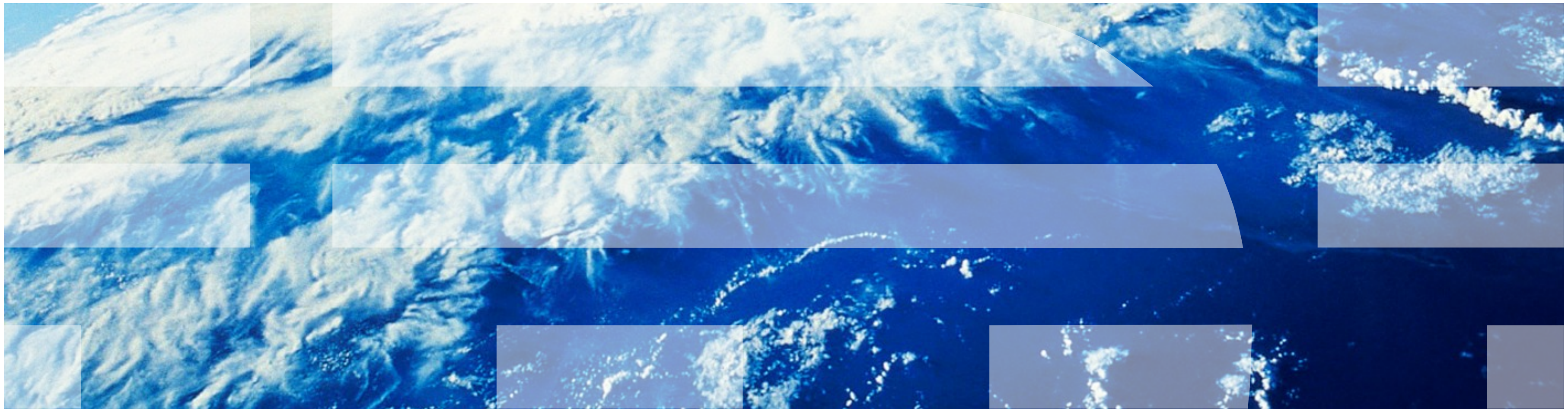
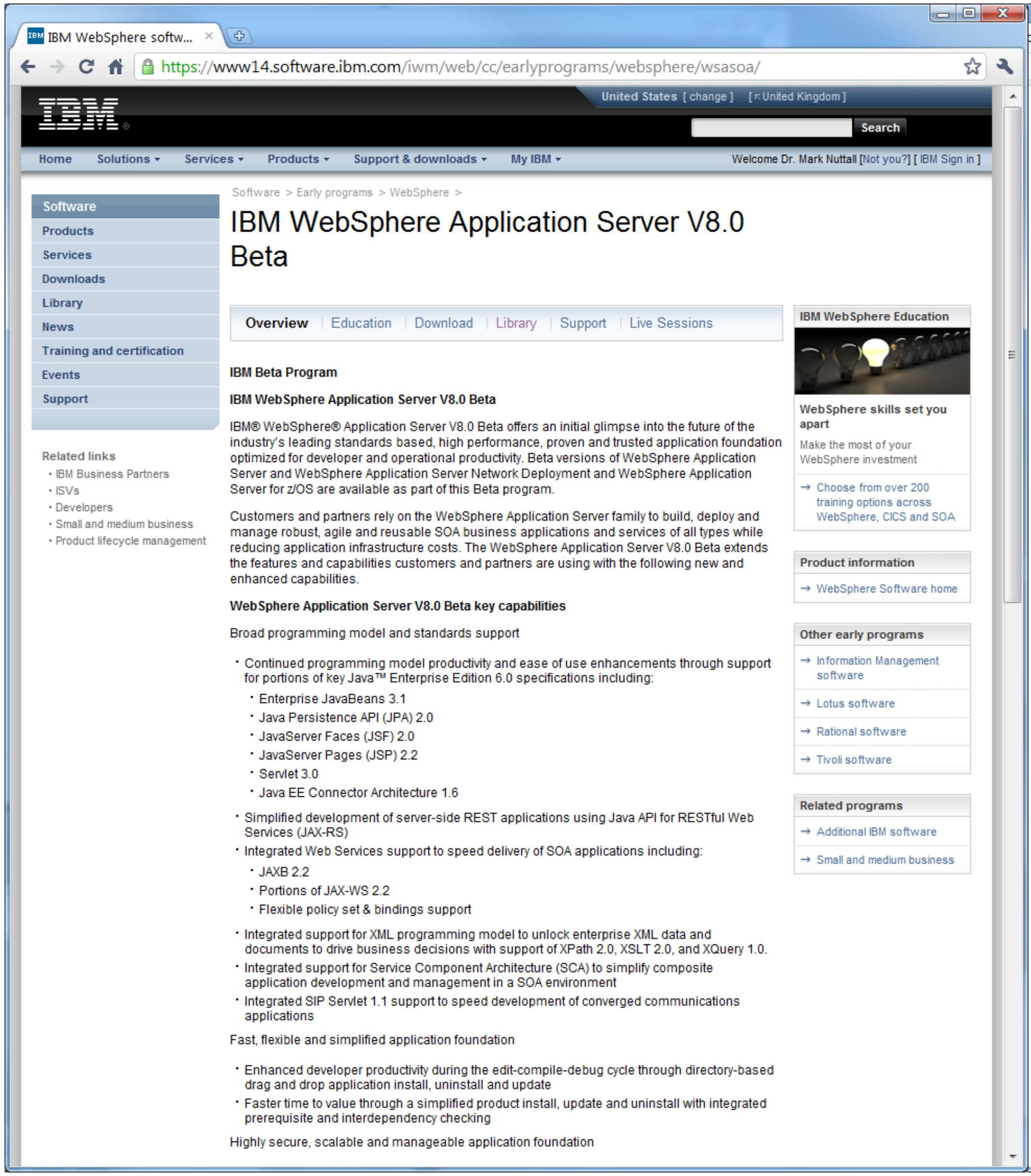
IBM WebSphere Application Server V8 Beta New Features

