Module 1 Content

Part 1 - Introduction

Dear students,

These are your first days in the course! There are some things you need to do during the introduction week.

- The first thing is to become familiar and comfortable with the Blackboard tool; the assignment this week will help. You will post on the Discussion Board and will submit the module assignment.
- You should also make sure to access and to navigate through all parts of the online course, and to become familiar with the Calendar, Assignment and other tools that you will be using regularly.
- Review the course syllabus.
- Make sure to have required textbooks purchased, if you haven't yet done so.
- Check the Special Software Note page (located under “Syllabus & Course Information”) to make sure that you have enough power and memory capacity to run the software needed for our course.

Part 2 – Overview of the Java EE

Challenges of Enterprise Computing

Due to evolving technologies, constant increase of the complexity of underlying infrastructure, and rapid change of business requirements, we are clearly observing the following challenges in architecting enterprise applications:

- A continuous trend towards building and operating more distributed infrastructure and application functionality on multi-tier architectures. Looking back, we can clearly observe an architectural shift from two-tier to three-tier and later to N-tier topology. This trend represents a challenge in architecting, designing, and implementing software components.
- The existence of highly integrated yet heterogeneous hardware and software environments produces another challenge of integration among such diverse pieces of computing.
- Increasing demands on a highly available, yet scalable environment.
- The need for distributed application components to interoperate seamlessly.
- A requirement for high-level security mechanisms for authentication and authorization on accessing application components.
- A requirement for the rapid application development processes and an emerging need for the software reuse strategy are motivations for a unified programming model.
- The need to decouple a business logic code from service level programming routines.
- Access to and integration with existing legacy systems.

What is the Solution?

The Java TM Platform, Enterprise Edition (Java EE) was developed to address many of these challenges by providing a platform-independent, portable, multi-user, secure, and standard enterprise-class platform for server-side deployments written in the Java language.

The Java EE platform represents a single standard for implementing and deploying enterprise applications. The Java EE platform addresses the core issues that impede organizations' efforts to maintain a competitive pace in the information economy. Organizations have recognized this and quickly adopted the new platform standard.

As we cover the Java EE platform, you will learn different types of application components and their inter-operational capabilities as well as variety of services provided by Java EE-centric run-time environment.

The Java EE contains the following:

- API and technology specification
- Development and deployment platforms
- Reference implementation

  This is an operational definition of the Java EE platform. It supposed to prove that the Java EE specification is implementable. It is good for learning and experimental purposes, however is not used for any serious real projects.

- Compatibility Test Suite (CTS)

  This consists of three components:

  1. The first tests for basic API-level compatibility through method signature testing and tests to make sure the APIs in the implementation are neither a superset nor a subset of the required set of APIs.
  2. The second component includes a set of tests to make sure the individual components work properly.
  3. A third part tests for end-to-end compatibility and focuses on the product's use of the technologies in an effort to mandate a minimum level of functionality across all tiers of the Java EE environment. It verifies that platform vendors have correctly implemented to the specification.

  Examples of the products certified by CTS include but are not limited to Oracle Weblogic Server, IBM WebSphere, Adobe JRun, Oracle GlassFish Application Server, RedHat JBoss Server, SAP AG NetWeaver Application Server, and few others.

- Java EE Blueprints

  These include a set of best practices and a collection of design patterns to facilitate and ease a process of creating better applications-products.

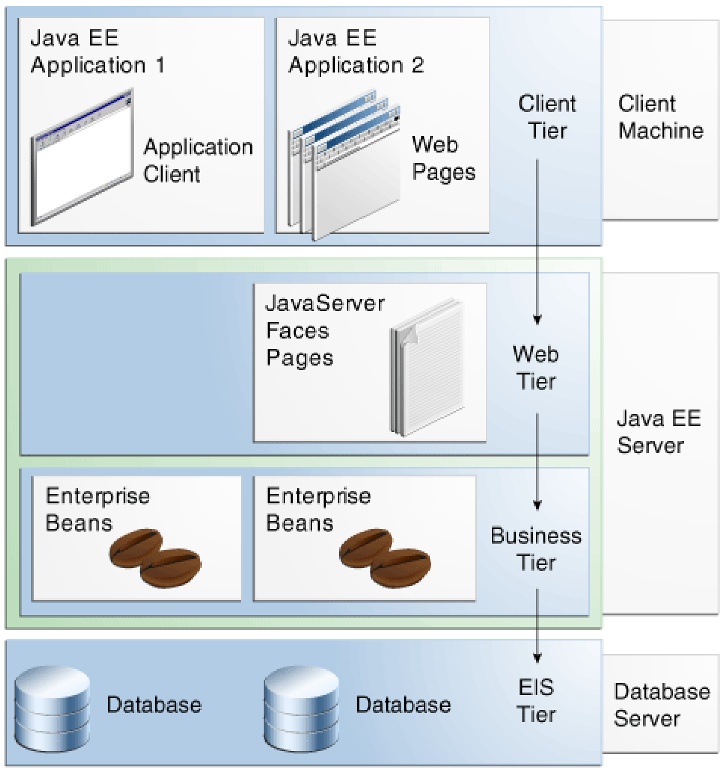
Why is Java EE the Solution

So, why are we saying that the Java EE is the solution to the challenges we talked about on the previous page?

