

J2ME Connected Applications

Santanu Dutt
Senior Consultant
PlatinumSolutions Inc.



Evolution of J2ME

- Antero Taivalsaari is the father of KVM
- 'Spotless' was the research project
- KVM is the product
- KVM is about 70 – 80 kilobytes in size
- Implemented in C language

What is KVM?

In Antero's own words -

KVM is a virtual machine whose size is measured in kilobytes rather than megabytes. There is another interpretation that comes from the name "Kauai," the code name for the product project when it started at Java Software. People can call it the "KVM" or the "K Virtual Machine," but not the "Kilo Virtual Machine".

What is J2ME?

- Unlike J2SE, J2ME is not a piece of software or one single specification
- J2ME is a collection of technologies & specs
- It comprises of *Configurations, Profiles & Optional packages*
- *J2ME consists of mainly 2 broad configurations – CLDC & CDC*
- *Profiles are built upon configurations and add more specific APIs*
- *Optional packages span across all profiles and add capabilities from Bluetooth communications to web services and SMS*

CLDC & MIDP

- Connected, Limited Device

Configuration (CLDC) is for small devices like cell phones, pagers & PDAs.

- It is not a complete dev environment by itself
- Mobile Information Device Profile (MIDP) is the first complete profile in J2ME built upon CLDC
- MIDP provides UI, network connectivity (GCF), local data storage (RMS) and application management
- CLDC with MIDP is targetted for devices with 16 to 32 bit CPUs and 128 to 512 KB of memory for the Java platform

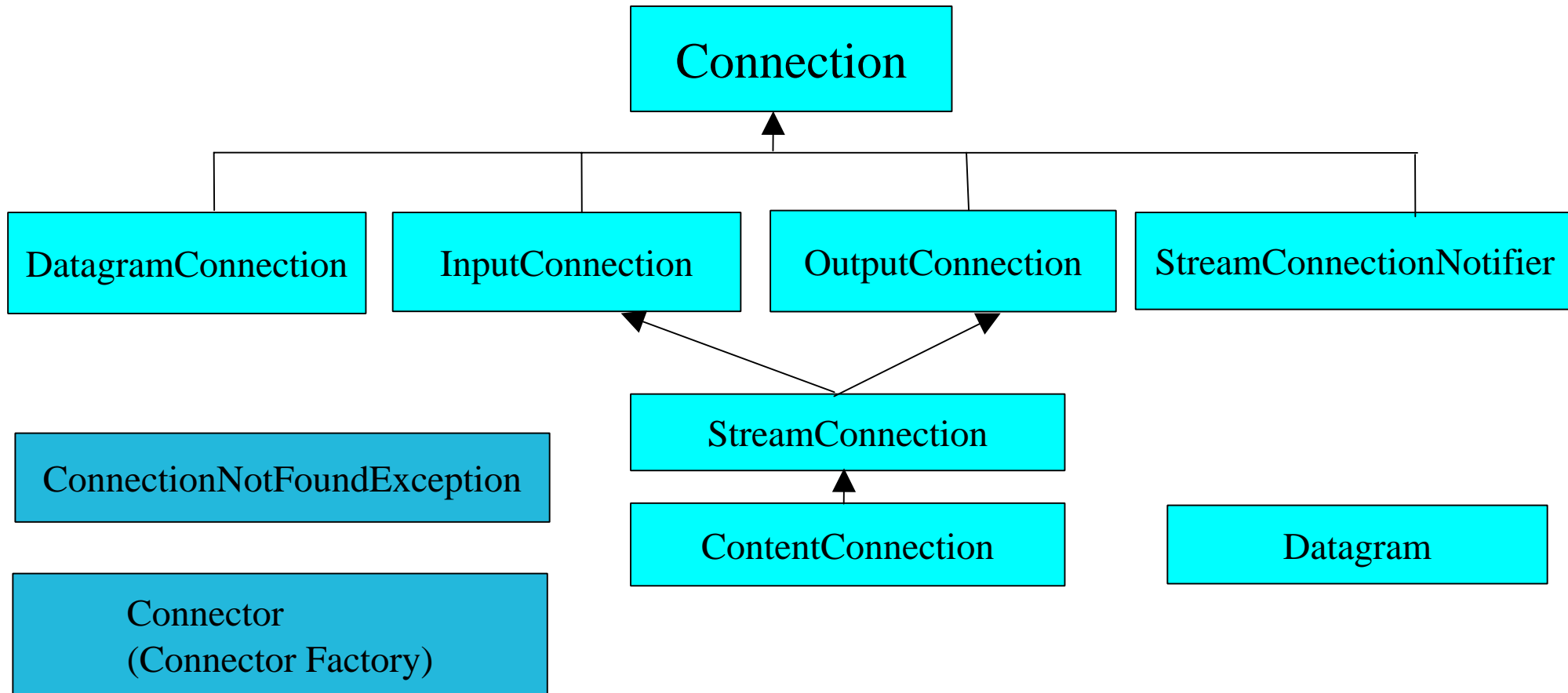
What is a MIDlet?