Mark Nuttall, mnuttall@uk.ibm.com



IBM WebSphere Application Server V8.0 Beta: New Features



IBM WebSphere softw...
https://www14.software.ibm.com/iwm/web/cc/earlyprograms/websphere/wsasoa/
United States [change] [United Kingdom]
Search
Home Solutions Services Products Support & downloads My IBM
Welcome Dr. Mark Nuttall [Not you?] [IBM Sign in]
Software
Products
Services
Downloads
Library
News
Training and certification
Events
Support
Related links
IBM Business Partners
ISVs
Developers
Small and medium business
Product lifecycle management
Software > Early programs > WebSphere >
Overview Education Download Library Support Live Sessions
IBM WebSphere Education
WebSphere skills set you apart
Make the most of your WebSphere investment
Choose from over 200 training options across WebSphere, CICS and SOA
Product information
WebSphere Software home
Other early programs
Information Management software
Lotus software
Rational software
Tivoli software
Related programs
Additional IBM software
Small and medium business

IBM WebSphere Application Server V8.0 Beta

IBM Beta Program

IBM WebSphere Application Server V8.0 Beta

IBM® WebSphere® Application Server V8.0 Beta offers an initial glimpse into the future of the industry's leading standards based, high performance, proven and trusted application foundation optimized for developer and operational productivity. Beta versions of WebSphere Application Server and WebSphere Application Server Network Deployment and WebSphere Application Server for z/OS are available as part of this Beta program.

Customers and partners rely on the WebSphere Application Server family to build, deploy and manage robust, agile and reusable SOA business applications and services of all types while reducing application infrastructure costs. The WebSphere Application Server V8.0 Beta extends the features and capabilities customers and partners are using with the following new and enhanced capabilities.

