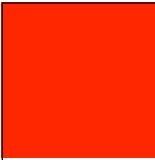




**ORACLE<sup>®</sup>**

## **Oracle Application Express on the iPhone**

**Brian T Spendolini**  
**Enterprise Solutions Group**  
**Oracle USA**



The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

# Agenda

- Why Apex on the iPhone
- Mobile Safari Development
  - Benefits
  - The two methods
  - Tools needed
- The two Methods in practice
  - Setting up APEX
  - Coding your app
  - Testing

# Why APEX on the iPhone?

- 40 Million iPhones/iPod Touches sold as of WWDC 2009
- iPhone in 81 countries, iPod touch world wide
- Great way to get your information/app out there
- Excellent source of revenue for your customers/consulting engagements
- Leverage the iPhone built in apps (google maps, phone, contacts, calendar...)
- Its cool

# Mobile Safari Development - Benefits

- One browser
  - One set of CSS
  - One set of templates
  - One set of script files
  - Ability to use iPhone apps in your app (google maps, youTube, etc.)
- Webkit
  - Amazing framework
  - Advanced toolset to work with (CSS 2.1)
  - Fast rendering times (JS/CSS)

# Mobile Safari Development - Benefits

Wait!!!! What is webkit?!?!?!?!?

- WebKit began in 2002 when Apple Inc. created a fork of the KDE project's HTML layout engine KHTML and KDE's JavaScript engine (KJS)  
wikipedia
- Has advanced HTML and CSS support
- SquirrelFish JavaScriptEngine
- Safari/Chrome is based on it
- For Win/mac

- 
- Webkit demo
    - web stickies

# The Two Methods

- The “Amazon.com” method
  - Minimized version of website
  - Leaves only key functionality
    - Lightweight website
    - Few or no images
  - Leverages Webkit/Safari
    - CSS 2.1+
    - Javascript

# Amazon.com Method

- Use Case – CRM Application
  - Focus on **a few** pieces of functionality
    - Customer lookup/Sales & Product Info
    - Get customer info on the go (names, addresses, phone #'s)
  - Use APEX/Database build in features
    - Secure with VPD
    - APEX computations/session variables
    - Browser/app based redirects
    - Templates/APEX framework

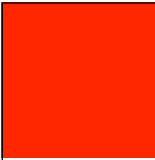
- 
- Method 1 Demo
    - Amazon.com

# The Two Methods

- iPhone Native App Method
  - Mimics the look and feel of a native iPhone app(mail, iCal)
  - Leverages Webkit/Safari as well as iPhone specific JS/Style Sheet/API libraries
  - Minimal Functionality

# iPhone Native App Method

- Use Case – CRM Application
  - Focus on 1 piece of functionality
    - Customer lookup
    - Get customer info on the go (names, addresses, phone #'s)
  - Use APEX/Database build in features
    - Secure with VPD
    - APEX computations/session variables
    - Browser/app based redirects
    - Templates/APEX framework

- 
- Method 2 Demo
    - soccer app
    - Smugmug.com

# Tools Needed

- I'm a Mac
  - iPhone SDK
  - Dashcode
  - Download Apple's Sample Code
    - Apple iPhone Dev Center
    - Mobile Safari Dev Center
  - Testing your app
    - Safari
    - iPhone Simulator
    - Personal iPhone



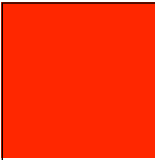


THE LIGHT WORKS  
VISUAL IMAGERY

# Tools Needed

- **Wait!!! I'm a PC!**
  - No Problem.
  - Download Apple's Sample Code
    - Apple iPhone Dev Center
    - Mobile Safari Dev Center
  - Safari for Windows for testing
  - iPhone for testing





# The Two Methods in practice

## The “Amazon.com” method

What do we have to work with?



