Introduction to Support for Java EE Technology in NetBeans IDE

NetBeans IDE has been developed in close cooperation with the Java EE and GlassFish teams to provide the tightest integration and easiest possible use of the Java EE specification. NetBeans IDE is the best way to quickly learn and become productive in Java EE programming.

This document provides an introduction to the major concepts of the Java EE specifications and how they relate to your hands-on programming. The following topics are covered:

- Annotations Instead of Deployment Descriptors
- Simplified EJB Software Development
- Use Dependency Injection to Access Resources
- Java Persistence API Model
- Web Services

You can find more information about developing and deploying Java EE applications in the Java EE 7 Tutorial and the Java EE 6 Tutorial.

Annotations Instead of Deployment Descriptors

The Java EE platform simplifies deployment by removing the need for deployment descriptors, except for the deployment descriptor required by the servlet specification, the `web.xml` file. Other deployment descriptors, such as `ejb-jar.xml` and entries related to web services in `web.xml`, are obsolete. J2EE 1.4 deployment descriptors were often complex and it was easy to make mistakes in filling them out. Instead, the Java EE platform makes use of "annotations". Annotations are Java modifiers, similar to `public` and `private`, that you specify in your code. For example, the EJB 3 specification, which is a subset of the Java EE specification, defines annotations for the bean type, interface type, resource references, transaction attributes, security, and more. A similar set of annotations is provided for web services by the JAX-WS 2.0 specification. Some annotations are used for generating artifacts. Other annotations are used for documenting your code. Still others provide enhanced services such as security or runtime-specific logic. In summary, the Java EE platform provides annotations for the following tasks, among others:

- Defining and using web services
- Developing EJB software applications
- Mapping Java technology classes to XML
- Mapping Java technology classes to databases
- Mapping methods to operations
- Specifying external dependencies
- Specifying deployment information, including security attributes

Annotations are marked with a `@` character. In the IDE, when you create a type that makes use of annotations in Java

EE, related placeholders are provided in the generated code. For example, when you use the IDE to create a stateless session bean, the following code is generated, which includes the `@Stateless()` annotation:

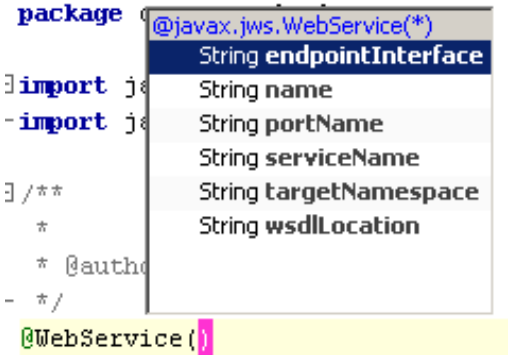
```
package mypackage;

import javax.ejb.*;

@Stateless()
public class HelloWorldSessionBean implements mypackage.HelloWorldSessionLocal {

}
```

Code completion provides access to annotation attributes specific to the item under the cursor. For example, when you press Ctrl-Space in the brackets of a `@WebService()` annotation, you see the following:



Each attribute has default values. Therefore, you do not need to specify any attributes unless you want to use a value other than the default value. In simple cases, the default value is sufficient, which means that you do not even need to provide attributes at all.

Simplified EJB Software Development

The new EJB 3.0 API makes software development easier by reducing and simplifying the amount of work required from the developer. In other words, fewer classes and less code. This is possible because more of the work is now performed by the container. Here are some of the features and benefits of the EJB 3:

- **Fewer required classes and interfaces.** You no longer need home and object interfaces for EJB components because the container is now responsible for exposing the necessary methods. You only need to supply a business interface. You can use annotations to declare your EJB components and the container will manage the transactions.
- **No more deployment descriptors.** You can use annotations directly in the class to tell the container about dependencies and configuration that you formerly defined in deployment descriptors. If there are no specific instructions, the container uses default rules to handle the most common situations.

- **Simple lookups.** The `EJBContext` enables you to lookup objects in the JNDI name space directly in the class.
- **Simplified object-relational mapping.** The new Java Persistence API makes object-relational mapping much simpler and transparent by allowing you to use annotations in POJOs to map Java objects to relational databases.

In the IDE, you can code enterprise beans just as you would code other Java classes, using code completion and editor hints to implement the correct methods and keep the classes in synch with their interfaces. You do not need to use special commands and dialog boxes to generate things like business methods or web service operations, although the commands are still available to help acquaint you with the syntax of Java EE code.

Use Dependency Injection to Access Resources

Dependency injection enables an object to use annotations to request external resources directly. This results in cleaner code because you no longer need to clutter your code with resource creation and lookup code. You can use resource injection in EJB components, web containers, and clients.

To request injection of a resource, a component uses the `@Resource` annotation or, in the case of some specialized resources, the `@EJB` and `@WebServiceRef` annotations. Resources that can be injected include:

- `SessionContext` object
- `DataSources` object
- `EntityManager` interface
- Other enterprise beans
- Web services
- Message queues and topics
- Connection factories for resource adapters

In the IDE, the Source Editor provides full code completion for resources injection annotations provided by the Java EE platform. In addition, the IDE automatically injects resources into your files when you run commands like Call EJB and Use Database.

