Java EE 5 Web Application

Asst. Prof. Dr. Kanda Runapongsa
(krunapon@kku.ac.th)
Department of Computer Engineering
Khon Kaen University

1

Agenda

- **Web Application, Components and Container**
- Web Application Life Cycle
- Web Modules
- Configuring Web Applications

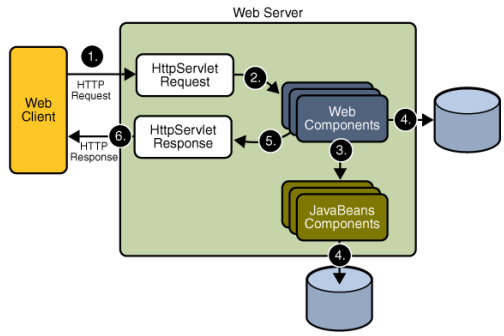
2

What is a Web Application?

- A Web application is a dynamic extension of a Web or application server
- There are two types of Web applications
  - Presentation-oriented: Generates interactive Web pages containing various types of markup language and dynamic content in response to requests
  - Service-oriented: A service-oriented Web application implements the endpoint of a Web service.
    - Presentation-oriented applications are often clients of service-oriented Web applications

3

Java Web Application Request Handling



4

Web Components

- Web components provide the dynamic extension capabilities for a Web server
- Web components are either Java servlets, JSP pages, or Web service endpoints
- Servlets are Java programming language classes that dynamically process requests and construct responses
- JSP pages are text-based documents that execute as servlets but allow a more natural approach to creating static content

5

Servlets

- Servlets are best suited for
  - Service-oriented applications (Web service endpoints are implemented as servlets)
  - The control functions of a presentation-oriented application, such as dispatching requests and handling non-textual data

6

JSP

- JSP pages are more appropriate for generating text-based markup
  - HTML
  - Scalable Vector Graphics (SVG),
  - Wireless Markup Language (WML)
  - XML

7

Web Container

- Web components are supported by the services of a runtime platform called a Web container
- A Web container provides services such as request dispatching, security, concurrency, and life-cycle management
- It also gives Web components access to APIs such as naming, transactions, and email

8

Web Application & Components

- Web Application is a deployable package
  - Web components
  - Static resource files such as images
  - Helper classes
  - Libraries
  - Deployment descriptor (web.xml file)
- Web Application can be represented as
  - A hierarchy of directories and files (unpacked form) or
  - *.WAR file reflecting the same hierarchy (packed form)

9

What is *.WAR File?

- Ready to deploy'able package over web container
- Similar to *.jar file
- Contain things to be deployed
  - Web components (servlets or JSP's)
  - Server-side utility classes
  - Static Web presentation content (HTML, image, etc)
  - Client-side classes (applets and utility classes)

10

Agenda

- Web Application, Components and Container
- Web Application Life Cycle
- Web Modules
- Configuring Web Applications

11

Web Application Life Cycle

1. Develop the web component code
2. Develop the web application deployment descriptor
3. Compile the web application components and helper classes referenced by the components
4. Optionally package the application into a deployable unit
5. Deploy the application into a web container
6. Access a URL that references the web application

12

Web Modules

- A web module is the smallest deployable and usable unit of web service
- A Java EE web module corresponds to a web application
- A web module can contain
  - Web components
  - Web resources
  - Server-side utility classes
  - Client-side classes

13

Document Root

- The top-level directory of a web module is the document root of the application
- The document root contains
  - JSP pages
  - Client-side classes and archives
  - Static web resources
  - /WEB-INF/ subdirectory

14

/WEB-INF/ Subdirectory

- The /WEB-INF/ subdirectory contains the following files and directories
  - web.xml: The web application deployment descriptor
  - Tag library descriptor files
  - classes: A directory that contains server-side classes
  - tags: A directory that contain tag files
  - lib: A directory that contains JAR archives of libraries

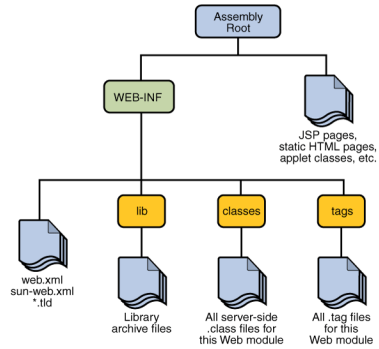
15

Web Modules Deployment

- A Web module can be deployed as an unpacked file structure or can be packaged in a JAR file known as a Web archive (WAR) file
- Because the contents and use of WAR files differ from those of JAR files, WAR file names use a .war extension
- The Web module just described is portable; you can deploy it to any Web container that conforms to the Java Servlet Specification

