SOFTWARE DEVELOPMENT SERVICES

WEB APPLICATION PORTAL (WAP) TRAINING

Intuit 2007

I've included this training in my portfolio because it was very technical and I worked with a SME to develop it. It demonstrates my ability to take a complex subject and work with others to turn it into usable material.

Starting around slide 27 you will start to see diagrams of the application to further demonstrate how complicated the material is. Because I come from a developer background, even though I may not know the technology, I am able to document it anyway.

INTRODUCTION

Introduce Your Teammate:

- What is their experience at Intuit with portal work?
- What team are they on and what are they building?
- What would they like to get out of this training?
- What are their interests or hobbies?

COURSE STRUCTURE

Module 1 - Overview of Web Architecture Applications

Module 2 – WAP Methodology

Module 3 – Programming with WAP

MODULE 1 – INTRODUCTION TO WAP

Learning Objectives:

- Introducing WAP
- Understanding WAP as a web based/component based application
- Defining the portal/portlet components and the WAP architecture
- Identifying WAP system requirements

MODULE 1 – INTRODUCTION TO WAP

What is WAP?

- A component model and framework for building customer-facing web applications at Intuit
- A server image with required runtime software
- Enabling libraries for application creation
- Shared features for re-use in offerings
- Web design tool offerings that automate and improve the speed of the development process

MODULE 1 – INTRODUCTION TO WAP

The goals of WAP are to provide:

- Web products and services that wow and delight our customers
- Web development tools and offerings that are tailored to customers needs
- Long term revenue and growth from these offerings
- Lots of web offerings that address specific market segments

MODULE 1 – INTRODUCTION TO WAP

Our Web Challenges:

- Web apps are costly to build, host, and maintain lots of offerings all built from the ground up on miscellaneous technology and architecture
- As offerings get more sophisticated they become brittle and difficult to change
- Sharing components across offerings is difficult because they are all unique
- Changes start to create more and more bugs as offerings become mature
- Changes take longer and longer because the internal coupling of offerings continue to get more complicated
- Because each offering is unique it takes a long time to add resources that can work with offerings
- Old models of web development don't address these challenges, so doing more of the same doesn't address the problem but continues to feed it

MODULE 1 – INTRODUCTION TO WAP

SPC and WAP Terminology:

- BHP – Basic Hosting Platform for hosted services
- SPC-F – Software Programing Client Foundations support for language portable engines
- MAP – Microsoft Application Platform for Windows desktop applications
- WAP – Web Application Platform for customer facing web applications (built on BHP, where BHP is the container, and WAP is the stuff inside)

MODULE 1 – INTRODUCTION TO WAP

WAP Solutions:

- Use standards based portable solutions to let us leverage commercial products and open source
- Are innovative by componentizing our offerings using the Portal/Portlet model to allow rapid composition of features
- Encapsulate functions into components so the scope of changes and testing can be contained
- Deliver shared features based on this component model to raise the entry level of all offerings

MODULE 1 – INTRODUCTION TO WAP

The State of WAP:

- WAP 1.0 was released in December '06
- WAP 1.1 is currently under development targeting a spring '07 release
- Early iterations of WAP 1.0 are being used in production by Zipingo
- Stable iteration builds can be used for betas with customers
- Projects such as CTG's Online Tax Engine, IMS Merchant Portal, and SB Connect are actively developing on WAP now for release in the near future
- A new hosting service in IIT called One-Stop-Shop-For-Hosting is being built around hosting WAP offerings and providing pre-production and production shared hosting in a virtualized environment

MODULE 1 – INTRODUCTION TO WAP

Project Development Gains:

- WAP helps make global changes to the navigation and look and feel without touching application code
- Can rapidly create offering variants without code changes
- Can build up a library of company level (WAP) or product line (SB Connect) re-useable features and start new offerings with more and more capability that is already built and tested
- Can work in parallel without conflicts on components and combine them at deployment time
- Can work easily in multiple sites and combine efforts in the finished offering

MODULE 1 – INTRODUCTION TO WAP

Project Development Gains:

- Can iterate on any component, a portlet, a theme, the navigation, the layout without having a dependency on the other parts
- Can offer global user features like customizable properties, layout, in an efficient and decoupled manner
- WAP helps make global changes to navigation and look and feel without touching application code
- Can rapidly create offering variants without code changes
- Can work in parallel without conflicts on components and combine them at deployment time

MODULE 1 – INTRODUCTION TO WAP

Project Development Gains:

- Can work easily in multiple sites and combine efforts in the finished offering
- Can iterate on any component, a portlet, a theme, the navigation, the layout without having a dependency on the other parts
- Can offer global user features like customizable properties, layout, in an efficient and decoupled manner

MODULE 1 – INTRODUCTION TO WAP

WAP Components:

- Are designed as a web-based applications
- Supply applets that are embedded in Java based HTML code
- Create applications (or functions) that are embedded and customized to provide access to functions.

MODULE 1 – INTRODUCTION TO WAP

WAP Components are http- and Java-Based:

No new protocols are introduced, portlets and portals just talk standard http. When you use the JSF bridge, the portlet only sees the standard JSF lifecycle and behaves like a JSF application.

The Portal:

- Acts as a common dispatcher for all requests
- Applies a pipeline to each request to process it and generate the right result
- Invokes portlets through the Portlet interface to obtain rendered markup and to invoke actions

MODULE 1 – INTRODUCTION TO WAP

WAP is a Java-based application that uses:

- HTTP request processing
- Simple request response; delivers headers then content
- URL references a resource
- CGI – Common Gateway Interface maps a process to URL parameters and streams are passed to process; process emits response; headers then content
- JBoss Tomcat – web container handles dispatching requests to processes (threads) and then hands off the correct handler

MODULE 1 – INTRODUCTION TO WAP

WAP is a Java-Based application that uses:

- Servlets – request processing application, i.e. CGI app in Java
- .JSP – a markup language mixing markup and java code compiles into Servlet
- Portal – JBoss Portal, servlet-based application that generates a site based on metadata, a style theme, and a set of page layout templates
- Portlet – like a servlet or .jsp. It only renders a fragment for composition in a portal, defined by the Portlet spec JSR-168

MODULE 1 – INTRODUCTION TO WAP

How to Use WAP:

- Use the new WAP Template in your project
- It contains a lot of the common tasks like creating offerings and portlets into an ANT-based template so that the project offerings in the project are automated
- WAP packages a standard build script
- WAP generates the starting configuration for a skeleton project

Then you:

- Generate your offering project and check it in
- Build it and deploy it
- Customize it by adding pages, portlets, themes, layouts etc…

MODULE 1 – INTRODUCTION TO WAP

Can Portlets Talk to Each Other Using WAP?

- If they are in the same portal application they can share state data in the PortletSession 'application' scope on the server
- Portlets can read parameters directly from the URL, which is good for creating screens that can be bookmarked
- Portlets can set and read session attributes in the HTTP session of the application server
- On the client, WAP has logical-named JavaScript events so that portlets can raise and handle events without prior knowledge of each other, allowing for dynamic interaction without communicating back to the server

MODULE 1 – INTRODUCTION TO WAP

What is a Portal?

