

# Glossary of J2EE terms for Forms and Designer Developers

Last updated 28th June 2005

The Java Platform brings whole host of opportunities and possibilities to the world of application development. However, the Forms and Designer developer is in danger of being drowned in the sea of new terms and acronyms. There are numerous J2EE glossaries available on the web, so this glossary is written specifically for Forms and Designer developers and explains, in terms with which you are already familiar, what these new concepts are. This is not meant to be an exhaustive list so please feel free to post any other acronyms/terms to [my blog](#).

A B C D E F G H I J K L M N O P Q R S T W X Y Z

A		
ADF	<b>Application Development Framework</b>	Oracle ADF is a comprehensive framework for <a href="#">J2EE</a> developers to help build <a href="#">J2EE</a> applications. In a similar way, Forms provides you with a "black box" framework that does things like persisting data you query from the database, locking records, mapping the data from your UI to the underlying database etc.. Infact, ADF does more than that and does so as a "white box" meaning you can change the code if you need. However, the analogy is that ADF provides a framework to do all the "plumbing" so you can concentrate on the business logic of your application.
ADF BC	<b>ADF Business Components</b>	Where ADF is a complete framework, ADF BC is one choice within <a href="#">ADF</a> that allows you to build the business services for your J2EE application. Business services are the part of the framework used for implementing your business logic and providing the ability to connect to the database. In Oracle Forms, you don't have to worry about how your data gets retrieved and sent to the database, Forms does it for you. This is the job that ADF BC performs. Although there are other technology choices for building the business services, ADF BC is closest to the development paradigm implemented by Oracle Forms.
ADF Faces		A fully <a href="#">JSF</a> compliant library of enhanced components for building web user interfaces. Includes color pickers, date choosers and tree controls.
Ant		<i>Coming Soon</i>
Application Module		In <a href="#">ADF Business Components</a> , the application module is a logical grouping of <a href="#">VO</a> instances relating to a specific business transaction. So, in the same way a single Form usually encompasses a number of blocks and is related to a defined business transaction (like maintaining employee details for a specific department), the application module is made up of <a href="#">VO</a> instances and performs a similar role.
B		
BC4J	<b>Business Components for Java</b>	Business Components for Java was a precursor and a subset of what has now grown into Oracle ADF (although <a href="#">ADF</a> is much more). It provided persistence and a framework for business logic. This functionality is now included in <a href="#">ADF</a> as <a href="#">ADF Business Components</a> .
C		
Container		A container is essentially the generic name for a <a href="#">J2EE</a> environment

		that provides runtime services, such as security, to the program running in that container. The container and any application code you write will run on a <a href="#">JVM</a> . You can get different containers which provide different services - e.g. an <a href="#">EJB</a> container or a <a href="#">servlet/JSP</a> web container. A loose mapping can be made to the Forms runtime engine which provides a number of services (analogous to a container) which can be used when running FMX files.
CSS	<b>Cascading Style Sheet</b>	In Oracle Forms if you want to give a common look and feel to your application you would define Visual Attributes and then assign these visual attributes to the user interface. A Cascading Style sheet is the same concept. It is a text file which defines a set of tags and visual properties to each of these tags. An HTML page can then be annotated with the CSS tags to give a visual appearance to the user interface. And by changing the CSS, the application can inherit a new visual appearance - exactly as you did with visual attributes.
<b>D</b>		
Design Pattern		Generally speaking - it is a "best practice"; an often used solution to a well defined problem. So a description of proved techniques and best practice for developing applications could be called a design pattern.
Domain		<i>Coming Soon</i>
<b>E</b>		
EAR	<b>Enterprise Archive</b>	When it comes to deploying a Forms application you need to get all your FMX, PLL and MMX files and make sure you put them on the correct place on the application server before you can run the application. Wouldn't it be great if you could zip up all of these files in the correct structure and just "drop" them onto your new application server? Well <a href="#">J2EE</a> already gives you that. The EAR file allows the <a href="#">J2EE</a> application to be bundled and deployed as a single file.
EJB	<b>Enterprise Java Bean</b>	Part of the <a href="#">J2EE</a> specification, the EJB specification defines an object or service in which you can place business logic and which is also responsible, amongst other things, for persisting data from the database, providing transactions, security, etc.; and all across a distributed application. The advantage of using EJB for your business logic versus a plain old Java object (POJO) is that the EJB specification, and the container in which it runs, defines a number of services and extra functionality you therefore don't have to manage yourself. Again, it is hard to find an analogy in the Forms world but if you can imagine a standardised version of the Forms record manager (which is responsible for locking records etc.) then you are in the right area.
EO	<b>Entity Object</b>	In <a href="#">ADF</a> Business Components, the EO is an object which represents the underlying database table. So the EO is doing a similar job to a Forms block in that in the middle tier it is persisting the data of the underlying database table (as well as providing some validation). It can almost be seen like a "copy" of the database table held inside the application code.
<b>F</b>		
<b>G</b>		
<b>H</b>		
<b>I</b>		
<b>J</b>		