16

Web Module Structure



17

Packaging Web Modules

- A Web module must be packaged into a WAR in certain deployment scenarios and whenever you want to distribute the Web module
- You package a Web module into a WAR by
  - Executing the jar command
  - Using the ant utility

18

Packing Web Modules hello1

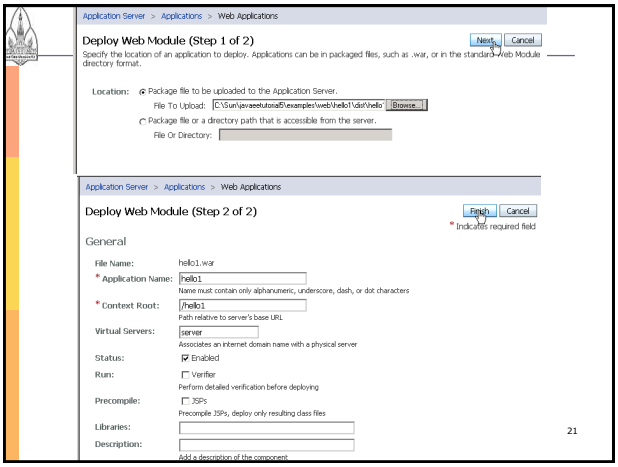
- In a terminal window, go to <INSTALL>/javaeetutorial5/examples/web/hello1/
- Run “asant” command
- This command will
  - Create the WAR file and copy it to the <INSTALL>/javaeetutorial5/examples/web/hello1/dist/ directory

19

Deploy a WAR File

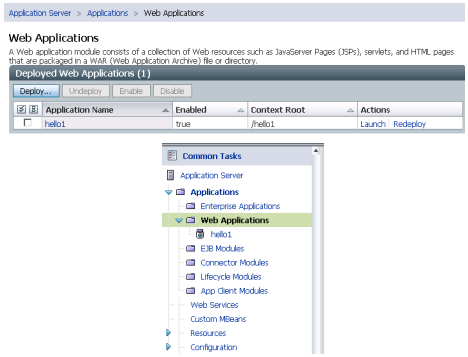
- You can deploy a WAR file to the Application server in a few ways
  - Copying the WAR into the <JavaEE_HOME>/domains/domain1/autodeploy/ directory
  - Using the Admin Console
  - By running asadmin or ant to deploy the WAR
    - asadmin deploy path-to-war-file

20



21

Deployed Web Applications



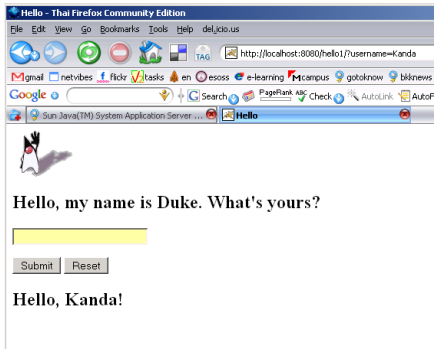
22

Testing Deployed Web Modules

- To test the application, follow these steps
  - Open a web browser
  - Enter the following URL in the web address box
    http://localhost:8080/hello1
  - Enter your name, and click Submit

23

Hello1 Web Application



24

Undeploying Web Modules

- You can undeploy web modules in three ways:
- Admin Console
  - Open the URL http://localhost:4848/asadmin in a browser.
  - Expand the Applications node.
  - Select Web Applications.
  - Click the checkbox next to the module you wish to undeploy.
  - Click the Undeploy button.
- asadmin
  - Execute

  asadmin undeploy context_root
- ant
  - In the directory where you built and packaged the WAR, execute

    ant undeploy

25

Listing Deployed Web Modules

- The Application Server provides three ways to view the deployed Web modules
  - deploytool
    - Select localhost:4848 from the Servers list
    - View the Deployed Objects list in the General tab
  - Admin Console
    - Open the URL http://localhost:4848/asadmin in a browser
    - Expand the nodes Applications → Web Applications
  - asadmin
    - Execute asadmin list-components

26

Agenda

- Web Application, Components and Container
- Web Application Life Cycle
- Web Modules
- Configuring Web Applications