WebSphere Application Server V8.0 Beta key capabilities

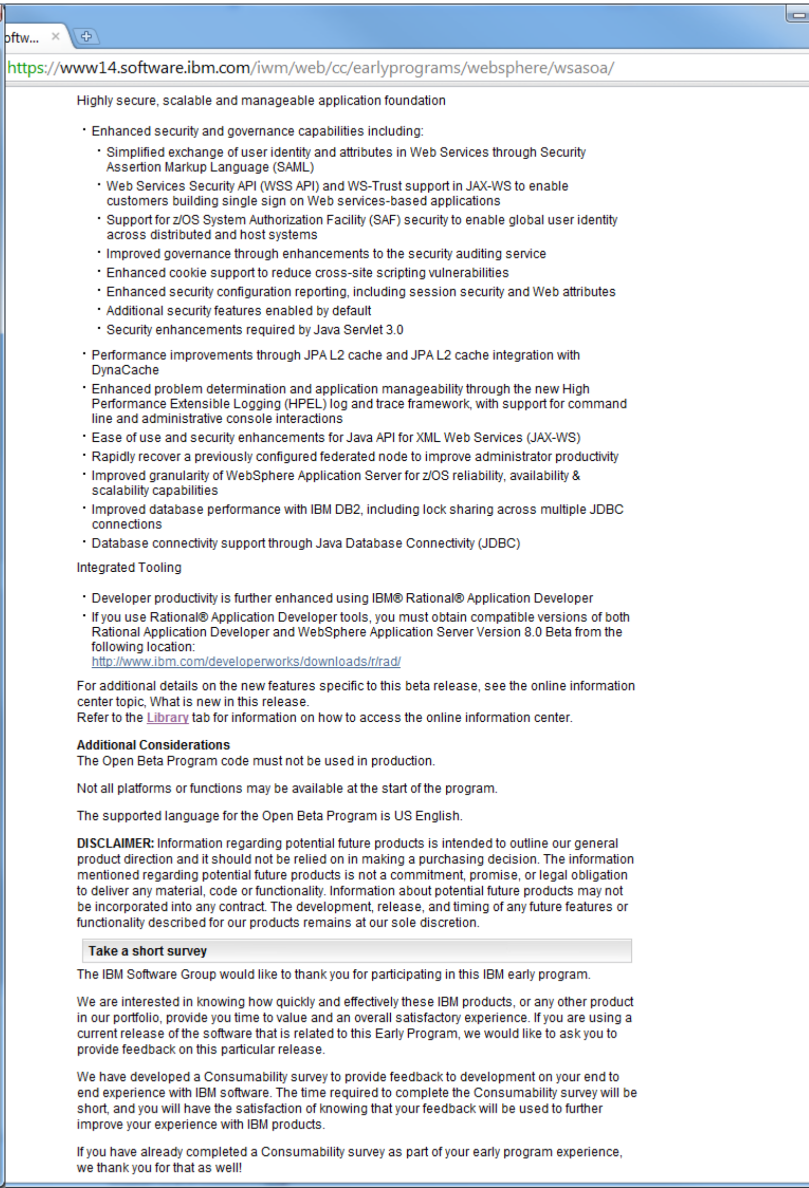
Broad programming model and standards support

- Continued programming model productivity and ease of use enhancements through support for portions of key Java™ Enterprise Edition 6.0 specifications including:
  - Enterprise JavaBeans 3.1
  - Java Persistence API (JPA) 2.0
  - JavaServer Faces (JSF) 2.0
  - JavaServer Pages (JSP) 2.2
  - Servlet 3.0
  - Java EE Connector Architecture 1.6
- Simplified development of server-side REST applications using Java API for RESTful Web Services (JAX-RS)
- Integrated Web Services support to speed delivery of SOA applications including:
  - JAXB 2.2
  - Portions of JAX-WS 2.2
  - Flexible policy set & bindings support
- Integrated support for XML programming model to unlock enterprise XML data and documents to drive business decisions with support of XPath 2.0, XSLT 2.0, and XQuery 1.0.
- Integrated support for Service Component Architecture (SCA) to simplify composite application development and management in a SOA environment
- Integrated SIP Servlet 1.1 support to speed development of converged communications applications