Java		Programming language developed by Sun. Often used for programming web applications and can run on any platform that has a <a href="#">Java Virtual Machine</a> .
JAZN		<i>Coming Soon</i>
J2EE	<b>Java 2 Enterprise Edition</b>	J2EE is a platform for developing and running Java enterprise applications. It includes the language and the definition of a number of services, protocols and interfaces. There is no real analogy with Oracle Forms since Forms is not written to a public specification. However, if Forms was, it would mean that the FMX file would follow a well defined specification and so other vendors could write their own implementation of a Forms runtime to run on their platform. Thus, the advantage of a defined specification is that anyone can write their own implementation of that specification.
J2ME	<b>Java 2 Micro Edition</b>	Where as <a href="#">J2EE</a> gives you the specification for writing full blown enterprise applications, J2ME is the specification for hand-held, PDA (Personal Digital Assistant). So think of J2EE for mobile types of devices.
J2SE	<b>Java 2 Standard Edition</b>	The core of <a href="#">J2EE</a> comprising of, basically, the Java language and the <a href="#">JVM</a> runtime.
JAAS	<b>Java Authentication and Authorization Service</b>	<i>Coming Soon</i>
JAR	<b>Java Archive</b>	A zip file full of Java classes. In Oracle Forms you are already using JAR files since the Forms client is downloaded automatically to the client machine via a file called F90all.jar
JDeveloper		Oracle's Java Builder. Oracle's tool for building <a href="#">J2EE</a> applications.
JClient		JClient is the part of <a href="#">ADF</a> used for building rich Java user interfaces. Oracle Forms uses a Java UI and JClient gives the similar rich interactive interface. Within <a href="#">ADF</a> , JClient provides the binding and framework responsible for integrating the actual UI components (Using Java Swing) and the underlying data.
JDBC	<b>Java Database Connectivity</b>	A programming interface that allows Java to execute SQL statements. Oracle Forms has a similar type of interface called OCI (Oracle Call Interface) since the Forms runtime is written in C instead of Java. Both interfaces are responsible for logging on, setting up cursors and querying data.
JMS	<b>Java Messaging Service</b>	<i>Coming Soon</i>
JSF	<b>Java Server Faces</b>	As a Forms developer you are used to using UI components to build your user interface. When you move to <a href="#">J2EE</a> , you may start with <a href="#">JSP</a> . As a Forms developer, you may find this like a step back to using code for your user interface instead of UI widgets. Hence the introduction of JSF. A specification/framework that provides a set of UI components (buttons, text fields) as well as the hooks to get data to and from the items and to pass user events. As a Forms developer you take it for granted that Forms did all this for you: UI controls, UI triggers etc. JSF provides that functionality for web UIs.
JSP	<b>Java Server Page</b>	An HTML page with Java code embedded in it. At runtime, the Java code is executed by a <a href="#">servlet</a> and outputs additional HTML tags. In Forms, the data that appears inside a multirow block is passed