- MIDlet is the main building block for MIDP
- It is run & controlled by the Application Manager in KVM
- Your MIDlet class must extend the *javax.microedition.midlet.MIDlet* class
- *Methods in this class allows the AM to create, start, pause & destroy your MIDlet*
- *There are special Listener interfaces to provide MIDlet event handling*
- *These interfaces provide callback methods to handle the events*
- *MIDlets provide normal Java Exception handling*

Why Java for Wireless?

- Java is a safe platform
- Encourages robust programming – GC is a big plus
- Portability – beats any other technology
- Network aware – easy download OTA
- Network agnostic – can exchange data over many protocols like TCP/IP, WAP , i-mode spanning across many bearers like GSM, CDMA, TDMA, etc.

Generic Connection Framework



GCF continued

GCF is generic due to -

- An Interface hierarchy easily extensible
- A connection factory
- URLs to indicate the connection types

GCF supports a large number of connection types, across profiles & platforms

Profiles and the Optional packages define the actual low level implementations

GCF continued

URLs play a very important role in GCF -
URLs identify connection types and end-points

URL format -

scheme://user:password@host:port/url-path;parameters

scheme – specifies access method or protocol like SMS, HTTP, FTP, etc.

url-path – specifies the path to the resource

parameters – optional params with the url-path

GCF continued

URL & connection examples -

- Creating an `HttpConnection`:

```
String url = "http://www.server.com:80/com/myServlet";  
HttpConnection c = (HttpConnection)Connector.open(url);
```

...

- Creating an `InputConnection` (Foundation Profile and J2SE):

```
String url = "file:///myResourceFile.res";  
InputConnection c = (InputConnection)Connector.open(url);
```

...

- Creating a `FileConnection` (JSR 75):

```
String url = "file:///myResourceFile.res";  
FileConnection c = (FileConnection)Connector.open(url);
```

...

- Creating a `SocketConnection`:

```
String url = "socket://www.server.com:80";  
SocketConnection c =  
(SocketConnection)Connector.open(url);
```

GCF continued

URL Scheme	Connectivity	GCF Connection Type	Defined By...
A datagram	Bluetooth Datagram	L2CAPConnection DatagramConnecti on	JSR 82. Support is optional. All CLDC- and CDC-based profiles, such as MIDP, Foundation and related profiles, and with JSR 197, JSR 75. Support is optional.
file	File Access	FileConnection, InputConnection	MIDP 1.0, MIDP 2.0, Foundation Profile, J2SE (JSR 197). Support is required.
http	HyperText Transport Protocol	HttpConnection	MIDP 2.0. Support is required.
https	Secure HTTP	HttpsConnection	JSR 120, JSR 205. Support is optional.
sms	Short Messaging Service	MessageConnectio n	
mm s	Multimedia		
cbs	Messaging Service		
	Cell Broadcast SMS		