Think of the portal as the windowing system for the web:

- The portal framework solves a lot of global problems that we don't want to embed in code
  - Access control, personalization, skinning the site, site wide navigation, per user/per component settings, multiple device rendering, deployment at the feature level, creation of offerings by combining components at deployment time, customizable layout…and more…
- If we built all of the things above, we'd end up building the portal framework, instead, with WAP, we just use it
- We can leverage open source and commercial portlets, portals, and tools to reduce the amount of new coding and testing we have to do

MODULE 1 – INTRODUCTION TO WAP

Portlets are:

- Mini-web applications that adheres to the JSR-168 Portlet specification
- They render a fragment of markup and handle actions addressed to it
- They are a full web application with business logic, persistence, and separate deployment in a .WAR file
- They can be built using common web technologies like JSF so that engineers need to know little about "portlet-ness"
- They contain a whole offering feature, including things like screen flows
- They derive styles and graphics from a portal wide theme to allow skinning for use in different offerings
- Each portlet class can be used in multiple logical portlet instances that each can have different initialization parameters enhancing reuse and requiring only one deployment for multiple uses

MODULE 1 – INTRODUCTION TO WAP

What a Portlet Is Not:

- A single UI control with no business logic meant to be embedded on a form: that's a JSF control
- The whole site including pages and navigation, this would degenerate a portlet into being the whole app, defeating most of the benefit of componentization
- A particular rendering on the client; portlets can render any markup and can in fact, support multiple renderings, html, mobile xhtml, wml, flex, openlaszlo etc…
- A servlet, even though it's packaged as a web application, it isn't a servlet; it is always invoked by the portal framework

MODULE 1 – INTRODUCTION TO WAP

Navigation Portlet:

- WAP provides a portlet class that reads the portal metadata and provides an interface to it
- The navigation portlet delegates to a .jsp for rendering so you can create whatever kind of navigation rendering you need
- We provide tag libraries for rendering tabs, menus, and outline (or tree) style navigation
- We provide pre-made views for the navigation portlet that render these kinds of navigation
- You can use them directly and customize the look by overriding the CSS styles

MODULE 1 – INTRODUCTION TO WAP

Navigation Portlet:

- Our navigation tags generate UI components that have a JavaScript event model so you can manipulate them either by sending named events to them, or by listening for their events and providing your own handlers
- The reason you need this is that the navigation in your application is dynamic, based on personalization or security, items may be added or removed for a given user. The model provided by the navigation portlet has been filtered for this
- There is some additional configuration for navigation portlets to allow them to access the portal metadata, we provide examples
- Other than that, they are normal portlets

MODULE 1 – INTRODUCTION TO WAP

Themes and Layouts:

- The theme is the look of your offering—it should be a style guide that identifies colors, fonts, graphics, and backgrounds
- The layout is the set of different page compositions that you use so most sites have a few different component layouts
- The two together are called the skin of an offering
- Along with the portal metadata, these items might just be the whole offering!
- In a JBoss portal, the style sheet for the offering is called a theme—layouts are .jsp's that are used to layout portlets in a page
- Themes and layouts are packaged in a .war file

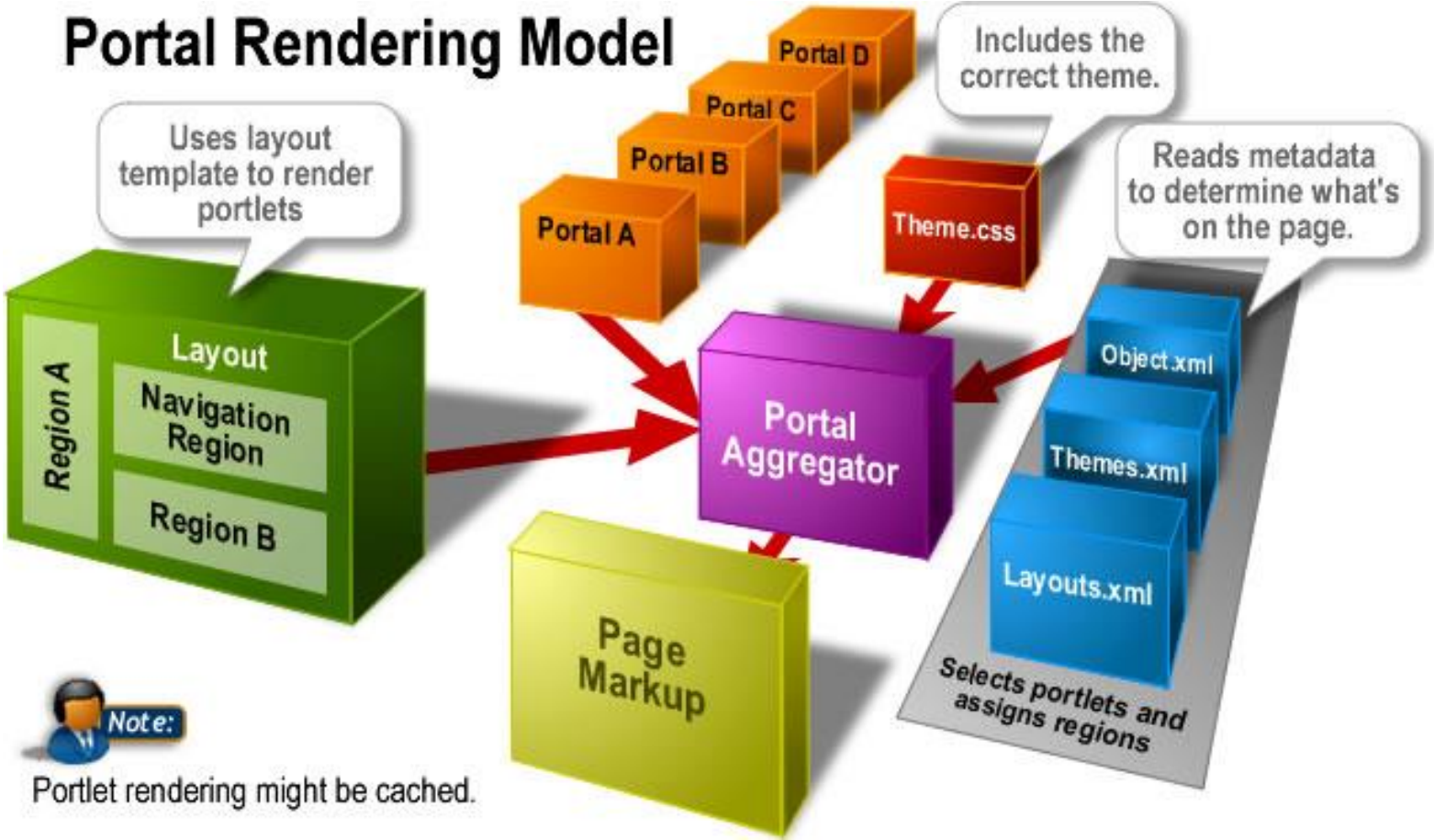
MODULE 1 – INTRODUCTION TO WAP

Themes and Layouts:

- The portal metadata selects a default theme and layout for the portal—each page can override the theme and layout
- Keep the skin and the global layout of pages separate from the application code
- A WAP Project Template generates a default theme that you can customize
- Named regions define the place where portlets will be substituted into a layout—multiple portlets can be arranged in a region
- There are also rendering classes for regions, the frames around portlets, and the div that portlet renders into that can be overridden for advanced effects

MODULE 1 – INTRODUCTION TO WAP

WAP Architecture:



MODULE 1 – INTRODUCTION TO WAP

Themes and Layouts:

- Portlet.xml – define portlet instances and give them names
- *-object.xml – portal metadata… define portal pages, what portlets they contain, what layout and theme, what regions of the layout should a portlet be assigned
- Jboss-portal-themes.xml – define available themes
- Jboss-portal-layouts.xml – define available layouts and the regions they support
- Web.xml for your portlet .war – sometimes there are standard web.xml configurations required, but mostly use the boilerplate the WAP Template Project provides
- Build.xml in one of you sub directories to add ANT build dependencies (we plan to automate this in future release)

