Jakarta EE 11 and WildFly

Brian Stansberry | bstansbe@redhat.com

WildFly Technology Interest Survey

Today we're launching an anonymous survey to learn about community interest in using various technologies in WildFly.

https://forms.gle/heh23E6kvSmAJnR97

Your input is very much wanted!

Jakarta EE 11 Overview

- Expected date -- Q1 2025
  - Core Profile may be out shortly, with WildFly Preview 34 as a compatible implementation
- SE requirements:
  - Minimum: SE 17
  - TCK will allow implementations to certify on SE 21
  - WildFly will certify on both

What's New -- High Level

- EE 11 incorporates 32 specifications:
    - 1 new spec -- Jakarta Data
    - 18 specs have changes
    - 13 specs are unchanged from EE 10
    - 5 specs from EE 10 have been removed
- WildFly also provides 'preview' stability support for Jakarta MVC, which is not part of Jakarta EE

Jakarta EE TCK

- A huge in-progress effort is modernizing the EE 11 TCKs
  - Big technical debt: the TCK test framework dates to the 2000s, was specialized even then, and there are tens of thousands of tests
  - The TCK is moving to an Arquillian-base framework
  - WildFly community members Scott Marlow and Scott Stark are playing a big part
- Why should I care? The TCK became a barrier to innovation.
  - Needing to learn it was a barrier to contributions
  - Pool of engineers able to maintain it is shrinking

Breaking Changes

- SE 11 is no longer supported
- Minor breaking changes (e.g. deprecated API removal) in a handful of specs
- and ...

Security Manager

- EE 11 no longer supports running with a Security Manager
    - No longer required by the EE 11 spec
    - Many EE API and implementation libraries have removed code meant to deal with the SM
- WildFly Preview will fail to boot if you enable the SM
    - When we bring in EE 11 support, standard WildFly will do the same when running EE 11
- Java SE 24 is removing support for enabling the SM
- If you're using the SM we'd love to hear from you

Removed Specifications

- Four in the Web Services area
  - Jakarta XML Binding (fka JAXB), Jakarta XML Web Services (fka JAX-WS), Jakarta SOAP with Attachments, Jakarta Enterprise Web Services
  - **In WildFly we intend to continue supporting these specs.**
- Jakarta Managed Beans
  - *Long* ago superseded by CDI beans.
  - Is anyone using `@ManagedBean` on WildFly 27 or later?
  - In theory we could continue to support it, but I'd want to see demand.

Java Records

- Three EE specifications have incorporated support for `java.lang.Record` types
  - Expression Language
  - Persistence
    - A Java record type may now be annotated `@Embeddable` or used as an `@IdClass`.
  - Validation
    - Hibernate Validator already supported this though

Concurrency 3.1

- Injection of concurrency resources using @Inject instead of just @Resource
- Integration with the Java SE `Flow` API as contextual invocation points
- Scheduled @Asynchronous methods

Concurrency 3.1 Virtual Threads

- A new '`virtual`' attribute has been added to
  - `@ManagedExecutorDefinition`
  - `@ManagedScheduledExecutorDefinition`
  - `@ManagedThreadFactoryDefinition`
- If set to '`true`' the container may run tasks on a virtual thread, if supported
  - Won't be supported on SE < 21

Virtual Threads

- New in SE 21; aim is to improve performance executing blocking tasks by not tying up a native thread
    - e.g. a Jakarta Concurrency task that calls a database
- Beware! This is still a maturing technology
    - Performance can often be worse even if none of the following are issues
    - Code using basic things like `synchronized` can result in "thread pinning"
        - Perf impact or even deadlock
    - Libraries using `ThreadLocal` can have poor memory performance, particularly if they use them as the foundation of an object pool

Current WildFly Support

- WildFly Preview 34 exposes the Concurrency 3.1 API, but the backing implementation is still Concurrency 3.0
- This will likely be the same for WildFly Preview 35

Persistence 3.2

- The changelog for Persistence 3.2 is one of the bigger ones in recent years
- Gavin King wrote an excellent blog about Persistence 3.2; I'll just cover a few items
  - I already mentioned support for Java Records as embeddable classes

Programmatic Configuration

- An alternative to `persistence.xml`

```
var emf =
        new PersistenceConfiguration()
                .name("Bookshop")
                .nonJtaDataSource("java:global/jdbc/BookshopData")
                .managedClass(Book.class)
                .managedClass(Author.class)
                .property(PersistenceConfiguration.LOCK_TIMEOUT, 5000)
                .createEntityManagerFactory();
```

Programmatic Schema Export

- Create/drop/validate/truncate schema

```
emf.getSchemaManager().create(true); // create all the tables and stuff
```

```
emf.getSchemaManager().truncate(); // destroy all data before my next test
```

Type Safe Named Things

- Since 2.0, Jakarta Persistence has supported a *static metamodel* feature
  - Use an annotation processor to create metadata regarding managed entities and persistence objects; use the Metamodel API to access it at runtime
  - Original use case for this was the criteria query API
- Persistence 3.2 makes this much more useful
  - Metamodel now contains `static final` constants with the names of entity fields, named queries, named graphs, and named SQL result set mappings.