Status bar: 20 pixels

URL text field: 60 pixels

Visible area for your content: 320 x 356 pixels

Button bar: 44 pixels

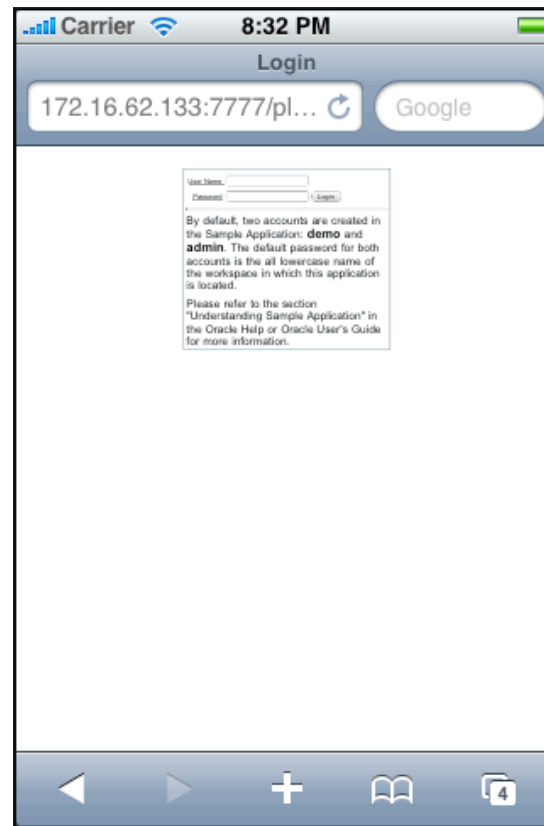
480 pixels

# Working with Apex Pages

- Create a custom page template
- Add iPhone meta tags
- Make width 320px
- Increase Font Sizes
- Careful with tabs
- Choose Content carefully
  - Master/Detail reports
  - No multi column reports
  - Value Attribute Pairs template works well

# Working with Apex Pages

- Let's start with the sample apex app
  - Login page



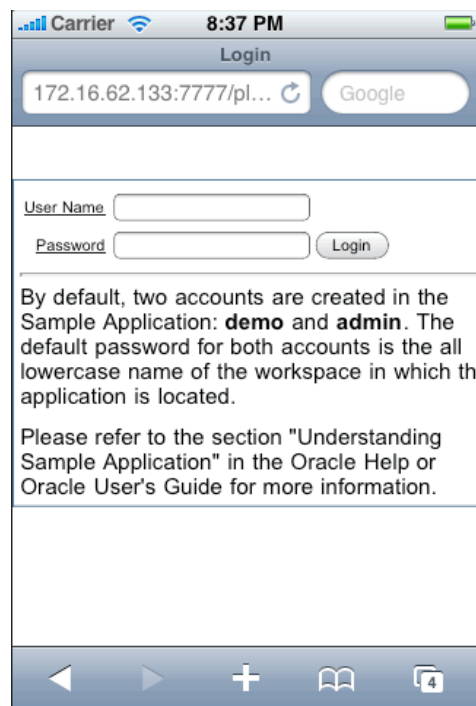
# Working with Apex Pages

- Start with the login template
  - Add iPhone specific meta tags

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

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

```
<meta name="apple-mobile-web-app-capable" content="yes" />
```



# Working with Apex Pages

- Style sections with CSS
  - First....what are CSS?

**Cascading Style Sheets (CSS) are logic on how a element on a web page is presented.**

Example : Change the font size and color of a form label.

```
.formLabel {  
    font-size:14px;  
    color: red;  
}
```