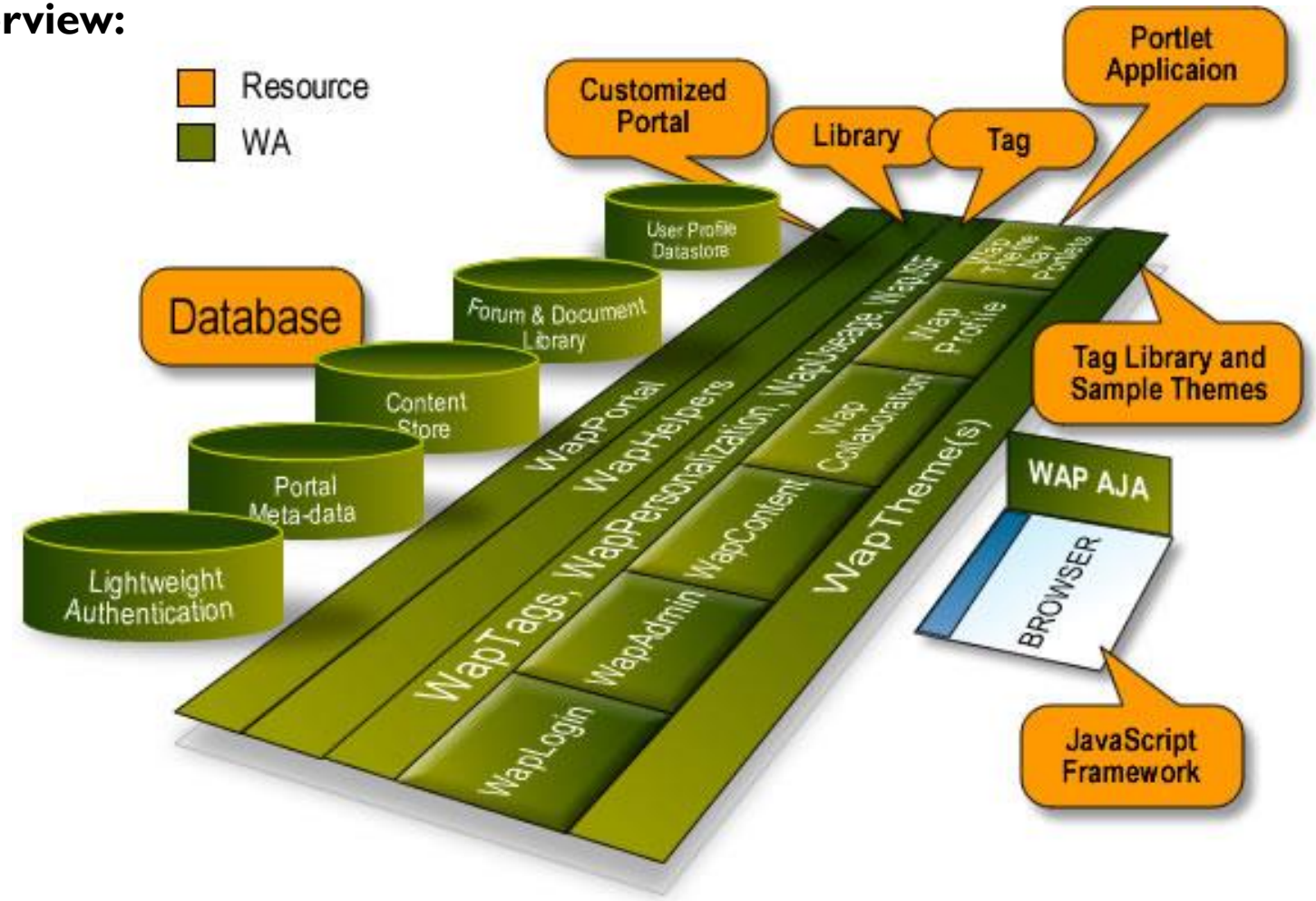
MODULE 1 – INTRODUCTION TO WAP

Themes and Layouts:

- *-spring-config.xml – configure services provided as spring components (used infrequently)
- *.hbm – hibernate mapping file will map your persistent objects and collections onto your schema and enable hibernate to manage their persistence
- These are in the order of most frequent changes and in general they are quite stable

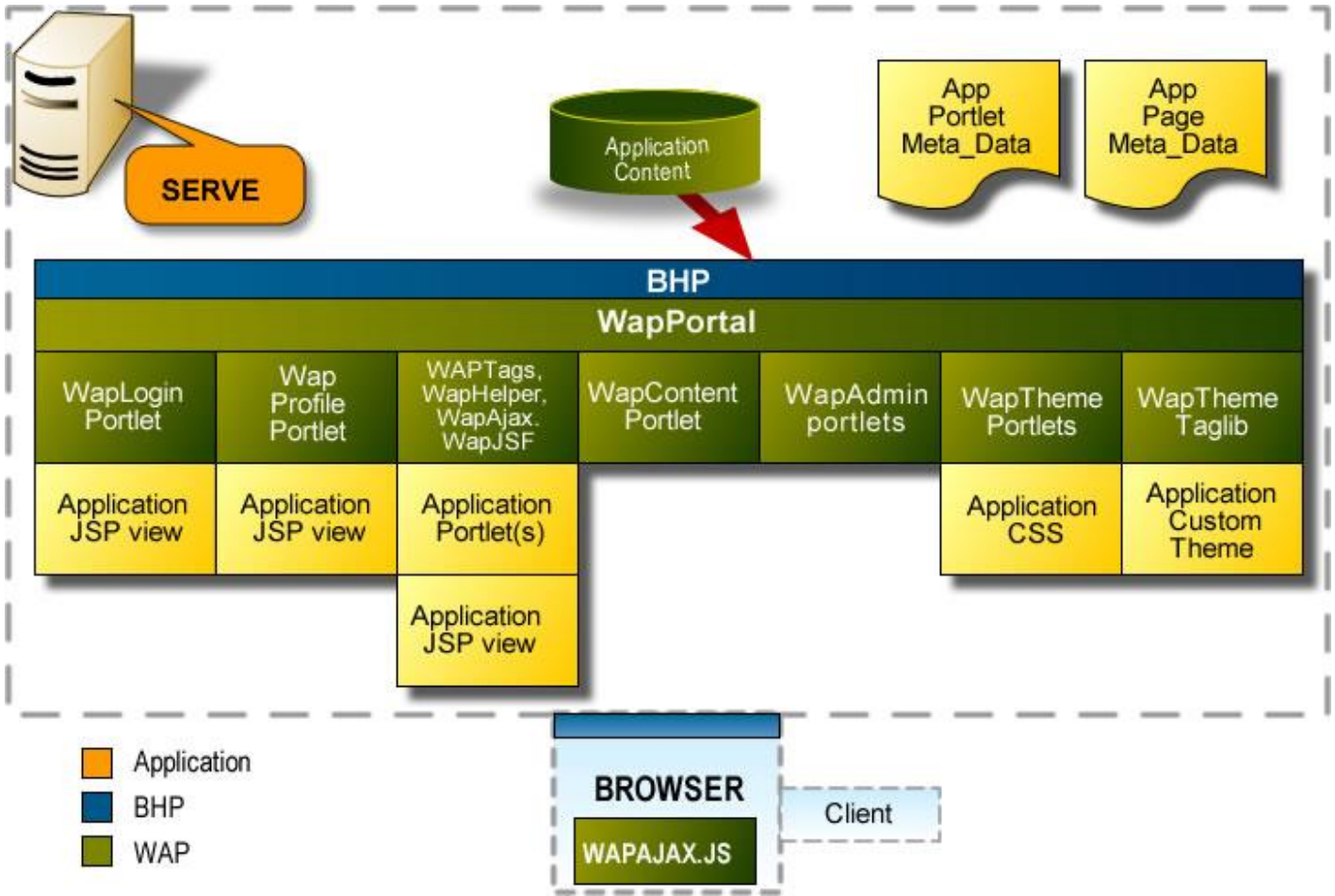
MODULE 1 – INTRODUCTION TO WAP

Runtime Overview:



MODULE 1 – INTRODUCTION TO WAP

Conical Web Site Application:



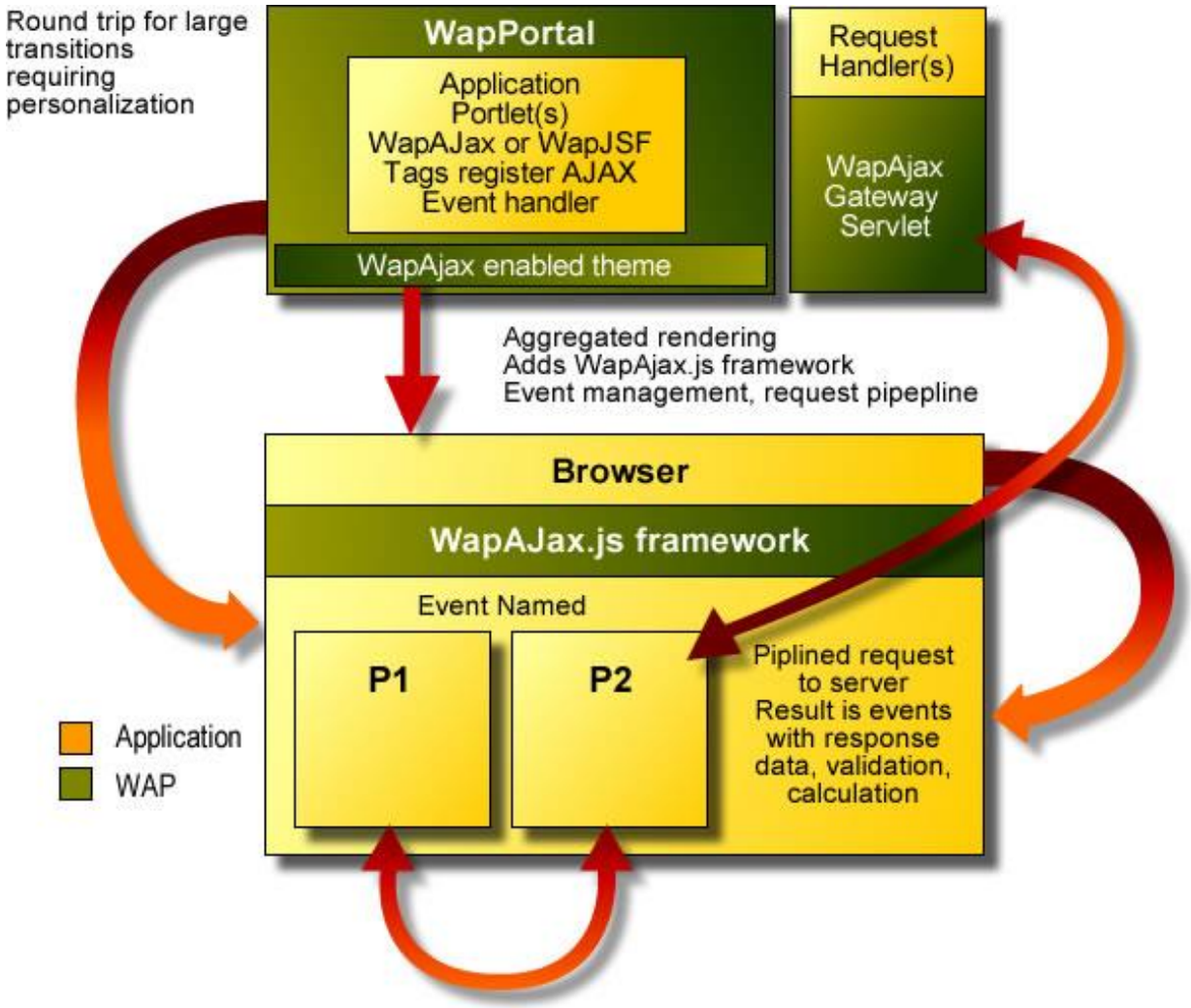
MODULE 1 – INTRODUCTION TO WAP

WAP and AJAX:

- Asynchronous JavaScript and XML is the latest name for using the XMLHttpRequest object in the browser to communicate in the background with the server via HTTP
- WAP packages and provides tag support for AJAX, applying the same event model as our interportlet events
- You can invoke any POJO or Spring object on the server using a JavaScript proxy
- WAP provides batching of requests into priorities so that many small requests can be sent to the server as one big request
- AJAX-enabled JSF controls are supported in WAP to provide partial rendering to speed up client UI's especially with large data sets

MODULE 1 – INTRODUCTION TO WAP

WAP Ajax Integration:



MODULE 1 – INTRODUCTION TO WAP

JavaServer Faces (JSF):

- Asynchronous JavaScript and XML
- UI component based model-view-controller web framework
- UI's are built out of controls referenced with tags in a .jsp view
- Controls are bound to backing objects backing beans execute actions and persist states
- Uses JSF navigation rules
- JSF restores the state of the controls in a view until a new view is made current by a navigation event
- Both standard JSF controls, open source (tomahawk, myfaces), and commercial Infragistics and as needed WAP controls

MODULE 1 – INTRODUCTION TO WAP

JavaServer Faces (JSF):

- Working with Infragistics to create AJAX-enabled advanced UI controls
- Tools support JSF either by WYSIWYG composition or non visual tag assist
- JSF fits well with the portal rendering model and executes seamlessly in a JSF bridge portlet

MODULE 1 – INTRODUCTION TO WAP

Hibernate:

- Hibernate is an ORM engine-object relational mapping engine
- It stores and retrieves and queries graphs of objects using a relational database
- It provides caching, lazy loading, database independence, and reduces the amount of SQL you have to write
- Objects managed by Hibernate are just POJOs (plain old java objects) and are easy to use and to make an existing object persistent

MODULE 1 – INTRODUCTION TO WAP

Hibernate:

- There are many ways to use Hibernate, one popular way is to annotate objects and generate the hibernate configuration, this makes most of hibernates function transparent
- Other options are to generate the hibernate mapping and data objects from an existing schema, or to generate the data objects and the schema from the hibernate mapping file

MODULE 1 – INTRODUCTION TO WAP

Security:

- We support login and single sign-on with Intuit's AUTH system
- Implemented using container managed security in JBoss via the JAAS standard
- Declare security assertions in web.xml for web applications based on users and roles
- Declare page, portlet security assertions in *-object.xml based on users and roles
- Declare portlet logical security role mappings in portlet.xml, create portlets that can run in multiple applications by mapping application roles to portlet roles

MODULE 1 – INTRODUCTION TO WAP

Security:

- In code use JAAS to check the UserPrincipal to see if has component roles
- For access to Intuit-specific user information, use the WAPPrincipal interface on the standard UserPrincipal to query and update these properties
- WAP's security is outside application code as much as possible so that it stays correct, can be easily reviewed and changed, and so that policy changes can be less risky and involve little or no code changes
- WAP contains a shared feature Portlet for login, account creation, account maintenance, password recovery, and userid recovery

MODULE 1 – INTRODUCTION TO WAP

Security:

Linux or Solaris	36 Gig Internal Storage
Java 1.5x	NFS Storage (optional)
JBoss Portal 2.2 (and requisite app server)	Oracle 9i on Solaris shared or dedicated (optional - any Hibernate supported database)
Apache Web Server	JSR-168 Portlet API
Dual CPU App and Web Servers	JSF user interface framework
8Gig memory on App Servers, 4Gig on Web Servers	Spring ICC (optional)

MODULE 2 – WAP METHODOLOGY

Learning Objectives:

- Define component based offerings
- Analyze and review a design concept
- Decompose a design concept (project)
- Defining a WAP Portal (parts and pieces)

MODULE 2 – WAP METHODOLOGY

WAP Development Process:

When developing a WAP project:

- Design a site concept
- Decompose into portlets and pages and define navigation and personalization
  - Use shared portlets wherever possible and customize look and feel with .css or custom .jsp view
  - Mock up site structure for pages, substitute content for portlets not done (usability test)

MODULE 2 – WAP METHODOLOGY

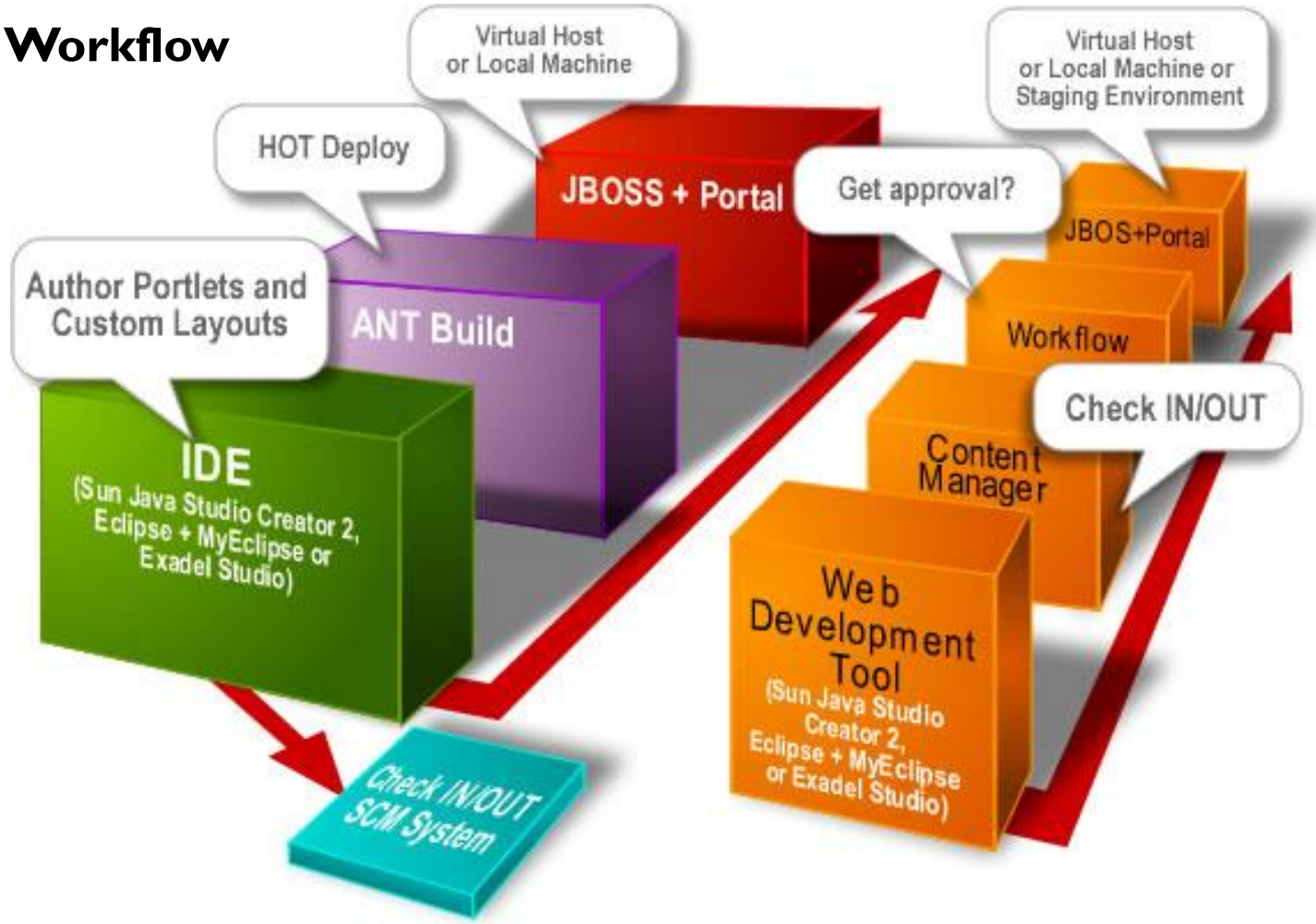
WAP Development Process:

In parallel:

- Create custom theme
- Create portlets using JSF GUI design tools
- Create static site content in content management system publish using content portlets
- Replace mock portlets and usability test (iterate)