Java Persistence API Model

The Java EE platform introduces the Java Persistence API, which was developed as part of JSR-220. The Java Persistence API can also be used outside of EJB components, for example, in web applications and application clients, and also outside the Java EE platform, in Java SE applications.

The Java Persistence API has the following key features:

- **Entities are POJOs.** Unlike EJB components that used container-managed persistence (CMP), entity objects using the new APIs are no longer components, and they no longer need to be in an EJB module.
- **Standardized object-relational mapping.** The new specification standardizes how object-relational mapping

is handled, freeing the developer from learning vendor-specific strategies. The Java Persistence API uses annotations to specify object-relational mapping information, but still support XML descriptors.

- **Named queries.** A named query is now a static query expressed in metadata. The query can be either a Java Persistence API query or a native query. This makes reusing queries very simple.
- **Simple packaging rules.** Because entity beans are simple Java technology classes, they can be packaged virtually anywhere in a Java EE application. For example, entity beans can be part of an EJB `JAR`, application-client `JAR`, `WEB-INF/lib`, `WEB-INF/classes`, or even part of a utility `JAR` in an enterprise application archive (EAR) file. With these simple packaging rules, you no longer have to make an EAR file to use entity beans from a web application or application client.
- **Detached entities.** Because entity beans are POJOs, they can be serialized and sent across the network to a different address space and used in a persistence-unaware environment. As a result, you no longer need to use data transfer objects (DTOs).
- **EntityManager API.** Application programmers now use a standard EntityManager API to perform `Create Read Update Delete` (CRUD) operations that involve entities.

The IDE provides tools to work with the new Java Persistence API. You can generate entity classes automatically from a database, or code entity classes by hand. The IDE also provides templates and graphic editors for creating and maintaining persistence units. See Getting Started with Java EE Applications for more information on using the Java Persistence API.

Web Services

In the Java EE platform, the use of annotations has greatly improved and simplified web services support. The following specifications contributed to this area: JSR 224, Java API for XML-Based Web Services (JAX-WS) 2.0; JSR 222, Java Architecture for XML Binding (JAXB) 2.0; and JSR 181, Web Services Metadata for the Java Platform.

JAX-WS 2.0

JAX-WS 2.0 is the new API for web services in the Java EE platform. As a successor to JAX-RPC 1.1, JAX-WS 2.0 retains the natural RPC programming model while improving on several fronts: data binding, protocol and transport independence, support for the `REST` style of web services, and ease of development.

A crucial difference from JAX-RPC 1.1 is that all data binding has now been delegated to JAXB 2.0. This allows JAX-WS-based web services to use 100 percent of XML Schema, which results in improved interoperability and ease of use. The two technologies are well integrated, so users no longer have to juggle two sets of tools. When starting from Java technology classes, JAXB 2.0 can generate XML Schema documents that are automatically embedded inside a Web Service Description Language (WSDL) document, saving users from performing this error-prone integration manually.

Out of the box, JAX-WS 2.0 supports the SOAP 1.1, SOAP 1.2, and XML/HTTP protocols. Protocol extensibility has been a goal from the very beginning, and JAX-WS 2.0 allows vendors to support additional protocols and encodings for better performance -- for example, the FAST Infoset -- or for specialized applications. Web services that use attachments to optimize the sending and receiving of large binary data can take advantage of the MTOM/XOP (short for message transmission optimization mechanism/XML-binary optimized packaging)

standard from W3C without any adverse effect on the programming model. (See this page for information on MTOM/XOP.) Before Java EE technology, defining a web service required long, unwieldy descriptors. Now it's as easy as placing the `@WebService` annotation on a Java technology class. All the public methods on the class are automatically published as web service operations, and all their arguments are mapped to XML Schema data types using JAXB 2.0.

Asynchronous Web Services

Because web service invocations take place over a network, such calls can take unpredictable lengths of time. Many clients, especially interactive ones such as JFC/Swing-based desktop applications, experience serious performance degradation from having to wait for a server's response. To avoid such performance degradation, JAX-WS 2.0 provides a new asynchronous client API. With this API, application programmers no longer have to create threads on their own. Instead, they can rely on the JAX-WS runtime to manage long-running remote invocations for them.

Asynchronous methods can be used in conjunction with any WSDL-generated interfaces as well as with the more dynamic `Dispatch` API. For your convenience, when importing a WSDL document, you can require asynchronous methods to be generated for any of the operations defined by the web service.

There are two usage models:

- In the polling model, you make a call. When you're ready, you request the results.
- In the callback model, you register a handler. As soon as the response arrives, you are notified.

Note that asynchronous invocation support is entirely implemented on the client side, so no changes are required to the target web service.

The IDE provides tools to work with JAX-WS. You can use templates in the New File wizard to generate JAX-WS artifacts. Asynchronous web services can be created by means of a Web Service Customization editor. The code completion functionality includes annotations that you can use in your web services.

Send Feedback on This Tutorial

Next Steps

For more information about using NetBeans IDE to develop Java EE applications, see the following resources:

- Getting Started with Java EE Applications
- Getting Started with JAX-WS Web Services
- Java EE & Java Web Learning Trail

To send comments and suggestions, get support, and keep informed on the latest developments on the NetBeans IDE Java EE development features, join the nbj2ee@netbeans.org mailing list.