# Working with Apex Pages

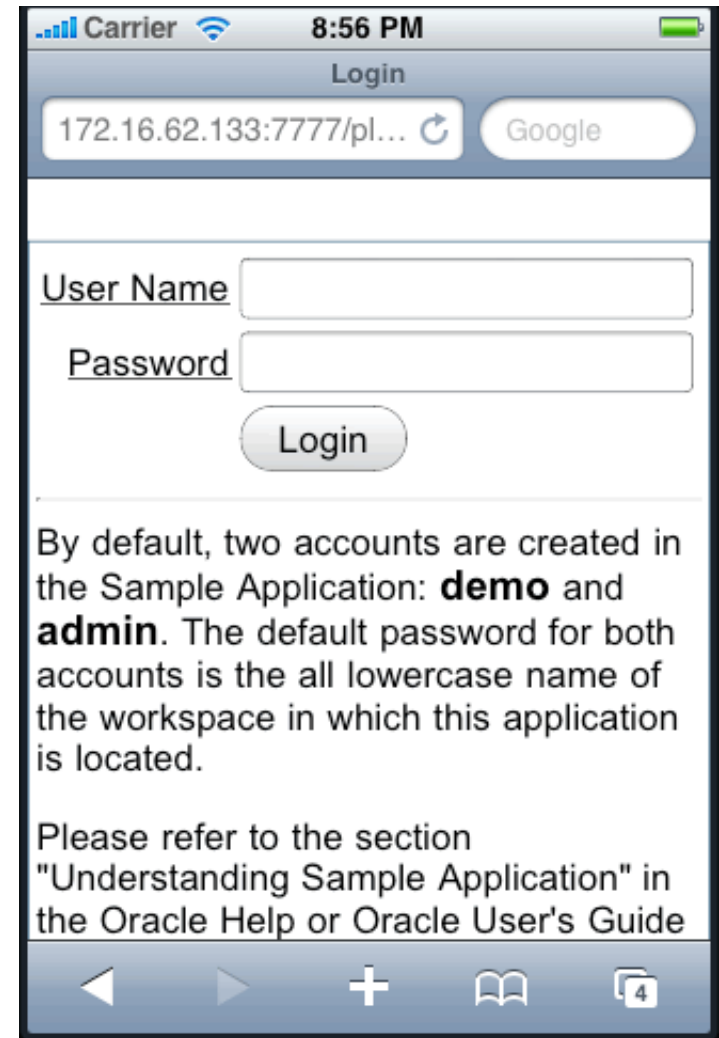
- Style sections with CSS

Add to Page header

```
<style>
.t12Login {
width:375px; -> 310px;
}
.t12OptionalLabelWithHelp {
font-size:12px; ->20px;
}
</style>
```

Or

```
html {-webkit-text-size-adjust:200%}
```



# Working with Apex Pages

- Style sections with CSS

Alternatively, set style sheets on the same app that will render based on device/browser

For the iPhone

```
<link media="only screen and (max-device-width: 480px)" href="small-device.css" type="text/css" rel="stylesheet">
```

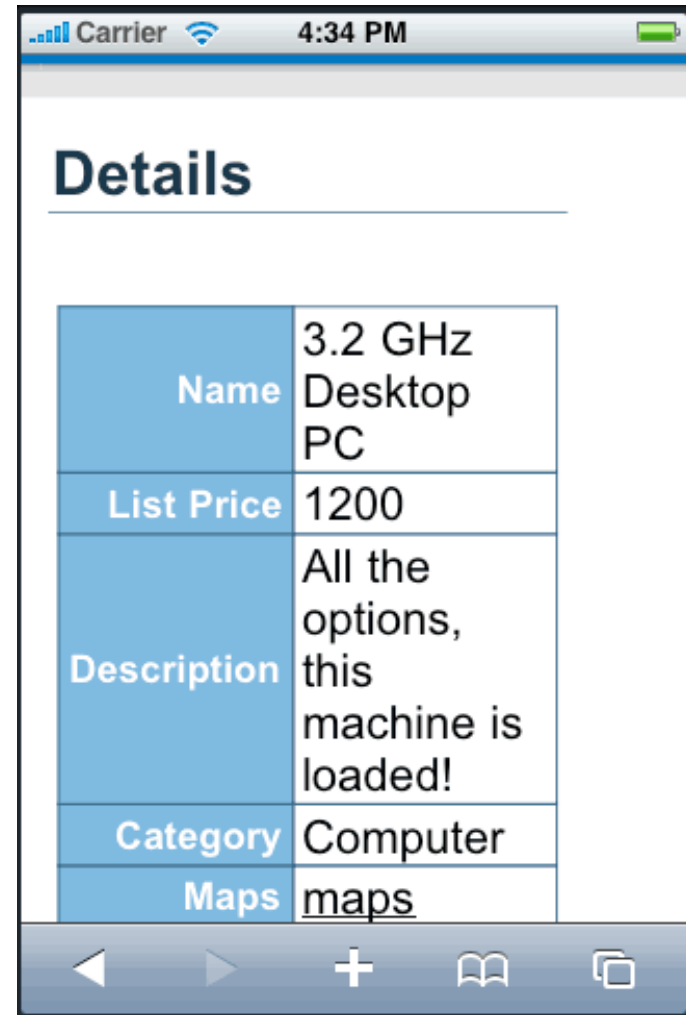
For a regular browser

```
<link media="screen and (min-device-width: 481px)" href="not-small-device.css" type="text/css" rel="stylesheet">
```