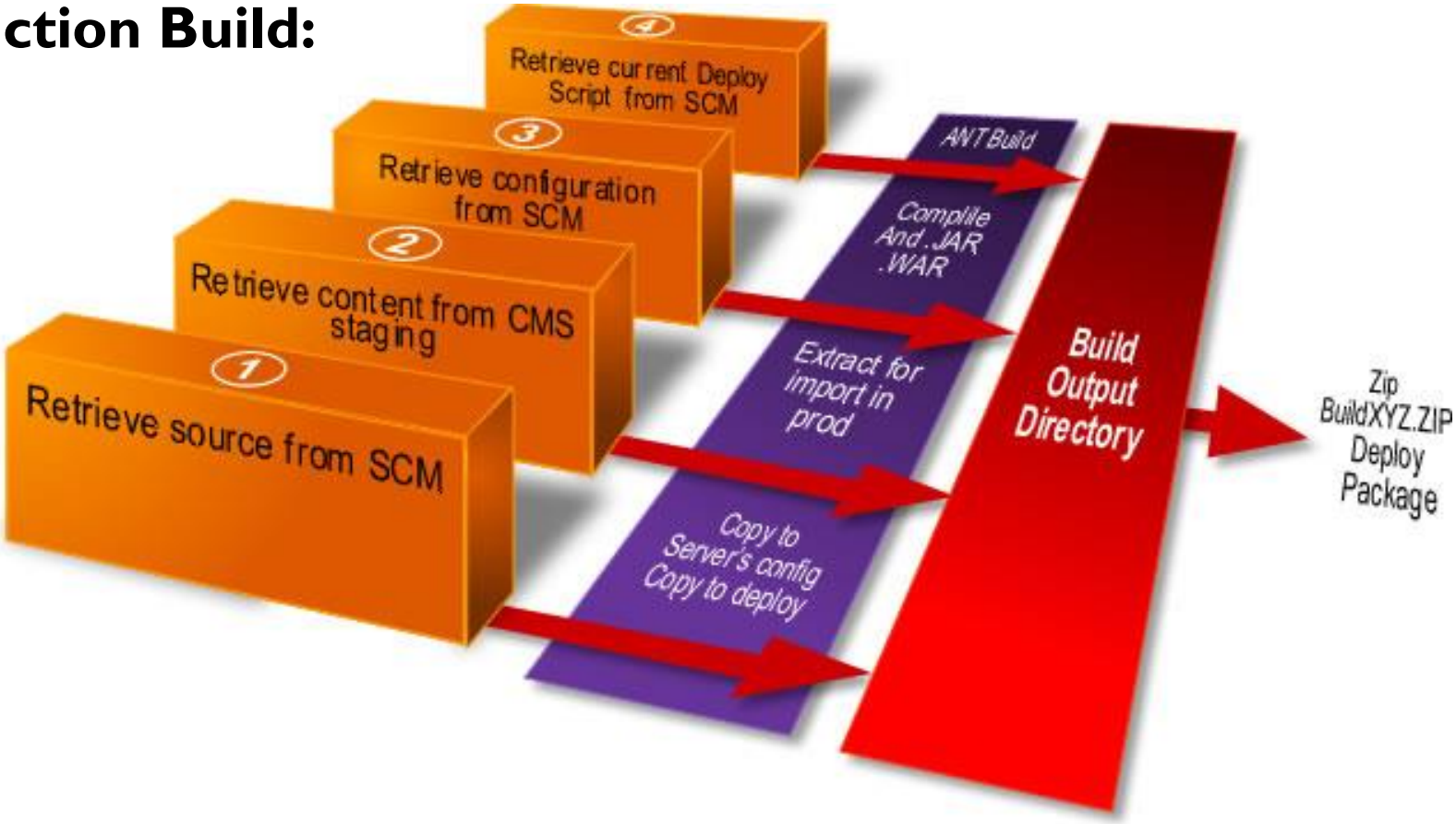
MODULE 2 – WAP METHODOLOGY

Development Workflow Example:



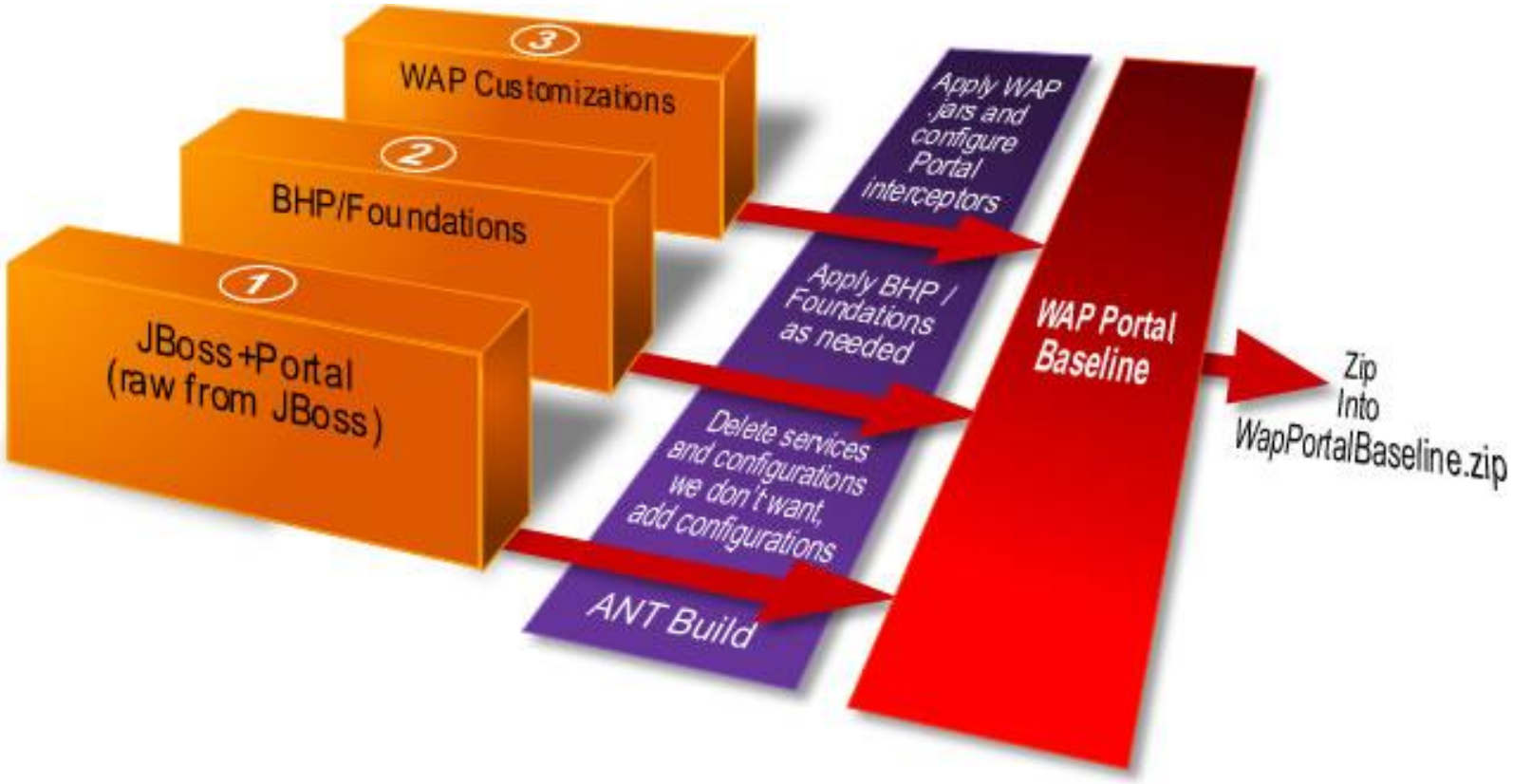
MODULE 2 – WAP METHODOLOGY

Production Build:



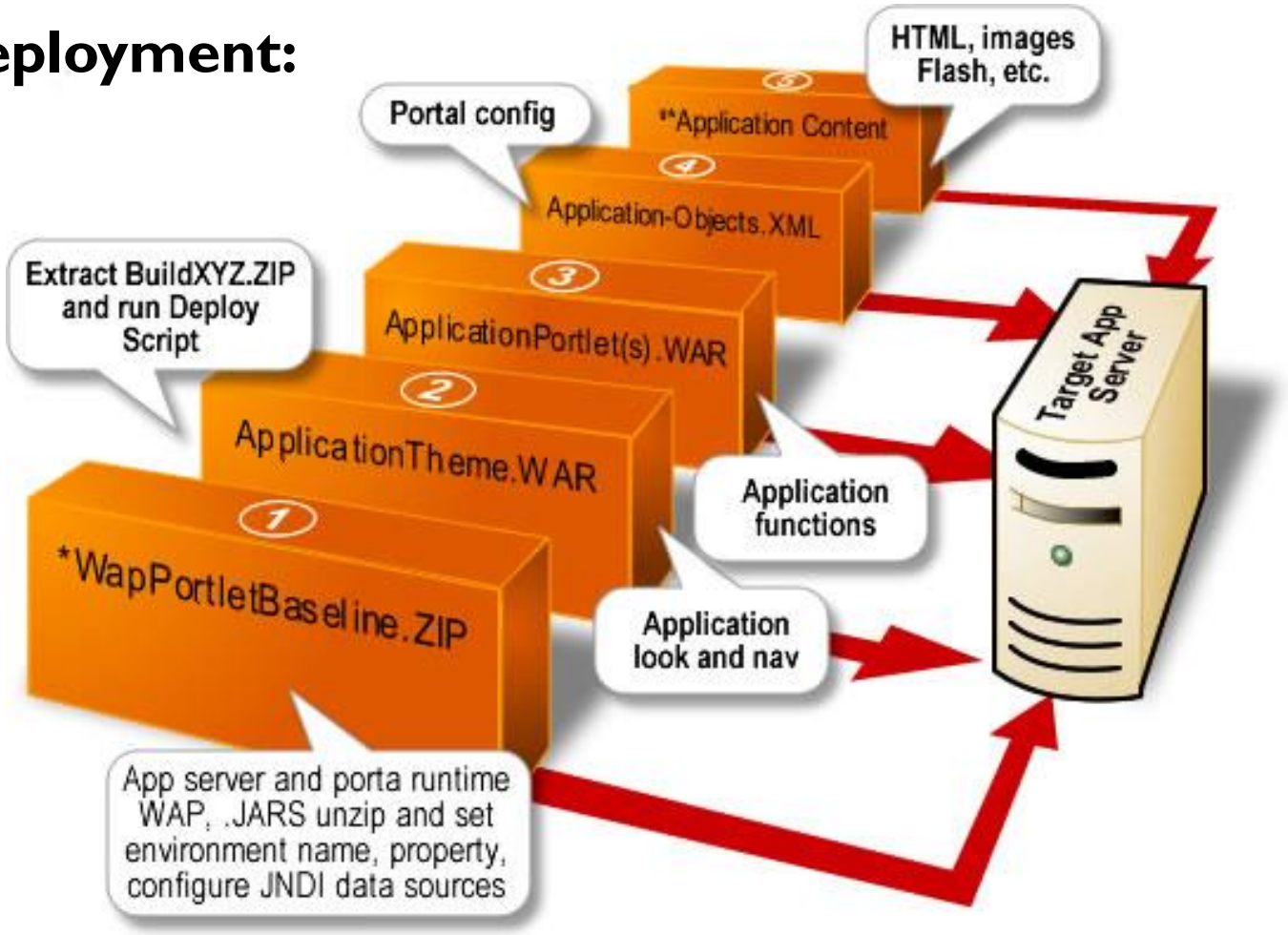
MODULE 2 – WAP METHODOLOGY

WAP Portal Baseline:



MODULE 2 – WAP METHODOLOGY

Production Deployment:



MODULE 2 – WAP METHODOLOGY

About Deployment:

- Baseline only needs to be done initially and at upgrades
- Content is pushed into production CMS from staging CMS or via bulk load from .zip
- Prior configuration is backed up for rapid rollback
- Incremental deployment encouraged:
  - ApplicationTheme – change look and feel or nav method
  - ApplicationPortlet – patch, upgrade product function
  - Application-Objects.xml – extend site structure
  - ApplicationContent – update content directly without engineering involvement
- Sample ANT script to push to deployment server from build output

MODULE 2 – WAP METHODOLOGY

WAP Offerings:

- Portals that help monitor applications
- Usage monitoring
- Collaboration
- Log in and authentication services
- Profiling
- Administration
- Navigation
- Managing content
- Credit card authorizations, PCI, etc.

MODULE 2 – WAP METHODOLOGY

WAP Capabilities:

- Portal Helpers
- Page Level Logical Events and Inter-Portlet Logical Events
- AJAX tags and Server objects
- Browser Capabilities
- Personalization expressions
- Account Interface and WAP Principal
- Profile Interface
- Navigation Tags and Portlets
- Login and Account Maintenance Portlet and Universal Login

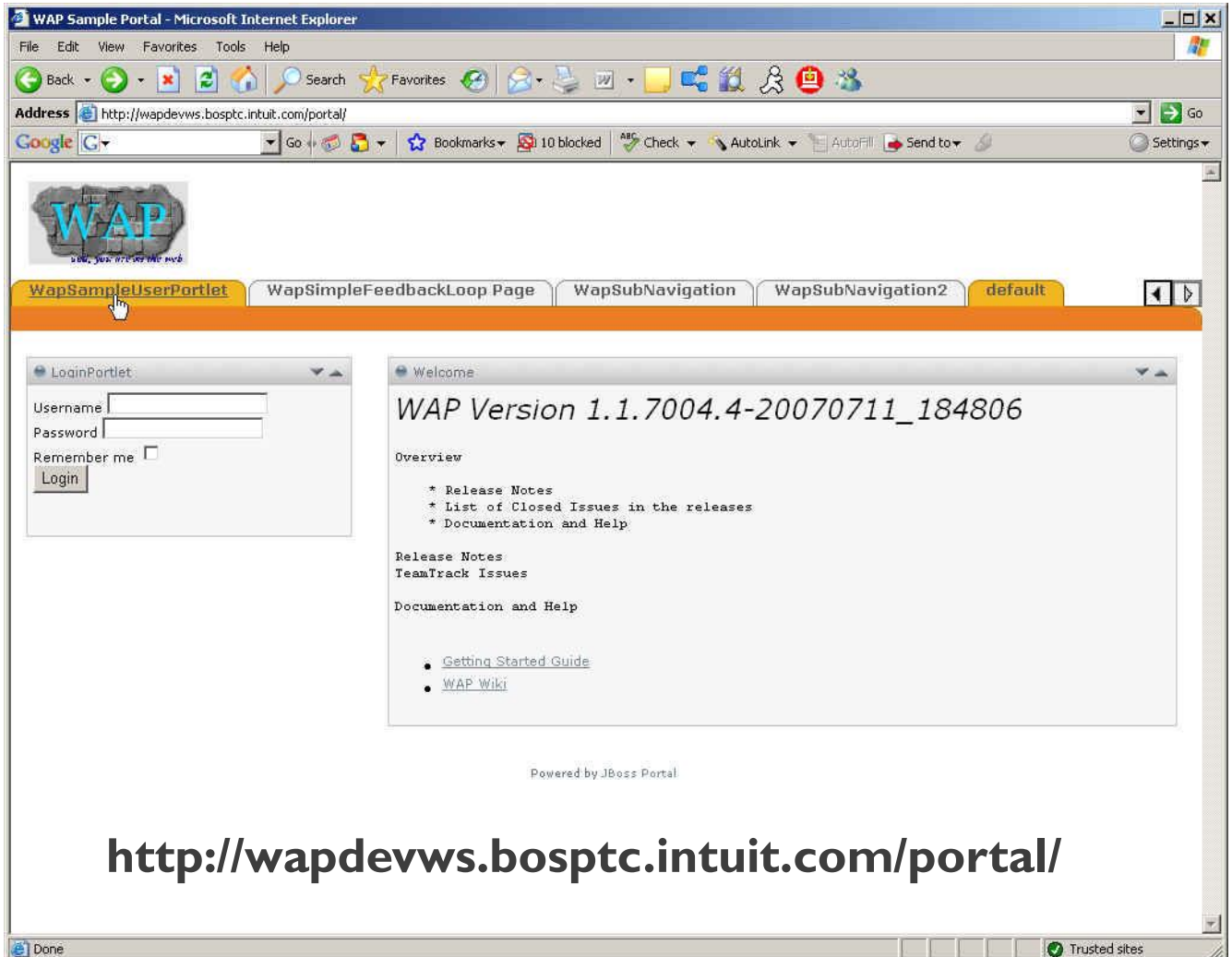
MODULE 2 – WAP METHODOLOGY

WAP Capabilities:

- AJAX Login Tags
- The IFrame portlet and service
- The Feedback tag, password feedback, custom feedback
- Namespace tag for making portlet scoped JavaScript instances
- Weblets
- Audit Logging

