BlackBerry.
DEVCON 2010

Automated Testing for Mobile Applications
An experimental technique

Eing Ong, Intuit Inc.

Louis daRosa, Intuit Inc.

LOVE working together.

Session outline

- Introduction
- Simulator basics
- Simulator library
- Building your mobile app library
- Writing reusable test cases
- Mobile test automation framework
- Advantages and limitations
- Q & A

Introduction

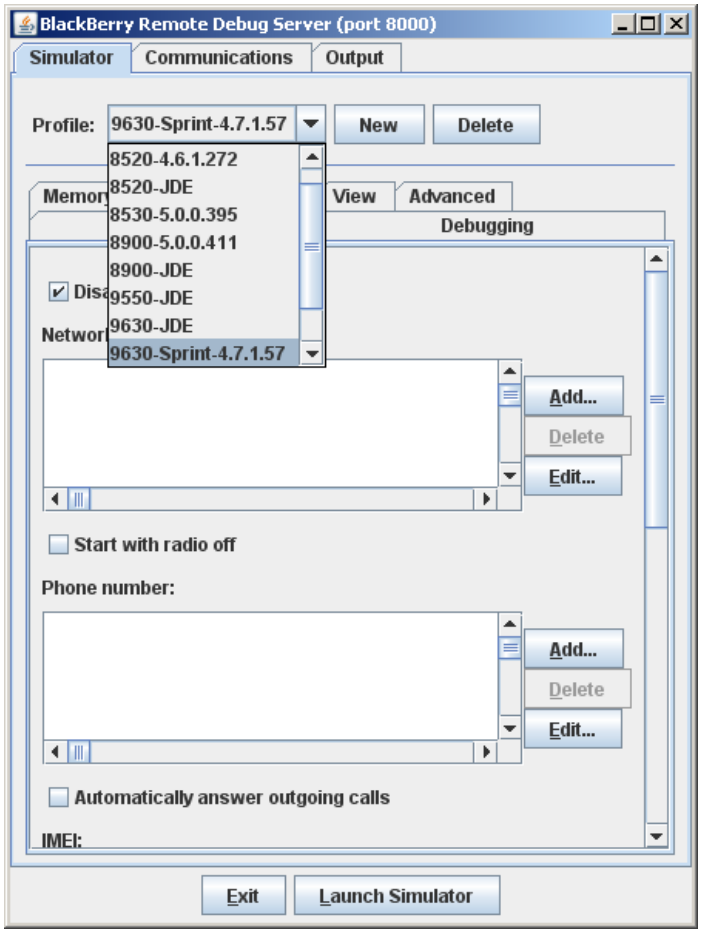
What are we solving for ?

- Various BB OS and devices
- End-to-end mobile tests
- Low cost solution
- Multiple mobile platforms

Simulator Basics

BlackBerry® JDE tool kit

- Mobile data service (MDS)
- Simulators suite
- App installation
- Image capture



Simulator Basics

BlackBerry® simulator

- **Starting simulator**

  fledge.exe
  /app=jvm.dll
  /session=<model>
  /app-param=
  JvmAlxConfigFile:<model>.xml
  /handheld=<model>

- **Communicating with simulator**

  fledgecontroller.exe
  /session=<model>

Simulator Basics

Actions	Steps
Start 9630 Tour simulator	fledge.exe /app=jvm.dll /session=9630 /handheld=9630 /app-param=JvmAlxConfigFile: 9630.xml
Install application	1. Copy app.jar, app.jad, app.cod to Javaloader directory 2. JavaLoader.exe –u load app.jad 3. Delete app.jar, app.jad, app.cod
Save screenshot as test.png in $TEST_OUTPUT	1. JavaLoader.exe –u screenshot test.png 2. mv test.png $TEST_OUTPUT

Simulator Basics

Actions	Steps	bblib.py
Start 9630 Tour simulator	fledge.exe /app=jvm.dll /session=9630 /handheld=9630 /app-param=JvmAlxConfigFile:9630.xml	fledgeStart()
Install application	1. Copy app.jar, app.jad, app.cod to Javaloader directory 2. JavaLoader.exe –u load app.jad 3. Delete app.jar, app.jad, app.cod	install()
Save screenshot as test.png in $TEST_OUTPUT	1. JavaLoader.exe –u screenshot test.png 2. mv test.png $TEST_OUTPUT	getScreenShot('test')

Simulator commands

Action	Steps
Enter 'Hello World'	StringInjection(Hello) KeyPress(SPACE) KeyRelease(SPACE) StringInjection(World)
Backspace 2 times	KeyPress(BACKSPACE) KeyRelease(BACKSPACE) KeyPress(BACKSPACE) KeyRelease(BACKSPACE)
Thumbwheel up twice	ThumbWheelRoll(-1) ThumbWheelRoll(-1)