Example -- Easy type-safe query

Imagine a `Book` entity with

- a @NamedQuery named '`byTitle`'
- a generated static metamodel class `Book_`

You can easily execute your query and get a type-safe result:

```
List<Book> books = em.createQuery(Book_.byTitle).getResultList();
```

JPQL Enhancements

- Streamlined syntax for queries with a single entity
  - `from Book where title like :pattern`
- Spec support for Hibernate's `union`, `intersect`, `except`
  - `select name from Person union select name from Organization`
- Ad-hoc joins
  - `from Author a join Customer c on a.name = c.firstName||' '||c.lastName`
- New functions: `cast()` `left()` `right()` `replace()` `id()` `version()`
  - `select cast(left(fileName,2) as Integer) as chapter from Document`
- Improved sorting
  - `from Book order by lower(title) asc, publicationDate desc nulls first`

Current WildFly Support

- WildFly Preview 34 exposes the Persistence 3.2 API, but the backing implementation is still Persistence 3.1
    - Hibernate ORM 6.6
- This will likely be the same for WildFly 35

Jakarta Data 1.0

- Jakarta Data brings the 'repository' pattern to the Jakarta ecosystem
- An application developer defines a repository by providing an interface annotated with the `@Repository` annotation
  - Declares methods used for data retrieval and modification of one or more entity types
- The Jakarta Data provider provides the implementation of the repository interface

Repository Example -- CRUD

```
@Repository
public interface Library {
    @Insert
    void addToCollection(Book book);

    @Delete
    void removeFromCollection(Book book);

    @Insert
    void newAuthor(Author author);

    @Update
    void updateAuthor(Author author);

    ...
```

Repository Example -- Queries

```
@Repository
public interface Library {
    ...

    @Find
    Book book(String isbn);

    @Find
    List<Book> booksByTitle(@Pattern String title, Category category,
                            Order<Book> order, Limit limit);

    @Query("select b from Book b join b.authors a " +
           "where a.name = :authorName order by a.ssn, b.isbn")
    List<Book> booksBy(String authorName);
}
```

Build Time Repository Generation

- We use Hibernate Data Repositories as our Jakarta Data implementation
  - Part of Hibernate ORM 6.6
- Hibernate Data Repositories use an *annotation processor* to generate the repository interface impl during your app build
- You must add the processor to your pom and include a dependency on `org.hibernate.orm:hibernate-core`

Annotation Processor Config

```
<build>
     <plugins>
         <plugin>
             <groupId>org.apache.maven.plugins</groupId>
             <artifactId>maven-compiler-plugin</artifactId>
             <version>3.12.1</version> <!-- must be 3.12 or later -->
             <configuration>
                 <annotationProcessorPaths>
                     <path>
                         <groupId>org.hibernate.orm</groupId>
                         <artifactId>hibernate-jpamodelgen</artifactId>
                     </path>
                 </annotationProcessorPaths>
             </configuration>
         </plugin>
     </plugins>
 </build>
```

Current WildFly Support

- WildFly Preview 34 provides Jakarta Data 1.0 at 'preview' stability level
  - Use the '`jakarta-data`' Galleon layer or the CLI
    - `$ /extension=org.wildfly.extension.jakarta.data:add`
    - `$ /subsystem=jakarta-data:add`
- *Perhaps* we can bring it into standard WildFly 35, also at 'preview' stability
- We'll move to 'community' stability in WildFly 36

Specs with Smaller Changes

Specification Changelog	**Highlights**
Contexts and Dependency Injection 4.1	`@Priority` allowed on producer methods/fields
Expression Language 6.0	`Optional`, `Record`, resolve length for arrays
Faces 4.1	
Interceptors 4.2	interceptor binding access from `InvocationContext`
Pages 4.0	
RESTful Web Services 4.0	
Security 4.0	`@InMemoryIdentityStoreDefinition`
Servlet 6.1	`ByteBuffer` support in servlet IO streams
Validation 3.1	`Record` support
WebSocket 2.2	

WildFly Support for EE 11

- Currently limited to WildFly Preview
- WFP 34 exposes nearly all the EE 11 APIs to application code
    - Missing: Jakarta Authentication 3.1 and Jakarta Pages 4.0
- WFP provides all the EE 11 Core Profile backing impls
- But, a significant number of the Web Profile and Full Platform backing impls are still the EE 10 variants
    - Authentication, Authorization, Concurrency, Pages, Persistence, Security, Servlet, Websocket

Roadmap

- WildFly 35 (beta in Dec) will be much the same as 34
- We're aiming to have all the EE 11 implementations integrated in WildFly Preview for the 36 release (April '25)
- Whether we bring EE 11 to standard WildFly in 36 or 37 is still TBD.
  - We're not planning to feature-box standard WildFly for EE 11; we'll move to it when it's ready for a normal quarterly release

'Dual Support' -- EE 10 and EE 11

We're evaluating producing parallel feature packs for standard WildFly, one with EE 10 APIs and the other with EE 11.

User story:

> *"I'm an architect who emphasizes keeping up with WildFly releases in order to get CVE and general bug fixes, but I need a longer time to adapt to Jakarta EE changes."*

The EE 10 variant of standard WildFly would provide that "longer time".

'Dual Support' Caveats

We can only continue releases of the 'older' (i.e. EE 10) variant as long as all the components that are part of it have compatible releases that are acceptable.

- No CVEs. (Or perhaps none with a severity score greater than some TBD #)
- No other bugs with critical impact on WildFly.

We integrate hundreds of libraries, and many of those only maintain their 'main' branch. They might not produce bug-fix releases for their EE 10 variants.

Think of dual support as buying you an extra quarter or two or three to transition to EE 11.

Questions

After the talk

- If you have further questions, please feel to ask in Zulip!
  - Session-specific thread: https://bit.ly/3ASQmL3
  - General user Zulip channel: https://bit.ly/3UWam6r
- We'd love to get your input on the WildFly Technology Interest Survey!
  - https://forms.gle/heh23E6kvSmAJnR97