Wireless Clients for J2EE Apps

Wireless devices can interact with J2EE servers in 2 ways -

- The traditional browser based using WML & CHTML. Works in the same line as JSPs by creating dynamic content
- Using the standardized rich client environment in J2ME – provides much better user experience with advanced UI presentation layer & more computing facilities

J2ME Client Advantages

- Provides rich UI with more flexibility
- UI can be tailored to fit the specific devices layout providing better performance and visibility
- Allows the application to run even in disconnected mode. More power to work on the enterprise data
- Allows storing of the data using RMS
- Network capabilities on demand by user permission
- Developers may know HTTP protocol only

Design Constraints

- Limited screen size & input capabilities
- Limited processing power
- Short battery life
- Very small memory
- Limited persistent storage
- Limited networking capabilities – high latency, low bandwidth & intermittent connection

Overcome Design Constraints

- Try to avoid expensive operations like complex calculations
- Avoid unnecessary looping
- Use network connectivity only when needed
- Obtain very less data from the J2EE server
- Remain useful when disconnected
- Choose the less complicated messaging data structure

Client Side Architecture

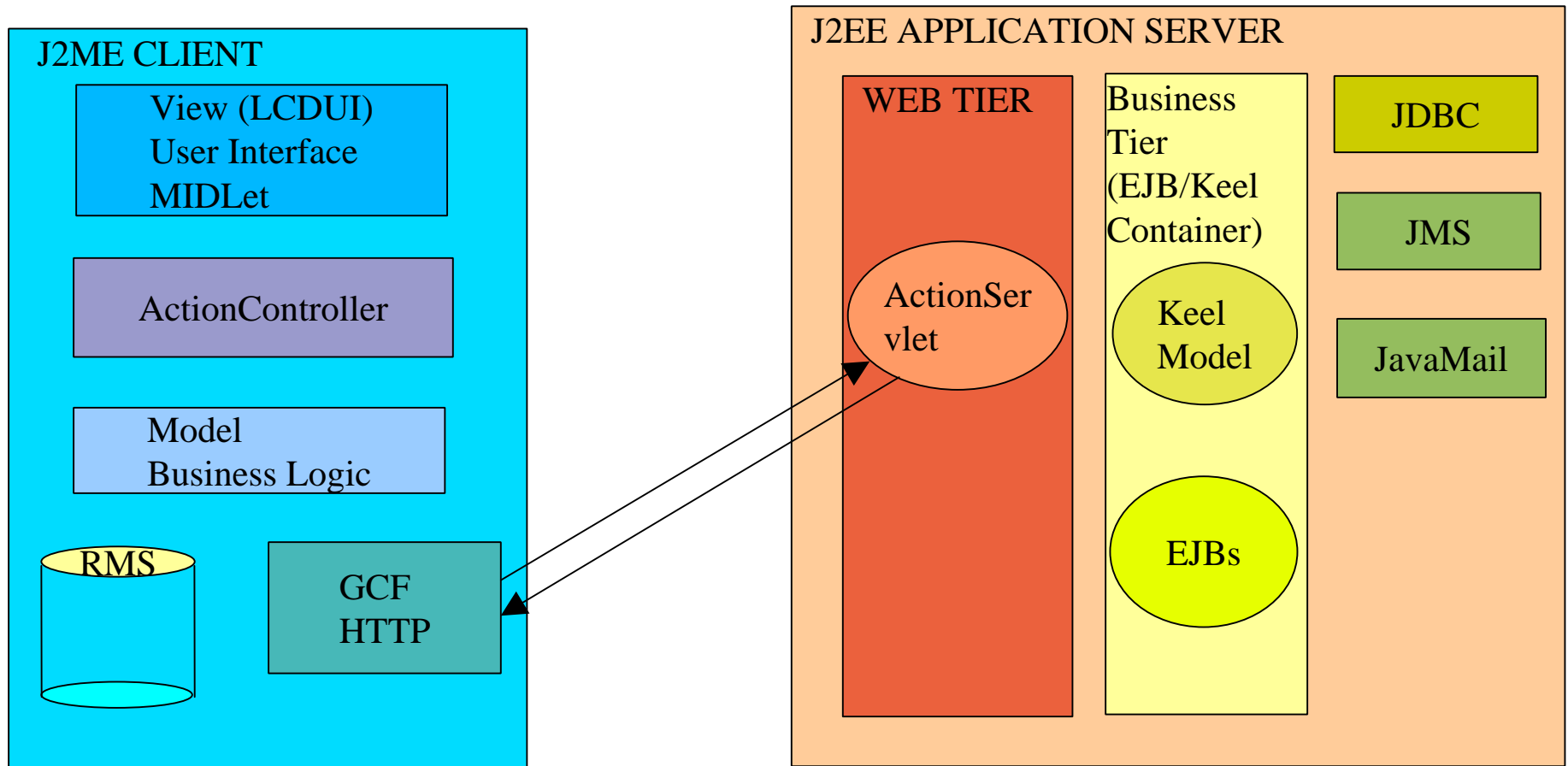