Simulator commands

Action	Steps	bblib.py
Enter 'Hello World'	StringInjection(Hello) KeyPress(SPACE) KeyRelease(SPACE) StringInjection(World)	enter('Hello World')
Backspace 2 times	KeyPress(BACKSPACE) KeyRelease(BACKSPACE) KeyPress(BACKSPACE) KeyRelease(BACKSPACE)	backspaces(2)
Thumbwheel up twice	ThumbWheelRoll(-1) ThumbWheelRoll(-1)	thumbwheel ('up', 2)

Simulator commands

Action	Steps	bblib.py
Touch screen at (10, 100)	TouchScreenPress(10, 100, 0) TouchScreenClick() TouchScreenUnclick() TouchScreenUnpress(0)	touch(10, 100)

bblib.py

Setup

- fledgeStart()
- javaloaderStart()
- mdsStart()

Basic key events

- backspaces(count)
- enter()
- enter(string)
- menu()
- escape()
- pause(seconds)

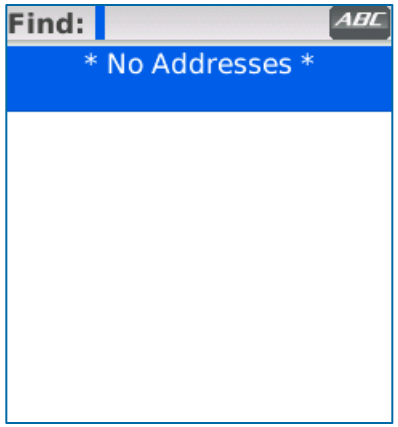
bblib.py

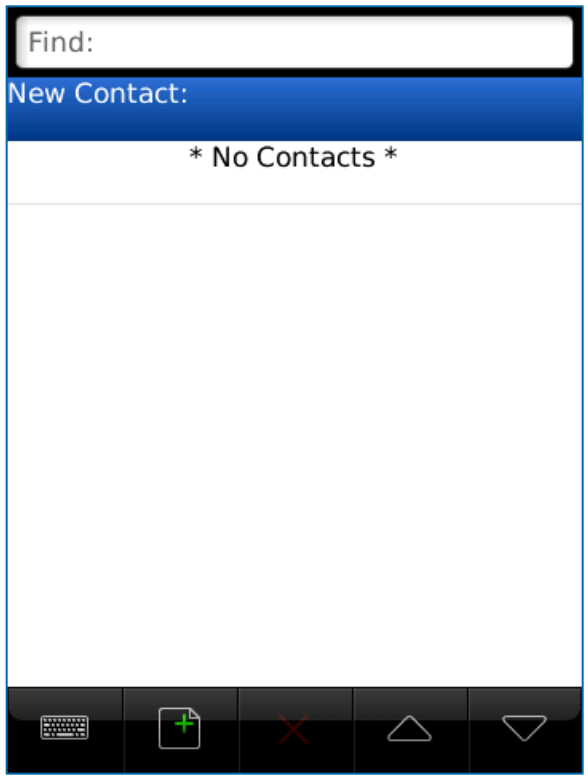
Navigation

- thumbwheel('up', count)
- thumbwheel('down', count)
- trackball('up', count)
- trackball('down', count)
- trackball('left', count)
- trackball('right', count)
- touch(x,y)

Case study – Address Book Demo

- Devices
  - Pearl
  - Storm
- Simulator library
- Address book
  - Add contact
  - Find contact
  - Delete contact
- Automated tests





Test Automation Overview

1. **Define application interface**

   This interface is device-agnostic.

2. **Implement the interface**

   Implement interface in BlackBerry®

   Utilize Python™ mobile libraries e.g. bblib.py.

