology

- Use the network efficiently
  - Only sends data when it has changed
  - Users are not required to manually check for updates

- Increases information availability and timeliness
  - Information can be cached on the device so users can recover data even if they are outside coverage area

Application Models

- Browser Model
  - BlackBerry MDS Browser Applications
- Web Service Model
  - BlackBerry MDS Studio Application
- Client/Server Model
  - BlackBerry MDS Java Application

Browser Model

- Provides mobile access to intranet content
- Create web-based applications that are accessible over the wireless network through the BlackBerry Browser
- Use existing web content development tools

Web Service Model

- Create lightweight applications that provide mobile access to enterprise web services
- Provides design flexibility
- Use the BlackBerry MDS Studio
  - Visual application design and assembly tool

Web Service Model (Cont.)

- Requires less coding then traditional client/server application models
  - Web service, MDS runtime, MDS Services provides most of the functionality

- Enables developers to quickly create client applications that integrate with enterprise web services using a component-based drag-and-drop approach.

Client/Server Model

- Create custom Java applications for the BlackBerry device that provide a sophisticated user interface (UI) and navigation, robust data management, and flexible support for custom data formats
- Use BlackBerry JDE
  - Provides development and simulation tools that enable developers to build rich-client Java ME applications, which integrate with standard application servers, for BlackBerry devices.

MDS – Studio Application

- Uses Web Service Model
- Combines the lightweight MDS browser applications with the functional MDS Java application
- Allows users to access the functionality of remote web services from their devices
  - Access these web services through HTTP or HTTPS and communicates with them using SOAP

MDS – Studio Application (Cont.)

- Enables developers to produce applications with the same performance, responsiveness, and user experience of traditional Java client application, but in less time, without the need for extensive coding knowledge, and less load on the processing and memory constraints of the BlackBerry device