Fast, flexible and simplified application foundation

- Enhanced developer productivity during the edit-compile-debug cycle through directory-based drag and drop application install, uninstall and update
- Faster time to value through a simplified product install, update and uninstall with integrated prerequisite and interdependency checking

Highly secure, scalable and manageable application foundation



oftw...
https://www14.software.ibm.com/iwm/web/cc/earlyprograms/websphere/wsasoa/

Highly secure, scalable and manageable application foundation

- Enhanced security and governance capabilities including:
  - Simplified exchange of user identity and attributes in Web Services through Security Assertion Markup Language (SAML)
  - Web Services Security API (WSS API) and WS-Trust support in JAX-WS to enable customers building single sign on Web services-based applications
  - Support for z/OS System Authorization Facility (SAF) security to enable global user identity across distributed and host systems
  - Improved governance through enhancements to the security auditing service
  - Enhanced cookie support to reduce cross-site scripting vulnerabilities
  - Enhanced security configuration reporting, including session security and Web attributes
  - Additional security features enabled by default
  - Security enhancements required by Java Servlet 3.0
- Performance improvements through JPA L2 cache and JPA L2 cache integration with DynaCache
- Enhanced problem determination and application manageability through the new High Performance Extensible Logging (HPEL) log and trace framework, with support for command line and administrative console interactions
- Ease of use and security enhancements for Java API for XML Web Services (JAX-WS)
- Rapidly recover a previously configured federated node to improve administrator productivity
- Improved granularity of WebSphere Application Server for z/OS reliability, availability & scalability capabilities
- Improved database performance with IBM DB2, including lock sharing across multiple JDBC connections
- Database connectivity support through Java Database Connectivity (JDBC)

Integrated Tooling

- Developer productivity is further enhanced using IBM® Rational® Application Developer
- If you use Rational® Application Developer tools, you must obtain compatible versions of both Rational Application Developer and WebSphere Application Server Version 8.0 Beta from the following location: http://www.ibm.com/developerworks/downloads/r/rad/

For additional details on the new features specific to this beta release, see the online information center topic, What is new in this release.
Refer to the Library tab for information on how to access the online information center.

Additional Considerations
The Open Beta Program code must not be used in production.

Not all platforms or functions may be available at the start of the program.

The supported language for the Open Beta Program is US English.

DISCLAIMER: Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Take a short survey

The IBM Software Group would like to thank you for participating in this IBM early program.

We are interested in knowing how quickly and effectively these IBM products, or any other product in our portfolio, provide you time to value and an overall satisfactory experience. If you are using a current release of the software that is related to this Early Program, we would like to ask you to provide feedback on this particular release.

We have developed a Consumability survey to provide feedback to development on your end to end experience with IBM software. The time required to complete the Consumability survey will be short, and you will have the satisfaction of knowing that your feedback will be used to further improve your experience with IBM products.

If you have already completed a Consumability survey as part of your early program experience, we thank you for that as well!

There's a lot of new features in WAS V8 Beta compared to WAS V7. These include:

- Portions of Java EE 6
  - EJB 3.1, JPA 2.0, JSF 2.0, JSP 2.2, Servlet 3.0, JCA 1.6
- Java API for RESTful Web Services (JAX-RS)
- Web Services support
  - JAXB 2.2, portions of JAX-WS 2.2
- XML and other programming models
  - XPath 2.0, XSLT 2.0, XQuery 1.0
  - SIP Servlet 1.1
- Directory-based drag and drop install, update, uninstall
- IBM Installation Manager
- Many new security features, performance improvements, enhanced problem determination