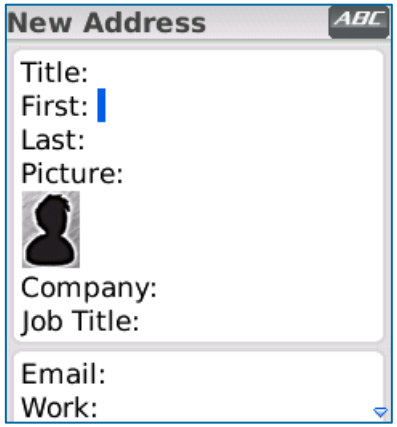
3. **Write your tests**

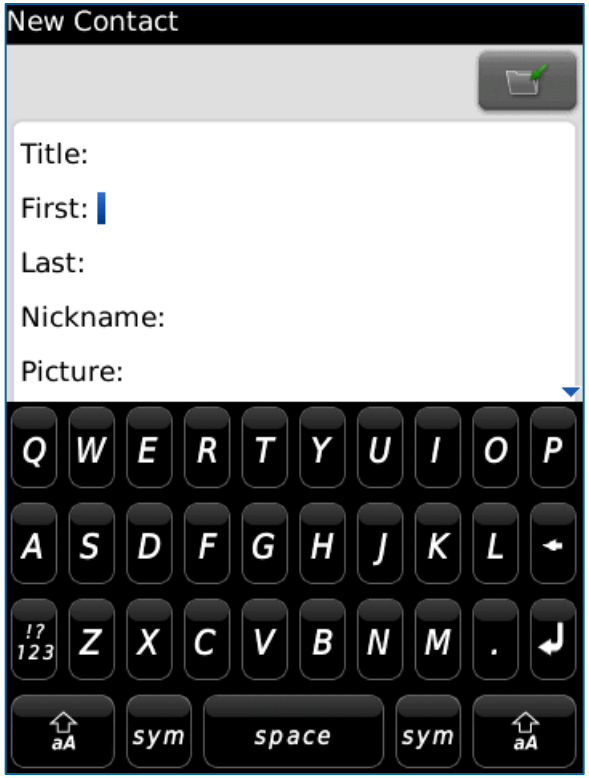
   Tests are device independent and reusable.

4. **Run**

Step 1 : Define application interface


```
class AppInterface:
    """Application Interface"""
    def  launch(self):
        """Launch app"""
    def add(self, contact):
        """Add a contact"""
    def find(self, contact):
        """Find contact"""
    def delete(self):
        """Delete current contact"""
```





Test Automation Overview

1. **Define application interface**

   This interface is device-agnostic.

2. **Implement the interface**

   Implement interface in BlackBerry®

   Utilize Python™ mobile libraries e.g. bblib.py.

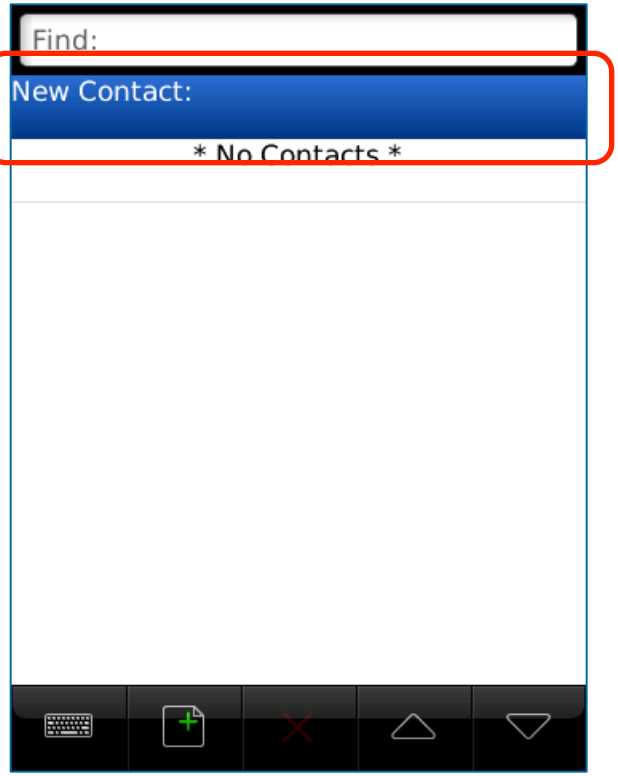
3. **Write your tests**

   Tests are device independent and reusable.