Advantages of using MVC

- Application flow is isolated from data & presentation
- Data is insulated from presentation
- Can easily swap the presentation layer
- Seamlessly switch network protocols
- Increases modularity

Disadvantage

- Increases code and jar size

Architecture of Java Wireless Enterprise Application



Messaging

Advantages of using HTTP protocol -

- All MIDP devices must support HTTP leading to maximum portability
- Firewall friendly – most firewalls allow
- HTTP is made easy by MIDP networking APIs
- Servlets are popular and extensively support Request/Response generation & handling
- MIDP 2.0 supports HTTP over Secure Socket Layer(SSL)

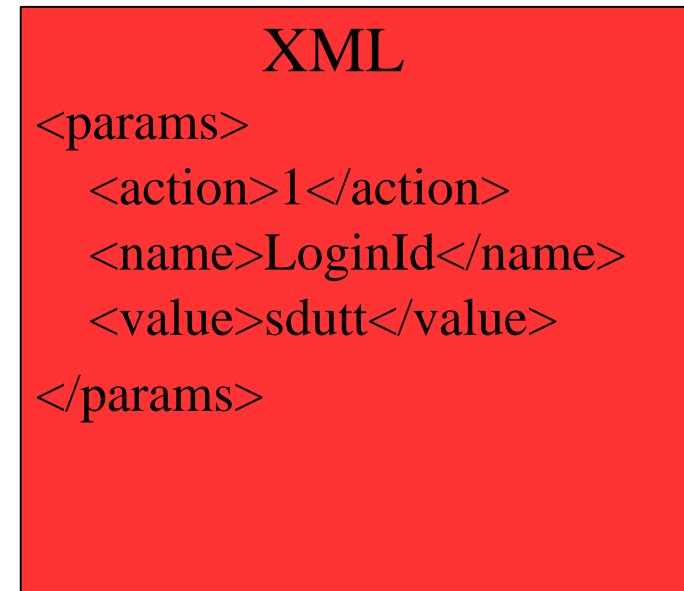
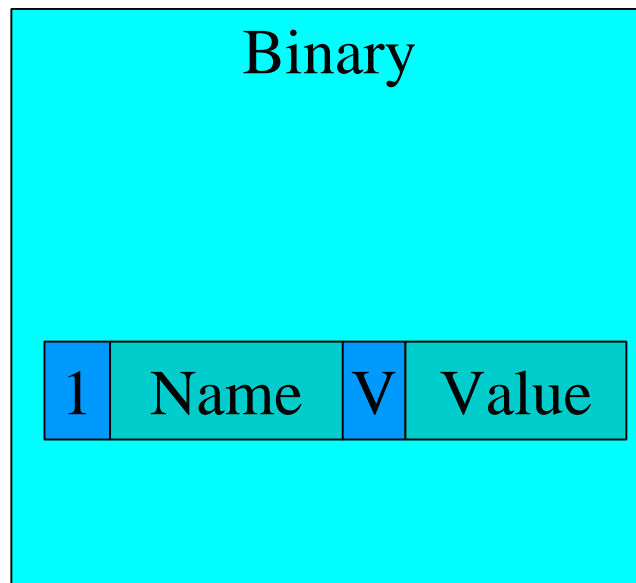
HTTP Code sample

```
private HttpURLConnection openConnection() throws IOException {  
    try {  
  
        HttpURLConnection connection =  
            (HttpURLConnection) Connector.open(this.serverURL);  
  
        connection.setRequestProperty("User-Agent",  
System.getProperty("microedition.profiles"));  
        connection.setRequestProperty("Content-Type",  
            "application/octet-stream");  
        connection.setRequestMethod(HttpURLConnection.POST);  
  
        return connection;  
    } catch (IOException ioe) {  
        ioe.printStackTrace();  
        throw ioe;  
    }  
}
```