What this talk will cover

- There’s far too much new technology in WAS V8 Beta to cover in an hour! This talk will focus on:
  - The WAS V8 Beta and WebSphere Customer Experience program
  - IBM Installation Manager
  - EJB 3.1
  - Servlet 3.0

WebSphere Version 8.0 Agile Alpha/Beta

- New Agile Alpha/Beta approach to:
  - Better leverage Agile development
  - Improve feedback
  - Reach a broader range of clients
  - Increase Development teams' involvement in the programs

- Beta available July 1, 2010
  - Windows, Linux, AIX , HP-UX and Solaris, z/OS and Linux for System z
  - Website download

    https://www14.software.ibm.com/iwm/web/cc/earlyprograms/websphere/wsasoa/index.shtml

  - Interactive Forum led by Development Teams

    https://www.ibm.com/developerworks/forums/forum.jspa?forumID=2180

WebSphere Customer Experience Program (CEP)

- Agile Alpha/Beta
  - Anyone can download and exercise the Alpha/Beta code
  - Anyone can participate in forum discussions with focused attention from IBM architects and developers

- Increased Interaction via Customer Experience Program (CEP)
  - Regular and frequent interactions offered during Feature Focus weeks
  - Milestone demos and feedback sessions to gather on-going client input
  - Consumability validation sessions with clients
  - Send a note to cep@us.ibm.com

IBM Installation Manager

- IBM Installation Manager based install of
  - WebSphere Application Server
  - IBM HTTP Server
  - Web Server Plugins
  - WebSphere Configuration Tools
- Multiple platforms supported
  - Windows, Linux, AIX, Solaris
- IBM Installation Manager based install of WebSphere Application Server of System Z machines
- Uses IBM Installation Manager (IM)
  - IM automatically downloads and installs exactly (and only) what is needed
  - Install via Local and/or Remote Repositories
  - User Interface or Silent Mode
- Dynamic download and install of exact desired product version
  - Product (WAS ND, etc.) + Feature Packs (SCA, etc.) + Fix Packs + iFixes
  - No need to statically pre-build and distribute an Install Factory package

EJB 3 support was introduced in WAS V7.

- EJB 3 simplified the EJB 2.5 programming model.
- Extensive use of annotation, making business logic unit testable outside an application server.
- POJO programming model for session and entity beans.
- Text editor now adequate: IDE still helpful, but not a prerequisite.
- Resource injection made EJB home lookup and usage optional.
- Deployment descriptors became optional due to ‘configuration by exception’ approach.

EJB 3.1: No-interface local view

- Session beans no longer *have* to implement an interface.
- All public, non-final methods can be looked up and used by local clients.
- Session bean with No-Interface view because no declared interfaces

```
session bean with No-Interface view because no declared interfaces
@Stateless
public class CartBean
```

- Session bean with No-Interface view via @LocalBean

```
session bean with No-Interface view via @LocalBean
@Stateless
@LocalBean
@Remote( Cart.class )
public class CartBean implements Cart
```

Asynchronous session beans

- Standardised in EJB 3.1
- Allows EJB methods to run asynchronously
- Improves performance and increases scalability
- Two modes:
  - Fire and forget
  - Fire and return results.