4. **Run**

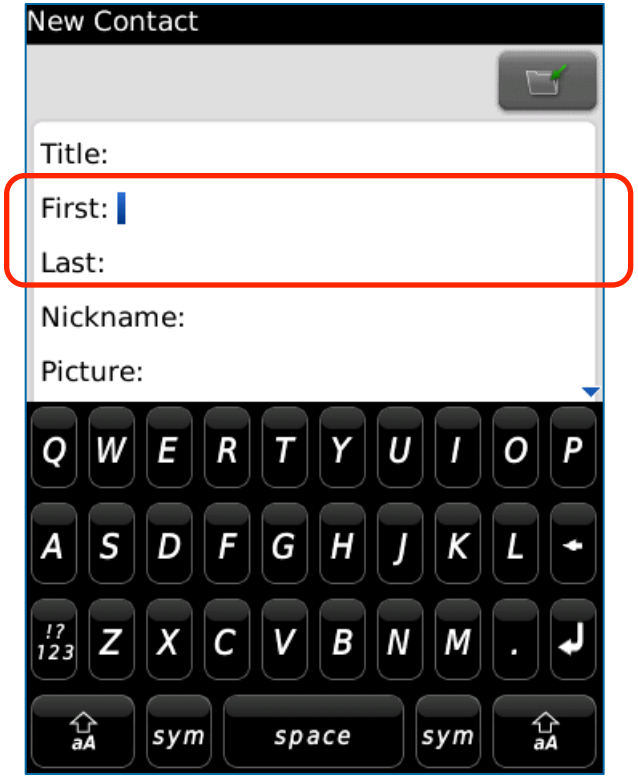
Step 2 (Storm): Implement the interface

```
def StormImpl(appbase.AppInterface):
    def add(self, contact):
        touch(100, 50)
        enter(contact.getFirstname()
        thumbwheel('down', 1)
        enter(contact.getLastname())
        # save
        menu()
        touch(50, 300)
```



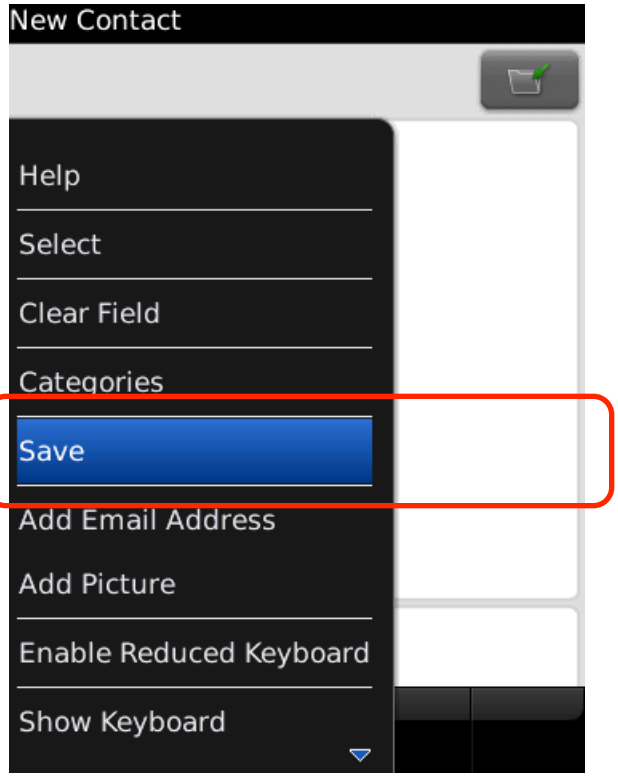
Step 2 (Storm): Implement the interface

```
def StormImpl(appbase.AppInterface):
    def add(self, contact):
        touch(100, 50)
        enter(contact.getFirstname()
        thumbwheel('down', 1)
        enter(contact.getLastname())
        # save
        menu()
        touch(50, 300)
```



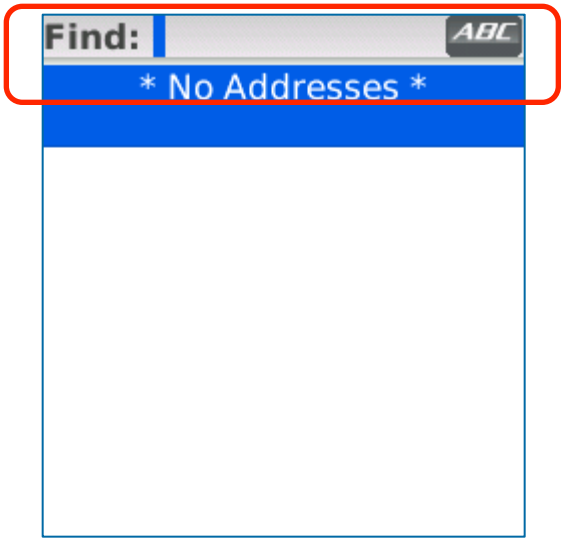
Step 2 (Storm): Implement the interface


```
def StormImpl(appbase.AppInterface):
    def add(self, contact):
        touch(100, 50)
        enter(contact.getFirstname()
        thumbwheel('down', 1)
        enter(contact.getLastname())
        # save
        menu()
        touch(50, 300)
```