Message Format

Falls between 2 extreme formats -

- Simple binary format
- Complex XML format



Binary Message Format

Advantages -

- Compact
- Easy to read and write using `DataInputStream` & `DataOutputStream` (`java.io`)
- Much faster processing

Disadvantages -

- Cryptic
- Both client & server needs to know the format
- Tight coupling

XML Message Format

Advantages -

- More descriptive
- Can be standardized

Disadvantages -

- XML support have to be implemented
- Slower process
- Requires more memory
- In some cases just not feasible

Security & Encryption

Why security is important?

- Data OTA can be easily intercepted
- Data in the device is also vulnerable
- HTTPs is not enough
- Applications must also be secured

Bottomline: Without proper security wireless enterprise clients are useless

Encryption

Keys – series of chars & numbers for encryption and decryption

Cipher – Algorithm to convert data to encrypted format

2 types of ciphers-

Symmetric – use the same key to encrypt & decrypt (secret key system)

Asymmetric – use different keys(public key for encryption; private key for decryption)

Bouncy Castle

Features -

- Open source API
- Java Cryptography Extension provider
- Basically provides algorithms
- Supports about 20 engines
- API is huge (about 1611KB)
- Without obfuscator not possible for use in device

Obfuscator

ProGuard is a free Java class file shrinker and obfuscator. It can detect and remove unused classes, fields, methods, and attributes. It can then rename the remaining classes, fields, and methods using short meaningless names. The resulting jars are smaller and harder to reverse-engineer.

The WTK 2.1 version has a plugin to support Proguard
URL - proguard.sourceforge.net

Managing Session State

2 ways to do this -

- Cookies – a small junk of data sent to the client from server. Client sends back in the Request header

```
// Query the server and retrieve the response.  
HttpConnection hc= (HttpConnection)Connector.open(url);  
InputStream in = hc.openInputStream();  
  
// Read the session ID from a cookie in the response headers.  
String cookie = hc.getHeaderField("Set-cookie");  
if (cookie != null) {  
    int semicolon = cookie.indexOf(';');  
    mSession = cookie.substring(0, semicolon);  
}
```

Session continued

URL Rewriting

In MIDP context the client creates a session id and sends it out on every HTTP Request. It's the responsibility of the MIDP/J2EE interface in the Web container to set the session id to establish the client context in the EJB container.

Will vary from container to container

J2EE Server Interface

Features-

- Mainly it is a servlet that intercepts the MIDP HTTP Request
- Uses the InputStream to obtain the data from Request
- If encryption is used then decrypts the data
- Parses the data to obtain the parameters
- Sets the session id to identify the client and associate it to the right context
- Formats the parameters as per container requirements

continued

J2EE Server Interface continued...

- Calls the business logic in SessionBeans or Model components
- Formats the response as per the agreed upon Message format
- Flushes out the response as OutputStream

Conclusion

Some useful links -

- <http://java.sun.com/j2me/index.jsp>
- <http://eclipseme.sourceforge.net/>
- <http://www.forum.nokia.com/main.html>
- <http://www.blackberry.com/developers/na/index.shtml>
- <http://www.palmsource.com/developers/>

Thank You All For Coming