Well, let's go through the bolts and nuts of the Java EE and see why.

- The Java EE defines the standard for developing multi-tier enterprise applications. The Java EE simplifies enterprise applications by basing them on standardized modular components, by providing a complete set of services to those components, and by handling many details of application behavior automatically without complex programming. As a single standard that can sit on top of a wide range of existing enterprise systems—database management systems, transaction monitors, naming and directory services, and more—Java EE breaks the barriers inherent between current enterprise systems. The unified Java EE standard wraps and embraces existing resources required by multi-tier applications with a unified component-based application model. This enables the next generation of components, tools, systems, and applications for solving the strategic requirements of the enterprise.

- The Java EE platform is essentially a distributed computing platform facilitating the design, development, assembly, and deployment of component-oriented enterprise applications. The Java EE platform is most relevant and best suited to supporting cross-platform, secure, transactional applications that expose corporate information to both Internet and Intranet clients.
- A Java EE application is a collection of software components that are engineered to be distributed across multiple computing tiers. The Java EE platform is server-centric and hence Java EE applications typically offer services to a diverse set of clients. A Java EE application is not required to be distributed and should be engineered so as to facilitate the utilization of a distributed computing environment if business or technical requirements warrant.
- A major goal of the Java EE application programming model is to reduce programming effort. One important way this is accomplished is by shifting the burden of implementing common tasks to the Java EE platform. These tasks include security checks, transaction control, as well as other management tasks. Developers select the behavior they desire from a set of standard alternatives. The set of choices a developer makes is maintained in the application as Java EE declarations, which is external to the actual code. The declarative definition of application behavior is a key part of the Java EE programming model. Application development is distinct from the acts of application assembly and deployment. The Java EE programming model promotes a model that anticipates growth, encourages component-oriented code reusability, and leverages the strengths of inter-tier communication.
- The Enterprise Java Beans (EJB) middleware component standard was developed with the following objectives:
    - To provide component interoperability. Enterprise beans developed with different tools will work together. Also, beans developed with different tools will run in any EJB environment.
    - To provide an easy-to-use programming model while maintaining access to low-level APIs.
    - To address lifecycle issues including development, deployment, and runtime.
    - To provide for compatibility with existing platforms allowing existing products to be extended to provide support for EJB.
    - To maintain compatibility with other Java APIs.
    - To provide interoperability between EJB and non-Java applications.
    - To be compatible with CORBA.
- The focus of the EJB standard is therefore on creating interoperability standard for Java middleware shielding programmers from having to deal with many of the difficult issues that arise when developing distributed applications. This allows software developers to concentrate on business logic instead of writing sophisticated homegrown infrastructure and tools. As a result businesses can put most of their educational resources into training staff in business processes which typically is what provides the greatest payoff.



Adopted from http://docs.oracle.com/javaee/7/tutorial/overview.htm#BNAAW

Within the Java EE architecture, we usually break up layers into the following components:

- **Client-side presentation layer** is an environment where "thin" client applications, like web browsers with HTML documents and perhaps Java applets, Java scripts, ActiveX controls and other executables could reside.
- **Server-side presentation layer** is where web servers run. Those host such application components as Java Servlets, JavaServer Pages (JSPs), JavaServer Faces (JSF), scripts to customize look-and-feel, workflow logic and static HTML pages.
- **Server-side business logic layer** is where application server resides. It provides necessary environment to run business logic components, Enterprise Java Beans (EJBs). It also provides a whole set of services, scalability and high availability (load balancing and fail-over functionality) for the infrastructure.
- **Enterprise information layer** where databases and some kind of data logic components may reside.

Parts of Java EE Application Model

The Java EE application model divides enterprise applications into three fundamental parts:

- components
- containers
- connectors

Components

Components are the key focus of application developers, while system vendors implement containers and connectors to conceal complexity and promote portability.

Containers

Containers intercede between clients and components providing services transparently to both including transaction support and resource pooling. Container mediation allows many component behaviors to be specified at deployment time rather than in program code.