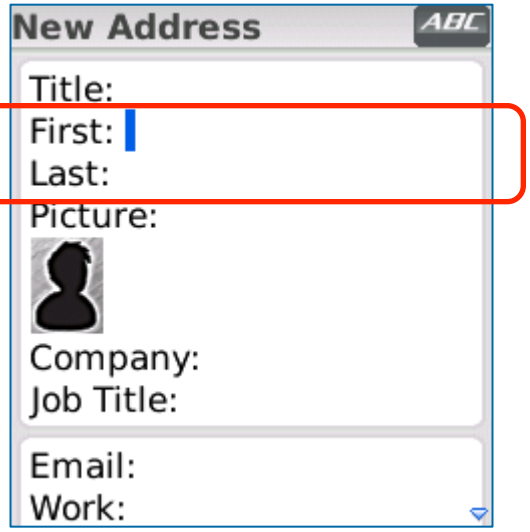
Step 2 (Pearl): Implement the interface


```
def PearlImpl(appbase.AppInterface):
    def add(self, contact):
        enter()
        enter(contact.getFirstname()
        thumbwheel('down', 1)
        enter(contact.getLastname())
        # save
        menu()
        enter()
```



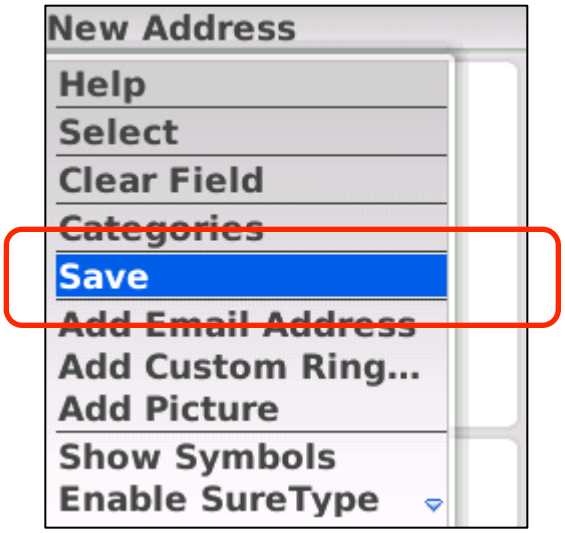
Step 2 (Pearl): Implement the interface

```
def PearlImpl(appbase.AppInterface):
    def add(self, contact):
        enter()
        enter(contact.getFirstname()
        thumbwheel('down', 1)
        enter(contact.getLastname())
        # save
        menu()
        enter()
```



Step 2 (Pearl): Implement the interface

```
def PearlImpl(appbase.AppInterface):
    def add(self, contact):
        enter()
        enter(contact.getFirstname()
        thumbwheel('down', 1)
        enter(contact.getLastname())
        # save
        menu()
        enter()
```



Step 2 Recap: Implement the interface


```
def StormImpl(appbase.AppInterface):
    def add(self, contact):
        touch(100, 50)
        enter(contact.getFirstname()
        thumbwheel('down', 1)
        enter(contact.getLastname())
        # save
        menu()
        touch(50, 300)
```

```
def PearlImpl(appbase.AppInterface):
    def add(self, contact):
        enter()
        enter(contact.getFirstname()
        thumbwheel('down', 1)
        enter(contact.getLastname())
        # save
        menu()
        enter()
```

Test Automation Overview

1. **Define application interface**

   This interface is device-agnostic.

2. **Implement the interface**

   Implement interface in BlackBerry®

   Utilize Python™ mobile libraries e.g. bblib.py.

3. **Write your tests**

   Tests are device independent and reusable.

4. **Run**

Step 3 : Writing tests


```
class AddContactTest(unittest.TestCase):

    device = testenv.getDeviceClass()

    def addContactWithOnlyFirstnameTest(self):
        self.contact.setFirstname(firstname)
        self.device.add(self.contact)

    def addContactWithOnlyLastnameTest(self):
        self.contact.setLastname(firstname)
        self.device.add(self.contact)
```

Step 3 : Runtime device binding


```
def getDeviceClass(self):
        """ Returns the device to test """

        mobileDevice = self.getMobileDevice()

       if mobileDevice == 'pearl':
            import pearl
            deviceClass = pearl.PearlImpl()
        elif mobileDevice == 'storm':
            import storm
            deviceClass = storm.StormImpl()
        else:
            import bb
            deviceClass = bb.BlackBerry()

        return deviceClass
```

More device-independent tests

Additional tests are easy to write

```
def addContactWithEmailTest(self):
def addContactWithAddressesTest(self):
def addContactWithAllDetailsTest(self):
def addContactWithLongDetailsTest(self):
def addContactAddressWithStateZip(self):
def addContactAddressWithCityStateZip(self):
def addContactAddressWithNoDataNegativeTest(self):
```

Step 4 : Run

- Basic run command
  - python <test.py>
- Python™ test frameworks
  - unittest
  - PyUnit
  - python-nose