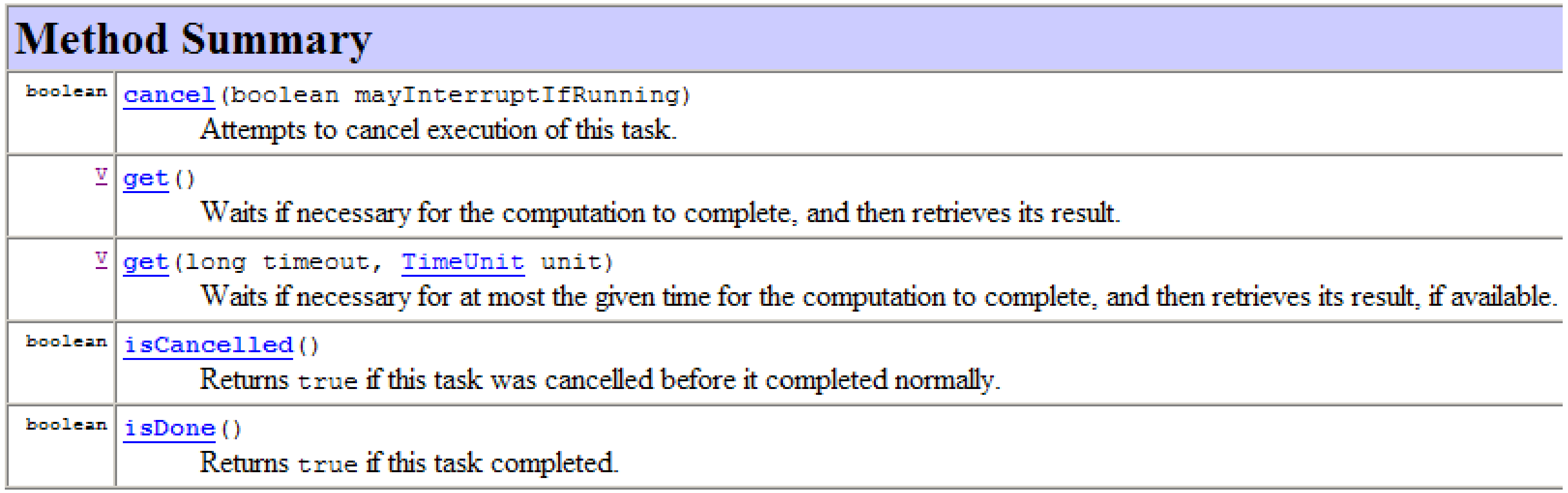
Asynchronous session bean invocations
Fire and forget

```
public interface Email
  {
   public void sendEmail (String name, String message);
  }

  @Stateless
  @Local(Email.class)
  public class CalculatorBean {
  @Asynchronous
public void sendEmail (String name, String message);
  {
    // ... Send email.;
}
```

Asynchronous session bean invocations
Fire and return results.