- The lifecycle of an application component is governed and managed by Java EE containers. Components can therefore be thought of as residing in "vessels" optimized for a specific computing tier. The precise services offered by Java EE containers are a function of the Java EE specifications, vendor-specific extensions, and implementation details of their platform.
- Containers are system-level entities that provide runtime support for Java EE components. Containers provide components with services such as life cycle management, security, deployment, and threading. Because containers manage these services many component behaviors (e.g. security, transactions) can be declaratively customized (via the deployment descriptor file) when the component is deployed in the container.
- Types of Containers:
    - **Application Client Container** - manages the execution of application client components. Application clients and their container run on the client.
    - **Applet Container** - manages the execution of applets. It consists of a web browser and Java Plug-in running on the client together.
    - **Servlet Container** - provides network services (by which requests and responses are sent), decodes requests and formats responses. Required to support HTTP as a protocol for requests and responses. May optionally support additional request-response protocols such as HTTPS.
    - **JSP Container** - provides the same services as a Servlet container and an engine that interprets and processes JSP pages into a servlet.
    - **Web Container** - provides the same services as a JSP container and access to the Java EE service and communication API.
    - **EJB Container** - provides a range of transaction, security and persistence services and access to the Java EE service and communication API.

Connectors

Connectors sit beneath the Java EE platform defining a portable service API to plug into existing enterprise vendor offerings. Connectors promote flexibility by enabling a variety of implementations of specific services.

Portability

Java EE component portability is primarily a function of the dependency an application component has on the underlying Java EE container. Components using a vendor-specific feature that falls outside of the Java EE requirements may have limitations in the area of portability. However, the Java EE specifications do spell out a base set of capabilities that a component can count on.

Hence, there is a minimum cross-container portability that an application should be able to achieve. Needless to say, an application developer expecting to deploy on a specific vendor implementation of the Java EE platform should be able to do so across a wide range of operating systems and hardware architectures. The Java EE standard includes complete specifications and compliance tests to ensure portability of applications across the wide range of existing enterprise systems capable of supporting Java EE.

Java EE Application Scenarios

The Java EE architecture allows a variety of application scenarios to address the broad scope of enterprise-wide systems.

The subject is presented in great detail in Chapter 1.3, Java EE Application Scenario of the Sun Enterprise Team book "Designing Enterprise Applications with J2EE Platform, Second Edition": http://www.adobe.com/support/jrun/documentation/pdf/j2ee_ent_app_design.pdf

While considering an architecture for an enterprise application we usually face a variety of options in particular user cases that would lead to different application scenarios. The Java EE model is rich, flexible, and supports a wide range of such scenarios. We can construct practically any combination of multi-tier layers and software components supported by Java EE services to satisfy a given business model.

Our choice to use a particular architecture depends on such factors as the:

- necessity to change any components quickly without affecting other components.
- simplicity of the process of software development.
- ability to better separate roles.
- ability to conduct a software reuse paradigm.

Java EE can provide an easy way to break-and-build application logical components. We can learn the Java EE implementation of this model by studying the "Java PetStore" application. This is a Java EE sample application provided by Sun as a part of Java EE reference model. This sample application is a great example and learning tool. You can download the Java EE programming model, design patterns, and practices from this webpage, http://www.oracle.com/technetwork/java/index-jsp-136701.html; look for Enterprise Java Pet Store Sample Application.

History of Java EE Specification

The Java EE Specification has about 15 years of history. It goes all the way back to 1999-1998 with the first introduction of the J2EE 1.1. After that time, Java EE platform and its specification went long way, exhibiting lots of enhancements, improvements, introducing new ways of designing and developing application components and connecting them using communication protocols, etc.

The latest Java EE specification 7 came out in 2013, introducing yet several new powerful features and improvements.

For the complete list of separate Java EE specifications and APIs that are included in the Java EE 7, please see Antonio Goncalves’s book, chapter 1.

Note: At the time of preparing these lecture notes, only one web application server, Oracle GlassFish, is supporting Java EE 7 APIs and is available for working with Java EE 7 programming model components.

Web components services	
• Java Servlets • Java Server Pages (JSP • JavaServer Faces (JSF) • JSTL • WebSockets • Expression Language (EL)	javax.servlets javax.servlets.http javax.servlets.jsp
Database access services	
• Java Database Connectivity (JDBC)	javax.sql
• Java Naming and Directory Interface (JNDI)	javax.naming, javax.naming.directory javax.naming.event javax.naming.ldap javax.naming.spi
Messaging services	
• JavaMail • Java Message Service (JMS) • JavaBean Activation Framework (JAF)	java.mail javax.jms javax.activation
Distributed Objects Access	
• Remote Method Invocation (RMI) • Internet Inter-ORB Protocol (IIOP) • Java IDL (for CORBA objects interoperability)	javax.rmi javax.rmi.CORBA
Transaction Processing services	
• Java Transaction Service (JTS) • Java Transaction API (JTA)	javax.transaction
Enterprise Java Beans (EJBs)	
• A component model specification for server side. • A framework for deployment of business components. • A model which is customizable at deployment time. • Support for any transport protocol.	javax.ejb javax.ejb.deployment
Security services	

• Secured socket layer (SSL) • Access control list (ACL) • Security model Signed JAR files	

As we move on through our course we will be studying a number of these technologies and working with specific APIs as we develop different parts of our project. At this time I want you to get a brief introduction to them through our textbooks and other resources.