Test Verification

- Server hosted apps
  - API assertions
  - Database assertions
- Image assertions

```
self.assertTrue(imagelib.compare(self.device, testname, '100%x90%‘))

imagelib.py :

    def compare(device, imageName, crop=None, tolerance=500)
```

Logging

- Initialization

```
self.log = self.device.initLogger(self._testMethodName,
                                  self.__class__.__name__)
```

- Usage

```
self.log.info('Starting test: ' + self._testMethodName)
self.log.debug(self.contact)
self.log.error('Missing image to compare')
```

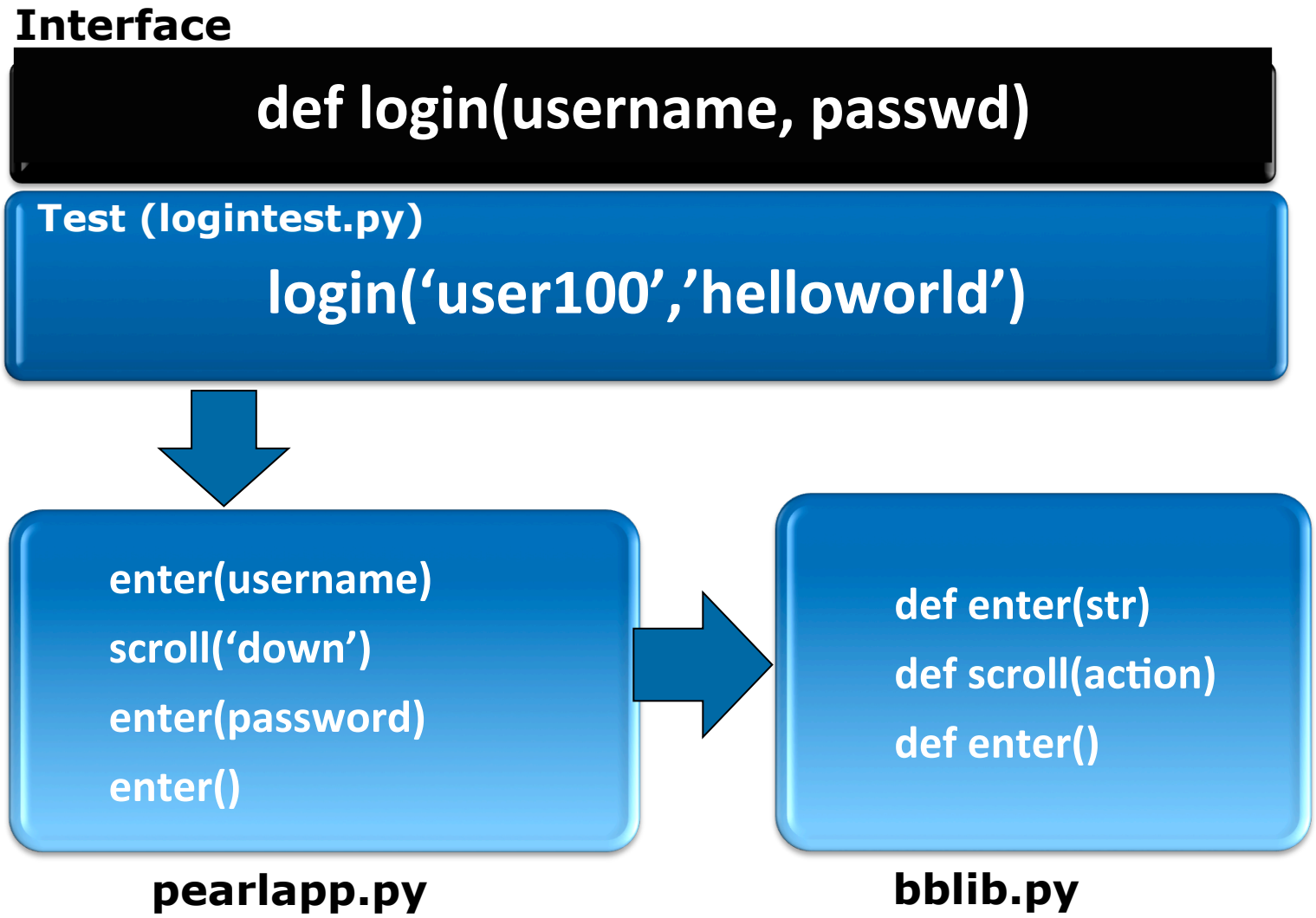
- Logs

AddressTest.log :