For a future Apex release (please)

```
Mozilla/5.0 (iPhone; U; CPU iPhone OS 3_0 like Mac OS X; en-us) AppleWebKit/525.18.1 (KHTML, like Gecko) Version/3.1.1 Mobile/XXXXX Safari/525.20
```

# Working with Apex Pages



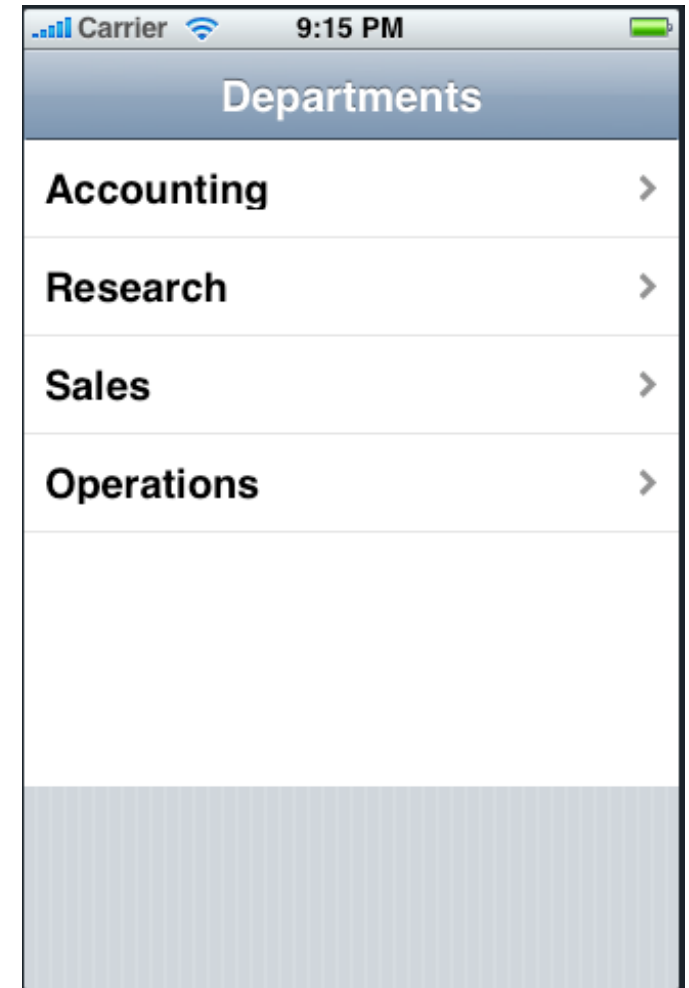
# Working with Apex Pages

- Demo
  - master-detail

# The Two Methods in practice

## The iPhone native App

- Look and feel of a native iPhone app
- Uses lists to display data
- Report driven UI