Java EE Separation of Roles

Java EE creates markets and partitions developer responsibility with its concept of platform roles.

The Java EE specification layers responsibilities into roles that various types of developers, integrators, and deployers might play. Each role produces a particular product such as a server, a component, or an installed application.

The roles defined by the Java EE specification are as follows: (note that this section is essentially a quick paraphrase of the corresponding section of the Java EE specification, section 2.4.):

- **Java EE Product Provider:** Implements and provides either Java EE component containers or APIs defined by the Java EE specification. Product providers also supply tools that work with their products. For example, a Java EE product provider might provide an EJB container plus the tools necessary to deploy the enterprise beans into the container.
- **Application Component Provider:** Creates application components such as HTML or JSP pages, servlets, EJBs, or other components that run within a Java EE container. For example, a developer who creates an EJB suite for enterprise scheduling would be an application component provider.
- **Application Assembler:** Combines existing Java EE components into an enterprise application by packaging them into an "ear" (**E**nterprise **AR**chive) file for later deployment. The application assembler identifies and documents dependencies between the application and the environment in which it runs. These dependencies are resolved during the deployment process to the target platform. For example, someone who assembles components into an online shopping application would be an application assembler.
- **Deployer:** Installs and configures a Java EE application in a particular environment. This role requires both knowledge of the particular Java EE application and the installation environment. For example, a deployer might know how to install a particular Java EE-compliant Web server and servlet container on any of several operating systems.
- **System Administrator:** Maintains the hardware and software infrastructure underlying the Java EE system. System administrators perform resource management and runtime monitoring tasks, usually with tools provided by the product provider. For example, a database administrator is one type of system administrator.

Summary on Enterprise JavaBeans (EJB)

EJB is a server-side software component that can be deployed in a distributed multi-tier environment.

EJB is a key component of Java EE programming model.

With EJB model existence there is a clear distinction in roles played by different parties:

- The bean provider

- The container provider
- The server provider
- The application assembler
- The deployer
- The system administrator

There were two types of beans in EJB 1.1 specification:

1. Session bean - usually represents a business action or function (to perform a bank transaction or to apply for customer account). Session beans can be **stateless** or **statefull**. Session beans are **not persistable**.
2. Entity bean - is a business entity (bank account or customer account) that otherwise would reside in a database. Entity beans are **object data models** and are **always persistable**.

EJB 2.0 has introduced a new kind of enterprise bean, **Message-Driven Bean**. This is a special kind of a bean that can receive JMS messages, and therefore communicates with the rest of the world in asynchronous mode.

From EJB 3.0 there are only two types of beans, Session beans and Message-Driven beans, and Entity beans have been removed, replacing those with Java Persistence API entities.

Summary of Java EE 6 and Java EE 7 Specifications and related JSRs

Java EE 6		**Java EE 7**	
Java EE 6 specification	JSR 316	Java EE 7 specification	JSR 342
WS Tech-s		**WS Tech-s**	
JAX-RS 1.1	JSR-311	JAX-RS 2.0	JSR-339
Implementation Enterprise Web Services 1.3	JSR 109	Implementation Enterprise Web Services 1.3	JSR 109
JAX-WS 2.2	JSR 224	JAX-WS 2.2	JSR 224
JAXB 2.2	JSR 222	JAXB 2.2	JSR 222
WS Metadata	JSR 181	WS Metadata	JSR 181
JAX-RPC 1.1	JSR 101	JAX-RPC 1.1	JSR 101
JAXM 1.3	JSR 67	JAXM 1.3	JSR 67
JAXR 1.0	JSR 93	JAXR 1.0	JSR 93
Web App Tech-s		**Web App Tech-s**	
Java Servlets 3.0	JSR 315	Java Servlets 3.1	JSR 340
JSF 2.0	JSR 314	JSF 2.2	JSR 344
JSP 2.2	JSR 245	JSP 2.3	JSR 245
EL 2.2	JSR 245	EL 3.0	JSR 341
JSTL 1.2	JSR 52	JSTL 1.2	JSR 52
Debugging Support for Other Lang-s 1.0	JSR 45	WebSockets	JSR 356
		JSON Processing	JSR 353