MODULE 2 – WAP METHODOLOGY

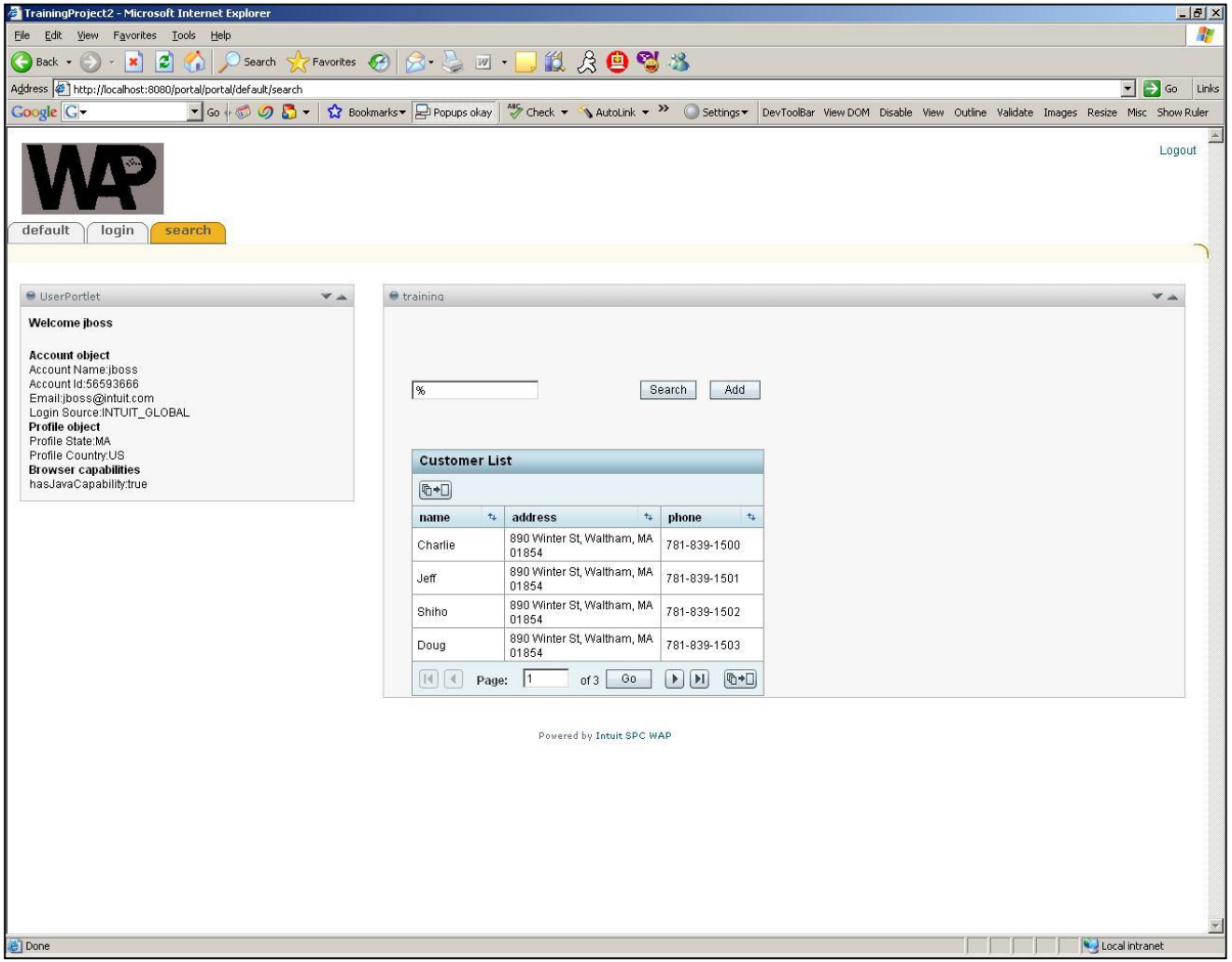
Example WAP Site:



http://wapdevws.bosptc.intuit.com/portal/

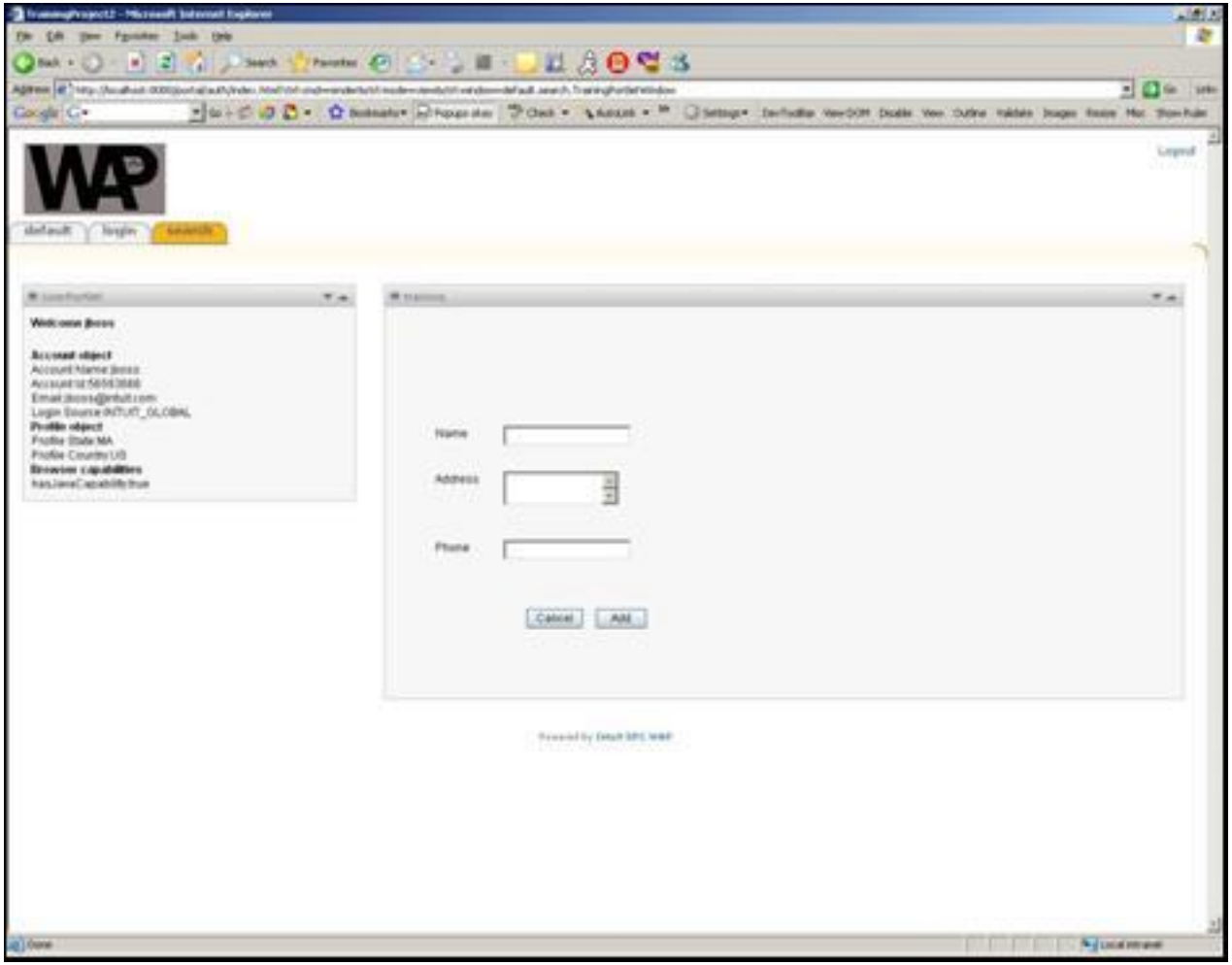
MODULE 2 – WAP METHODOLOGY

Data Search Example:



MODULE 2 – WAP METHODOLOGY

Data Entry Form Example:



MODULE 3 – PROGRAMMING WITH WAP

Learning Objectives for Hands-On Session:

- Create a WAP project framework
- Create a WAP portlet (XML meta data)
- Create a WAP template (Layout, look and feel)
- Create a new offering
- Deploy the offering
- Add WAP Portlets
- Change the default page name
- Update the default page
- Update and deploy the theme
- Create and deploy a new JSP Portlet
- Update dependencies