		down to the client at runtime and displayed inside the appropriate text fields. An HTML user interface displays data using standard HTML tags which are generated at runtime by some code on the application server.
JSR	Java Specification Request	<i>Coming Soon</i>
JSTL	Java Standard Tag Library	<i>Coming Soon</i>
JVM	Java Virtual Machine	The engine on which Java code is run: in the same way that PL/SQL code is executed in a PL/SQL engine.
<b>K</b>		
MVC	Model View Controller	In the Forms world, the definition of the user interface, business logic and navigation logic is all contained within the FMB/FMX. MVC is an architecture (or often called a <a href="#">design pattern</a> ) in which the implementation for the user interface (view), the business logic (model), and the navigation logic (controller) are implemented separately. The advantage of MVC is that you can develop, for example, different user interfaces (views) ontop of the same business application (model and controller)
<b>L</b>		
<b>M</b>		
<b>N</b>		
<b>O</b>		
OC4J	Oracle Containers for Java	Oracle's implementation of a <a href="#">J2EE container</a> .
<b>P</b>		
<b>Q</b>		
<b>R</b>		
<b>S</b>		
Servlet		Simply a Java program that runs on your application server. Possibly without you knowing it, you are using servlets when you run Forms on the web. The Forms Servlet, for example, is a small Java program that runs on the application server and is used to start Forms on the web by reading the configuration files and converting into HTML to be passed to the browser.
SOAP	Simple Object Access Protocol	The protocol for passing <a href="#">XML</a> based messages across an HTTP connection. Forms uses a proprietary protocol inside HTTP messages for passing data from the browser client to the application server. In the same way, <a href="#">web services</a> use the SOAP protocol for transferring data; and being a standard it can be understood by all services which subscribe to that standard.
Struts		<i>Coming Soon</i>
<b>T</b>		
Toplink		<i>Coming Soon</i>
<b>U</b>		
UDDI	Universal Description Discovery and	In <a href="#">web services</a> , UDDI is simply a directory of available web services - like a yellow pages of web services.

	Integration	
UML	Unified Modelling Language	Coming Soon
<b>V</b>		
VO	View Object	In <a href="#">ADF Business Components</a> , the VO defines the query on top of the entity objects (EO). So a view object can define a query across a number of different EOs in the same way that a database view does with database tables.
<b>W</b>		
WAR	Web Archive	Whereas the <a href="#">EAR</a> is the complete application zipped up into a single file, the WAR file is only the web interface part of the application zipped up. So an <a href="#">EAR</a> file will/may contain a WAR file.
Web services		Simply put, a web service is a piece of functionality that exists out on the web. Imagine you could write a PL/SQL function and allow ANY program running on the web to access it (like a remote procedure call). You would put your package spec on the web and that would define what the function did and what results it returned. Anyone could then call that function, passing any required parameters, and would get a result. So, you could publish an "order lookup" web service that would accept an order number and return the status of the order that any 3rd party application could use. In web services, the actual implementation of the web service (analogous to the package or function body) is hidden and is implementation agnostic (so it could be written in Java or C or C++). The analogy of the package spec is implemented by the <a href="#">WSDL</a> and the messaging protocol for actually calling the web service is called <a href="#">SOAP</a> .
Web Start		Coming Soon
WSDL	Web Services Description Language	In the description of a <a href="#">web service</a> you were presented with the analogy of a remote call to a package spec. In web services the WSDL is in effect the package spec. It is an <a href="#">XML</a> file that defines the name of the web service, the parameters it takes and the types of values that it returns.
<b>X</b>		
XML	Extensible Markup Language	A tagged text file with a defined format that follows rules in the same that an HTML file is a text file with tags that define data. So comparing XML and HTML, XML was designed to describe data and what that data does, HTML was designed to focus on how data is displayed.
<b>Y</b>		
<b>Z</b>		