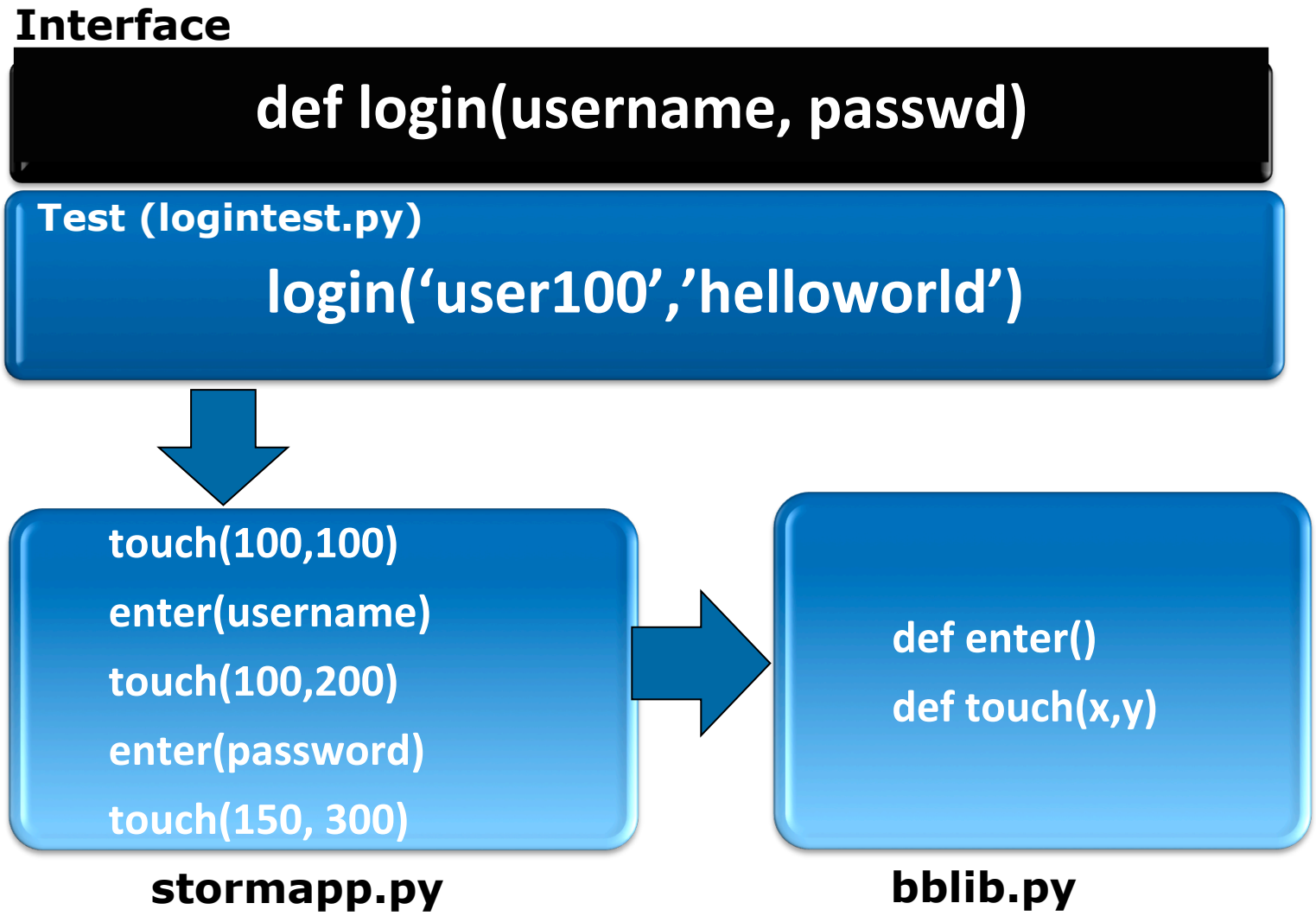
*2010-06-10 15:19:46,773 - **testCreateAddressMethod** - INFO -*