Ent. Apps Tech-s		**Ent. Apps Tech-s**	
CDI (WebBeans) 1.0	JSR 299	CDI 1.1	JSR 346
DI 1.0	JSR 330	DI 1.0	JSR 330
Bean Validation 1.0	JSR 303	Bean Validation 1.1	JSR 349
EJB 3.1 (w/Interceptors 1.1)	JSR 318	EJB 3.2	JSR 345
Java EE Connector Arch. 1.6	JSR 322	Java EE Connector Arch. 1.7	JSR 322
JPA 2.0	JSR 317	JPA 2.1	JSR 338
Common Annotations 1.1	JSR 250	Common Annotations 1.2	JSR 250
JMS 1.1	JSR 914	JMS 2.0	JSR 343
JTA 1.1	JSR 907	JTA 1.2	JSR 907
JavaMail 1.4	JSR 919	JavaMail 1.5	JSR 919
		Batch Apps for Java Platform	JSR 352
		Concurrency Utilities 1.0	JSR 236
		Interceptors 1.2	JSR 318
Mngmt and Security Tech-s		**Mngmt and Security Tech-s**	
Java Authentic. SP Interface for Containers	JSR 196	Java Authentic. SP Interface for Containers	JSR 196
Java Authoriz. Contract for Containers	JSR 115	Java Authoriz. Contract for Containers	JSR 115
Java EE App-s Deployment 1.2	JSR 88	Java EE App-s Deployment 1.2	JSR 88
J2EE Management 1.1	JSR 77	J2EE Management 1.1	JSR 77
		Debugging Support for Other Lang-s 1.0	JSR 45
JEE related specs in J2SE		**JEE related specs in J2SE**	
Java API for XML Processing (JAXP) 1.3	JSR 206	Java API for XML Processing (JAXP) 1.3	JSR 206
Java Database Connectivity 4.0	JSR 221	Java Database Connectivity 4.0	JSR 221
Java Management Extensions (JMX) 2.0	JSR 255	Java Management Extensions (JMX) 2.0	JSR 003
JavaBeans Activation Framework (JAF) 1.1	JSR 925	JavaBeans Activation Framework (JAF) 1.1	JSR 925
Streaming API for XML (StAX) 1.0	JSR 173	Streaming API for XML (StAX) 1.0	JSR 173

Part 3 - Web Application Server Installation

What is Oracle WebLogic Application Server?

This week we begin learning about and working with one of the most popular and powerful Java EE implementation platforms, Oracle WebLogic Application Server (often called as **WebLogic Server** or **WLS**). As the course progresses, we will be studying technical documentation around functionality, services, APIs, operational, administrative, and monitoring capabilities of this product, reinforcing reading with hands-on assignments. Although the main objective for the course is to learn about the Java EE technology as a foundation of enterprise Java computing, you will need to use a product that implements the Java EE specifications and provides hosting environment for programming assignments in this course. The WebLogic Server is your best choice.

The WebLogic Server is widely used Java application server for hosting multi-tier, secure, large-scale, distributed web applications. These types of applications require sophisticated, fast, fault-tolerant networked communication among application tiers and components. In modern web-centric architecture, many enterprises have moved from client-server environments to n-tier solutions with efficient network connectivity as a critical factor.

In the multi-tier architecture, the WebLogic Server provides a framework for developing and deploying server-side business logic. It supports a distributed programming model that hides the complexity of the environment from the programming effort.

As it was stated earlier, the WebLogic Server is a product that implements Java EE specifications (including all application components, connectivity protocols, security realms, transaction handling, messaging and more):

- It maintains and manages application logic and business rules for a variety of clients, such as web browsers, applets, Java and non-Java applications running on client machines.
- It supports software clustering of the WebLogic Server instances for running both Web and EJB services to ensure reliability, scalability, and high performance.
- It provides application services necessary for building robust, scalable, web-based applications.
- It provides current and complete implementation of the communication protocols of Sun's Java EE specification.

Before we start, let me address one of the questions that students usually ask me at the beginning of each semester:

- **Why do we use the WebLogic Server vs. other web application servers?**
- **Why not let's say JBoss or Oracle's GlassFish or IBM's WebSphere Application Server?**

Well, I think there are several reasons for that:

- First, the WebLogic Server is the most used web application server out there. Its market share is somewhere around 35–40% (according to latest Gartner reports). Most likely, some of you might already have some business related to this product, or soon will have.
- Second, this is a commercial product that has a very good documentation, and is easy to install and to configure. This latter characteristic is critical in an educational environment.

Remember that our prime objective is learning the Java EE technology and programming model, using one of the existing application server products. But while we are doing so, we should not spend too much time on learning the product itself. The WebLogic Server is a perfect choice.

By the way, this product comes with free evaluation license that we will be using during the semester.

Downloading WebLogic Server

Your next step is to download and to install the latest release of the WebLogic Server.