# Working with Apex Pages

- Modify an Apple Sample App
  - Drilldown sample app
  - Allows you to drill into categories and see products in those categories
  - Based on static JS arrays



```
function selectProducts(product)
```

```
{
```

```
    if (product == "iPods"){
```

```
        return ["iPod classic", "iPod nano", "iPod shuffle", "iPod  
touch"];
```

```
    }
```

```
    else if (product == "Macs"){
```

```
        return ["MacBook", "MacBook Air", "MacBook Pro"];
```

```
    }
```

```
    else if (product == "Applications"){
```

```
        return ["Mac OS X", "iLife", ".Mac", "iWork", "QuickTime",  
"Aperture", "Final Cut Studio"];
```

```
    }
```

```
    else if (product == "Servers"){
```

```
        return ["Xserve", "Xsan", "Mac OS X Server"];
```

```
    }
```

```
    else {
```

```
        return null;
```

```
    }
```

```
}
```

- 
- Demo
    - Drilldown sample app

# Working with Apex Pages

- Modify an Apple Sample App
  - Change from static arrays to dynamic
  - Use APEX on-Demand processes to query data from emp/dept tables
  - Return results into a array based on user selection (dept/emp/emp details)
  - Modify an APEX template to correctly display our results

# Download and start with the sample app

- [developer.apple.com](http://developer.apple.com)
  - Get a free account
- [developer.apple.com/safari/mobile.php](http://developer.apple.com/safari/mobile.php)
  - Reference guides
  - Sample Code
  - Tutorials
- Get the Drilldown app



# Working in the main.js file

- Replace the static arrays with onDemand Apex calls
  - First onDemand Process will start by filling the list with the departments

```
var macs = new Array; /*Create initial array */
```

```
function setMacs(){  
    var get = new htmldb_Get( null,12,'APPLICATION_PROCESS=getDept',  
0);  
    gReturn = get.get();  
    get = null;  
    var macs1 = gReturn.split(",")  
    return macs1;  
}
```

# Working in the main.js file

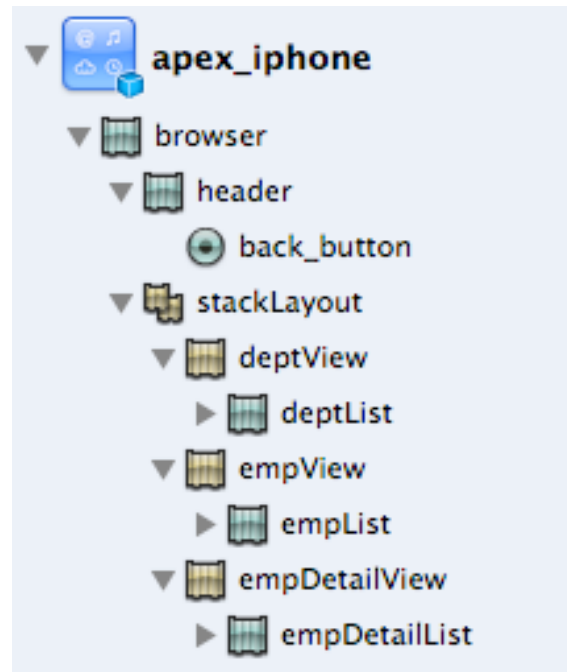
- Second onDemand Process will populate emps based on department selected

```
function selectDept(dept)
{
    var get = new htmldb_Get( null,12,'APPLICATION_PROCESS=
getEname',0);
    get.addParam('x01',dept);
    gReturn = get.get();
    get = null;
    var macs2 = gReturn.split(",")
    return macs2;
}
```

# Working in the main.js file

## The stacks

- The main.js file creates the list stacks with script vars, then put the values in the correct locations
- We need to change the names and add 1 more stack