[Address1] 200 Villa St Mountain View CA 94040 BUSINESS ADDRESS

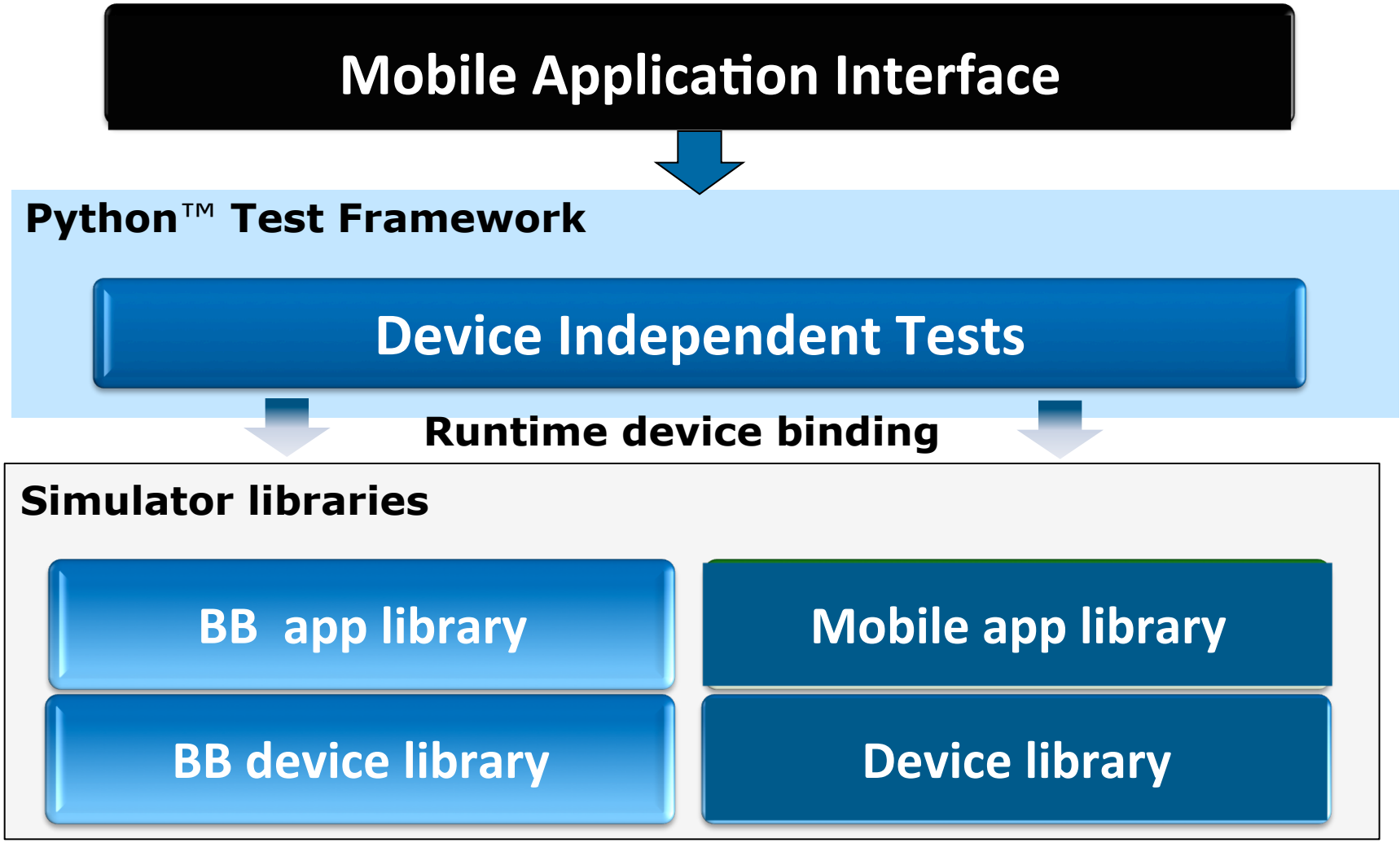
Login example on BlackBerry® Pearl



Login example on BlackBerry® Storm



Mobile Testing Framework



Advantages

- Zero cost to use
- No device sharing/scheduling
- Bigger device pool
- Reduce manual testing time
- Run on developers machines
- Debugging capabilities

Limitations

- Requires ethernet or internet connectivity
- Does not simulate network performance
- Does not support hardware controls testing
- Dependent on simulator reliability
- Limited peer-to-peer applications testing

Resources

BlackBerry® JDE Downloads

http://na.blackberry.com/eng/developers/javaappdev/javadevenv.jsp

Documentation

Fledge controller

http://docs.blackberry.com/en/developers/deliverables/15476/Test_BBSmrtphnSmltr_events_using_script_607587_11.jsp

BlackBerry® Simulator Testing

http://docs.blackberry.com/en/developers/deliverables/6338/Testing_apps_using_the_BBSmrtphnSmltr_607559_11.jsp

Python ™

http://docs.python.org/

BlackBerry® Developers Forum

http://supportforums.blackberry.com/

Acknowledgments

- Desiree Gosby, Mobile Architect
- Paul Hau, QA Manager
- Jaron Jones, QA Manager
- Jason Pugh, Architect (definitely not least)

Q & A

BlackBerry.
DEVCON 2010

Thanks and enjoy the rest of DevCon 2010!

For more details on the presentation, contact devcon@intuit.com