- *Pre-install considerations.*
    - Go to https://login.oracle.com/mysso/signon.jsp and sign up for a free Oracle Web Account (if you haven't yet done so).
    - Check your system requirements to make sure you have an adequate hardware and software platform for installing and running WLS. For a complete list of supported hardware/software go to Supported Configurations for WebLogic Platform document, see

    - http://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html
  - About Java Development Kit (JDK) version required by WebLogic Server.
    - It is important to verify the required JDK version by reviewing the certification information on the *Oracle Fusion Middleware Supported System Configurations* page.
    - For WLS 12*c* release 12.2.1.2, the certified JDK is 1.8.0_101 and later.
    - In order to get ready for WebLogic Server installation process, please start and run DOS shell as Administrator.
    - Set JAVA_HOME variable by running “**set JAVA_HOME=c:\jdk1.8.0_112**” command.
    - Set PATH variable by running “**set PATH=%JAVA_HOME%\bin;%PATH%**” command.
    - Validate Java version by running “**java –version**” command.

Note: Be aware that next steps might look different depending upon Oracle's decision to change download process, URLs involved, installation file names, etc. In this text, I am using the latest information available at the time of writing.

- *Download WebLogic Server.*
  - **Step 1**. Use the following link to get to Oracle WebLogic Download Center page: http://www.oracle.com/technetwork/middleware/weblogic/downloads/wls-main-097127.html
  - **Step 2**. Click on "Accept License Agreement" radio button (located at the top of the page). Then, under "Oracle WebLogic Server 12.2.1.2" section, click on “Generic Installer” link to start downloading the installation file.
  - **Step 3**. The installation file called **fmw_12.2.1.2.0_wls_Disk1_1of1** (approx. 791 MB size) will be downloaded.
  - **Step 4.** Unzip this file to the directory that you will use later for installing WebLogic Server.

Installation Steps

Now we are ready to install WebLogic Server.

- After saving the file, open the following WebLogic Installation Instructions document and follow the steps there to install WebLogic Server, see https://docs.oracle.com/middleware/12212/lcm/WLSIG/GUID-E4241C14-42D3-4053-8F83-C748E059607A.htm#WLSIG200

  - **Step 1.** From DOS shell, within the directory where the installation file was saved, run “**java -jar fmw_12.2.1.2.0_wls.jar**” command. It will start the installation wizard.
  - **Step 2**. On “Auto Updates” frame, leave it unchanged and click on “Next”.
  - **Step 3**. On "Installation Location” frame, provide home directory for WLS installation, i.e. C:\Oracle\wls12.2.1.2, and click "Next".
  - **Step 4**. On "Installation Type" frame, select "Complete with Examples" radio button and click "Next".
  - **Step 5**. On "Prerequisite Check" frame, make sure that all checks are OK and click "Next".
  - **Step 6**. On "Security Updates" frame, uncheck “I wish to receive …” and click "Next".
  - **Step 7**. Review Installation Summary frame and click on “Install”.
  - **Step 8.** Observer installation progress.
  - **Step 9**. Get the "Installation Complete" window, uncheck "Run Quick Start" box and click "Done".

Post-Installation Steps

Post-Installation steps are described in the "Post-Installation Tasks" document:

https://docs.oracle.com/middleware/12212/otd/install/GUID-D547A046-DD5E-4F7E-92B9-3886935BF03E.htm#OTINS1112

- **Step 1.** Start by learning the WLS directory structure. Make sure to understand the content of each subdirectory. I assume that you installed the product into **c:\oracle\wls12.2.1.2** directory, so proceed from there. (Otherwise, make appropriate adjustments to my text.)
- **Step 2.** Get into **wls12.2.1.2** subdirectory, navigate through the directories. Here are a few important subdirectories to be aware of:
  - **user_projects** – this is the location of WebLogic Server Domains and their files.
  - **wlserver/samples/server** – this is the location of sample code and its documentation.

OK, now you are ready to make a next step in preparation of your run-time WebLogic Server environment.

Create Your WebLogic Server Domain

Overview and Steps

The WebLogic Server represents a significant size and high level complexity software product. Its architecture is based on a concept of *domains* and *servers*. I recommend you spend a few minutes reading the following document "Understanding WebLogic Server Domains":

http://www.oracle.com/webfolder/technetwork/tutorials/obe/fmw/wls/12c/12_2_1/01-04-004-CreateDomain/creatingadomain.html

We are going to simplify WebLogic Server domain configuration as much as possible, so that it is not going to create extra challenge learning about it and working with it. I definitely don't want students to spend too much time on figuring out how WebLogic Server works and how to administer it, because that's not our focus and objectives. Let's remember that we are using this product only as an implementation platform of choice for learning Java EE. So, we are going to keep our run-time environment simple. Specifically, we are going to create and maintain a single WebLogic Server domain that is going to be non-clustered and will contain one stand-alone server instance that will perform administrative functions (so called *Admin Server*) and will serve as run-time environment at the same time (so called *Managed server*).

So, let's now create your own WebLogic Server domain.