27

Mapping URLs to Web Components

- When a request is received by the web container
  - It must determine which web component should handle the request.
- It does so by mapping the URL path contained in the request to a web application and a web component.
- A URL path contains the context root and an alias:

  http://host:port/context_root/alias

28

Setting the Component Alias

- The alias identifies the web component that should handle a request
- The alias path must
  - Start with a forward slash (/)
  - End with a string or a wildcard expression with an expression (for example *.jsp)

29

Example: hello2 Web Application

- In a terminal window, go to <INSTALL>/javaeetutorial5/examples/web/hello2/
- Run “asant”
  - This target will build the WAR file and copy it to the <INSTALL>/javaeetutorial5/examples/web/hello2/dist/ directory

30

Configuring hello2 Files

- To configure this example, the web.xml file must contain the following:
  - A display-name element set to hello2
  - Two servlet elements to add the GreetingServlet and ResponseServlet servlets to the WAR file
  - A servlet-mapping element to map GreetingServlet to the /greeting URL pattern
  - Another servlet-mapping element to map ResponseServlet to the /response URL pattern
- The sun-web.xml file needs
  - A context-root element set to /hello2

31

Configuring hello2 Security (1/3)

- Web.xml

```
<login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>file</realm-name>
</login-config>
```

- Sun-web.xml

```
<security-role-mapping>
  <role-name>helloUser</role-name>
  <group-name>user</group-name>
</security-role-mapping>
```

32

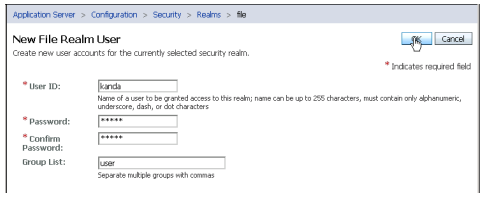
Configuring hello2 Security (2/3)

- Go to Sun Java System Application Server Platform Edition 9.0 Admin Console
- Then click at Configuration
- Then click at Security to expand the sub-menu
- Then click at Realm
- Then click at file
- Then click “Manager Users…”
  - Add new user with the group “user”

33

Configuring hello2 Security (3/3)

- Modify at Realms “file”
- Add user in the group list “user”



34

Run the Application

- To run the application,
  - Then deploy the web module which can be done by using Admin Console
  - Then open the URL http://localhost:8080/hello2/greeting in a browser.

35

Hello2 Web Application

36

Ant Build Tool

- Ant is a make tool that is portable across platforms
- Open source project under Apache
- Operates under the control of a build file, normally called build.xml
- build.xml is stored in the top-level directory of your source code directory

37

Ant Environment in Application Server

- In the Ant environment, build.xml files are analogous to Makefile. A build.xml file can define various targets that are used to compile and assemble a J2EE application
- A simple wrapper script named asant (.bat) is located in directory <J2EE_INSTALL_DIR>/bin

38

build.xml File

- Contains several targets for
  - Compiling the appliction
    - A temporary ./build directory is created beneath the root
    - ./build directory contains an exact image of the binary distribution for your Web application
  - Installing the application on a running server
  - Reloading the modified application onto the running server
  - Removing old copies of the application to regenerate their content

39

Targets in build.xml (1/2)

Target	Function
compile	Compiles all Java source code
war	Assembles the WAR file in <sample_dir>/assemble/war/
ear	Assembles the EAR file in <sample_dir>/assemble/ear/
core	Compiles all sources, builds stubs/skeletons and assembles EJB JAR, WAR, and EAR files This is the default target for all build.xml

40

Targets in build.xml (2/2)

Target	Function
docs	Creates Java docs in <sample_dir>/javadocs
all	Builds both core and javadocs, verifies, registers resources and deploys app.
deploy	Register resources, deploys app, but does not install Javadocs
undeploy	Removes the deployed sample from application server
clean	Remove <appname>/build/ and <appname>/assemble/ content

41

Common Commands

- Build codes
  - asant build
- Create a *.WAR file
  - asant create-war
- Deploy/undeploy, use
  - asant deploy full-path-war-file
- Can accomplish all of tasks by executing
  - asant all

42


References
Java Web Services Developer Pack Download
http://java.sun.com/webservices/downloads/webservicespack.html
Java Web Services Developer Pack Tutorial
http://java.sun.com/webservices/downloads/webservicespack.html
Java EE 5 Tutorial
http://java.sun.com/javaee/5/docs/tutorial/
43