# Create an Apex app

- Create an apex app with 1 page
  - Authentication is optional
  - Any theme will do
- Upload the needed files from the sample to the apex server (js, css, images)
  - Located in the root directory of the drilldown zip (css, main.js)
  - Also in the Parts folder (js files)
  - File system or apex repository

# Create the iPhone template

- Create a page template
- Transfer the index.html code to the apex template
- Make the changes to reflect the new values we used for the new stack layout. (emp,dept)

```
<div id="stackLayout">
  <div id="deptView">
    <ul id="deptList">
      <li id="deptRowTemplate" class="deptRowTemplate_template">
        <div class="deptLabel_template" id="deptLabel"></div>
        <div id="rowArrow" class="rowArrow_template"></div>
      </li>
    </ul>
  </div>
```

# On Page 1- Footer

Add to the page properties footer text area

```
<script type="text/javascript" src="/c/js/main.js" charset="utf-8"></script>
```

Footer Text

```
<script type="text/javascript" src="/c/js/main.js" charset="utf-8"></script>
```



# AJAX and Javascript...a closer look

```
function selectDept(dept)
{
    var get = new htmldb_Get( null,12,'APPLICATION_PROCESS=
getEname',0);
    get.addParam('x01',dept);
    gReturn = get.get();
    get = null;
    var macs2 = gReturn.split(",")
    return macs2;
}
```

# Create the onDemand processes

## getEame

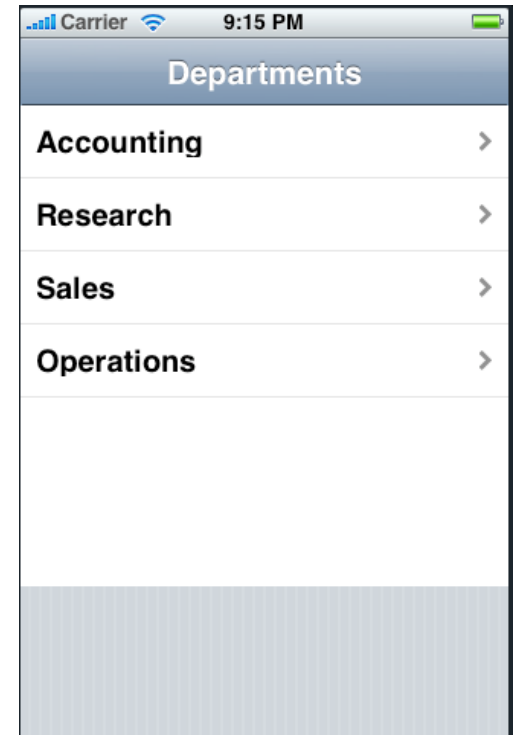
```
declare  
l_dept varchar2(200);  
begin
```

```
l_dept := wwv_flow.g_x01;
```

```
for x in (select rownum, ename from emp e, dept d  
where d.dname = upper(l_dept)  
and e.deptno = d.deptno)  
loop  
if x.rownum = 1 then  
htp.prn(initcap(x.ename));  
else  
htp.prn(', '||initcap(x.ename));  
end if;  
end loop;  
end;
```

# Testing your app

- On Windows
  - Use safari
  - If built on a public server, use your iPhone
- On Mac
  - Use safari
  - If built on a public server, use your iPhone
  - Use the iPhone SDK/Simulator



# To sum it up

- Apex on the iPhone is possible
  - Leverage existing apps/customers
  - iPhone Enable your offerings/apps
- Can develop on windows or mac
- Two methods
  - Minimized web pages (ie. Amazon.com)
  - Native iPhone apps look and feel
- Limited only by your imagination
  - Download samples
  - Google



# Resources

- <http://www.apple.com/webapps/whatarewebapps.html>
- <http://developer.apple.com/safari/mobile.php>
- <http://developer.apple.com/safari/library/navigation/index.html>

**All files hosted at <http://sumnertechnologies.com/iphone>**



Q/A