- **Step 1. Starting GUI-based Domain Configuration Wizard.**

  Let follow the steps described in "Creating WebLogic Domains" document, see

  http://www.oracle.com/webfolder/technetwork/tutorials/obe/fmw/wls/12c/12_2_1/01-04-004-CreateDomain/creatingadomain.html

  I recommend starting Domain Configuration Wizard from MS-DOS command shell:

    - Open an MS-DOS command prompt window (on Windows).
    - Go to the **wls12.2.1.2\oracle_common\common\bin** subdirectory of the product installation directory.
    - Invoke the following scripts to start the Configuration Wizard in graphical mode: **config.cmd**.

  At this point GUI-based wizard will start.

- **Step 2. Running the Wizard.**
    - On “What do you want?” frame, leave "Create a new domain" checked, and change “base_domain” to “xyz_domain” in “Domain location” text field (change xyz here to your initials, like my WLS domain name is “lf_domain”), leave Domain Location as is (it should be C:\oracle\wls12.2.1.2\user_projects\domains) then click "Next".
    - On “Templates” frame, leave "Create Domain Using Product Templates" checked, leave “Basic WebLogic Server Domain” checked (by default)
    - On "Administrator Account" frame, provide admin username and password of your choice. You can type in any arbitrary username and password (password has to be at 8 characters minimum). Click "Next".
    - On "Domain Mode and JDK" frame, leave start mode "Development" checked and Oracle’s Java SDK x.x.x selected and click "Next".
    - On "Advanced Configuration" frame, leave it unchanged and click "Next".
    - Observer the information on "Configuration Summary" frame and click "Create".
    - The wizard runs for a few seconds. Click on "Done".
- **Step 3. Studying WebLogic Domain Directory Structure.**

  Using Windows File Explorer, get into **C:\oracle\wls12.2.1.2\user_projects\domains\XYZ_Domain** directory (in your case, XYZ_Domain is whatever you typed in for the Domain Name) and study its content. I recommend that you use the following Oracle document to help navigating through, "Understanding Domain Configuration. Domain Configuration Files":

  https://docs.oracle.com/middleware/12212/wls/DOMCF/understand_domains.htm#DOMCF115

  The most important components (files and directories) in it are:

    - **startWebLogic.cmd** script to start WebLogic Server instance. You will be using this script to start the server. Optionally, you can use the same script, startWebLogic.cmd, located in **C:\oracle\wls12.2.1.2\user_projects\domains\XYZ_Domain\bin**.
    - **stopWebLogic.cmd** script to stop WebLogic Server instance, located in the same directory.
    - **setDomainEnv.cmd** script to set environment variables, like CLASSPATH. You will be using this script to work on developing, compiling, packaging and deploying Java EE application components.
    - **AdminServer.log** and XYZ_Domain.log files in **C:\oracle\wls12.2.1.2\user_projects\domains\XYZ_Domain\servers\AdminServer\logs**. Those are log files maintained by WebLogic Server at run-time. You will be using those files for analyzing problems and errors.

  OK, now you are ready to start your WebLogic Server instance running and to take a first look on some useful administration utilities.

Starting WLS

To start your WebLogic Server (WLS), you should take the following steps.

- **Step 1**. Go to the DOS command shell and then to **%WLS_HOME%\user_projects\domains\XYZ_Domain\** directory. Please make sure to use your domain name (see previous section) in place of XYZ_Domain.
- **Step 2**. Execute **startWebLogic.cmd** script to start your WebLogic Server instance**.**
- **Step 3**. Observe the stream of messages on your DOS console. Once you get it started, you will see the following message:

  ```
  <Notice> <WebLogicServer> <BEA-000360> <Server started in RUNNING mode>
  ```

  At that time, you are ready to start WLS administration console from your browser.

- **Step 4**. To verify that the server is working OK, open a web browser screen and request the following URL: **http://localhost:7001/console**

  Remember that Admin Console is a secured application. In order to login to it, you have to type proper UserID and Password that you have defined while creating this WLS domain in previous steps. Once you have logged in, if you see the Admin Console screen coming back from the WLS, it means that your server is running OK and you can communicate with it.

For more details on starting WLS, I recommend you review Oracle WLS document, "Starting and Stopping Servers: Quick Reference", see:

https://docs.oracle.com/middleware/12212/wls/START/overview.htm#START112

Working with WLS Admin Console

Now, having your WebLogic Server instance is up and running, please get into and study WLS Admin Console image on the Web browser screen. I recommend you use "The WebLogic Server Administration Console" document to learn about this tool:

http://docs.oracle.com/middleware/12212/wls/WLACH/

It explains the console's layout, different functions and controls, and what a user can do on Admin console to control WLS configuration and execution.

The Admin Console application is protected and requires UserID and Password (whatever you've provided when creating your WLS domain).

As you read the Oracle’s WebLogic Server document, please be aware of the following:

> During the course, you will be working with a simplified configuration of the WebLogic Server domain, which is a domain with one WLS instance, working both as Admin Server and as a Managed Server. You will not be using a WLS clustered environment. As a result, you will not be starting and stopping a WLS instance using an Admin Console. Nor will you be configuring WLS clusters using this tool.

Before doing anything on the Admin Console, make sure to read the Important Note below.

Important Note: At this point, do not make any changes on it! It might bring WLS into "out-of-service" condition!

Now, browse through the Admin console on your own and see how to get different controls and parameters displayed. For example, on the left hand side, find "Domain Structure" panel, in it there is "Diagnostics" link. If you click on it, it will give you drop-down men u of other links. One of them is "Log Files' link. Click on it and observer the list of different log files on the right hand side panel. If you select any of those log files, you can view it right on this console window. This is very handy feature while debugging or tuning your application.

Working with WebLogic Scripting Tool (WLST)

Your next step is to become familiar with WebLogic Scripting Tool, WLST, which is a command-line utility and one of several interfaces for managing and monitoring WebLogic Server.

For official documentation, please use the following:

1) **“**Use WebLogic Scripting Tool 12.2.1.2.0” at

   https://docs.oracle.com/middleware/12212/cross/wlsttasks.htm

2) “Oracle® Fusion Middleware WLST Command Reference for WebLogic Server” at

   http://docs.oracle.com/middleware/1221/wls/WLSTC/toc.htm

3) “WebLogic Server 12c (12.2.1): Using the WebLogic Scripting Tool“ at

   http://www.oracle.com/webfolder/technetwork/tutorials/obe/fmw/wls/12c/12_2_1/01-32-001-UsingWLST/Using_WLST.html

The WebLogic Scripting Tool (WLST) is a command-line scripting environment that you can use to create, manage, and monitor WebLogic domains. It is based on the Java scripting interpreter, Jython. In addition to supporting standard Jython features such as local variables, conditional variables, and flow control statements, WLST provides a set of scripting functions (commands) that are specific to WebLogic Server. You can extend the WebLogic scripting language to suit your needs by following the Jython language syntax (see http://www.jython.org).

Like Admin Console, for the majority of commands this utility assumes the role of a client that invokes administrative operations on the WLS Admin Server, which is the central management point for all servers in the domain. (All Managed Servers retrieve configuration data from the administration server, and the Administration Server can access runtime data from all Managed Servers.)

WLST can be used for connecting to a running Administration Server and managing the configuration of an active WebLogic domain, view performance data about resources in the domain, or manage security data (such as adding or removing users).

WLST can be invoked in a variety of ways:

- Interactively, on the command line;
- In batches, supplied in a file;
- Embedded in Java code.

For the purpose of this course, I suggest using starting and working with WLST in interactive mode.

The following document describes the steps to invoke WLST in interactive mode and different commands that can be used with it, see

http://www.oracle.com/webfolder/technetwork/tutorials/obe/fmw/wls/12c/12_2_1/01-32-001-UsingWLST/Using_WLST.html

So, from c:\oracle\wls12.2.1.2\oracle_common\common\bin, please run **wlst.cmd** script.

Once you get back **wlst:/offline** prompt, it means that you can start interacting with WLST. Here is a few important commands:

- help(‘offline’)
- help(‘connect’)
- connect('admin_userId','admin_password','t3s://localhost:7001')
- version
- state(‘x’)
- listApplications
- shutdown()
- exit()

and many other functions.

Stopping WLS

There are several ways to shutdown the WebLogic Server instance.

One way is to use the Admin Console. On the left hand side "Domain Structure" panel, you can click on "Environment" --- "Servers", Then click on "AdminServer (admin)" link on the right hand side panel, then click on "ControL' tab on the top of the right hand side "Settings for Admin Server" panel. At the bottom of next panel find section "Server Status", then "Shutdown" drop-down menu. You can select "When Work Complete" or "Force Shutdown now" option to shut down your WLS instance. After that on the right hand side panel, click on link to confirm your intention.

However, because you will be running the instance of the WebLogic Server that you are connected to through the Admin Console, do not be surprised if you lose a connection to it immediately after it starts the shutdown process. It is OK to use this method when you have multiple instances of managed servers running under control of one instance of the administration server. In the case of our environment, you will run only one instance to the server for both admin and managed roles. That is why shutting the server down using the Admin Console is the same as shooting yourself in the foot.

Another way to shut down the WLS instance is to use WLST utility. The WLST utility can run a SHUTDOWN () command that would bring a running instance of a WebLogic Server down. This is how I suggest you always shutdown your WLS instance.

Note: You can also "kill" the process of running WLS by using the operating system's "killing" capabilities. I would never recommend doing that, unless in an emergency (like when you can't connect to your server, no matter what you do).