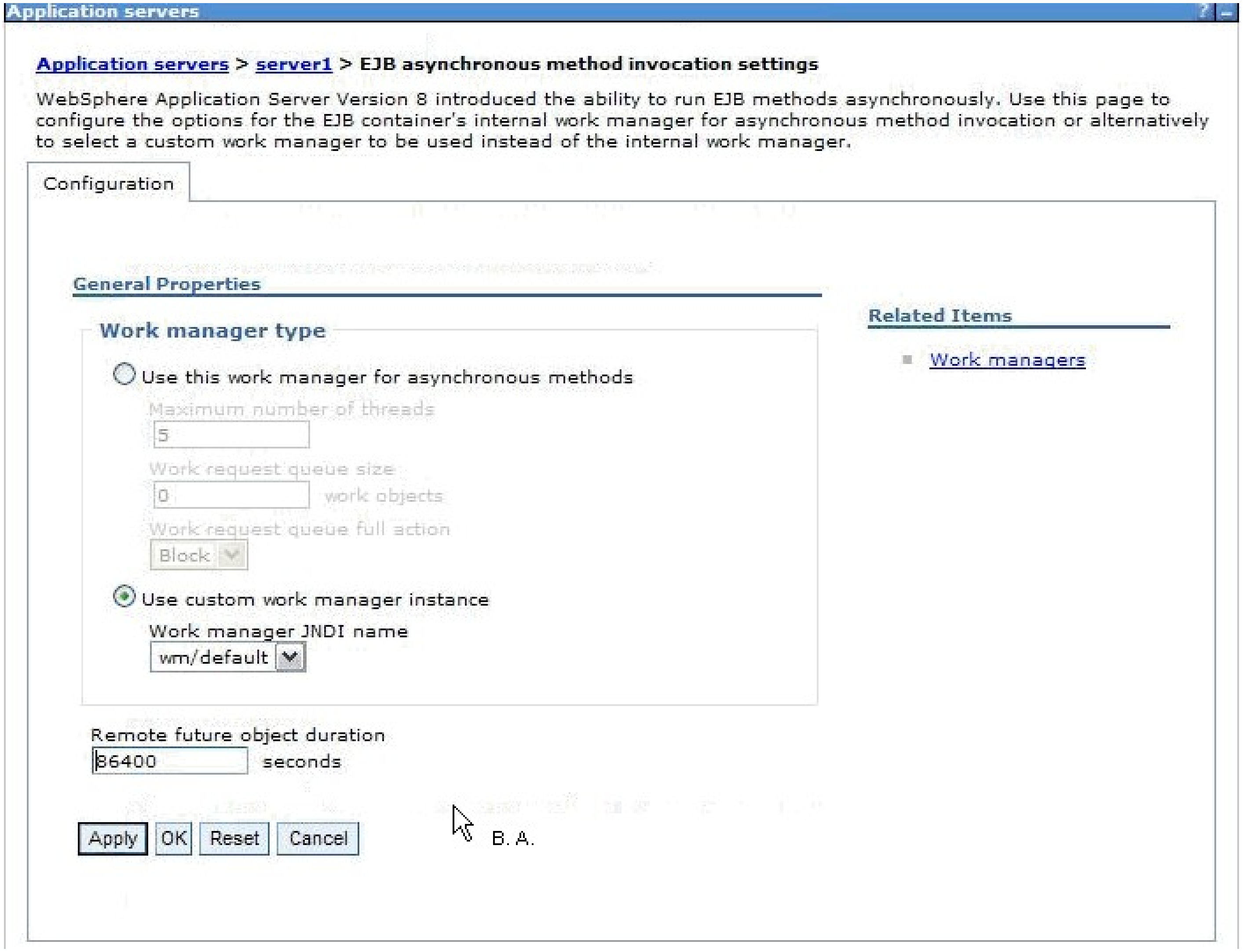
```
Import javax.ejb.AsyncResult;
….
 public interface Calculator
  {
      public Future<Integer> performCalculation(int a, int b);
  }

  @Stateless
  @Local(Calculator.class)
  public class CalculatorBean {
 @Asynchronous
  public Future<Integer> performCalculation(int a, int b)
  {
    // ... do calculation
    Integer result = ...;
    return new AsynchResult(result);
  }
```

Asynchronous session bean invocations
Future Object



Method Summary	
`boolean`	`cancel(boolean mayInterruptIfRunning)` Attempts to cancel execution of this task.
`V`	`get()` Waits if necessary for the computation to complete, and then retrieves its result.
`V`	`get(long timeout, TimeUnit unit)` Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.
`boolean`	`isCancelled()` Returns `true` if this task was cancelled before it completed normally.
`boolean`	`isDone()` Returns `true` if this task completed.

Asynchronous session bean invocations Configuration



Application servers

Application servers > server1 > EJB asynchronous method invocation settings

WebSphere Application Server Version 8 introduced the ability to run EJB methods asynchronously. Use this page to configure the options for the EJB container's internal work manager for asynchronous method invocation or alternatively to select a custom work manager to be used instead of the internal work manager.

Configuration

General Properties

Work manager type

( ) Use this work manager for asynchronous methods

Maximum number of threads: 5

Work request queue size: 0 work objects

Work request queue full action: Block

(●) Use custom work manager instance

Work manager JNDI name: wm/default

Remote future object duration: 86400 seconds

Singleton session beans

- New session bean type.
- Guaranteed single instance per JVM.
- Supports eager initialization during application startup.
- Allows for sharing of data across all the apps in the Server.
  - avoid extra database trips.
- Concurrency management.
  - @ConcurrencyManagement(BEAN)
    e.g. public synchronized setProductInfo….
  - @ConcurrencyManagement(CONTAINER)
    - @Lock(LockType.*READ*)
    - @Lock(LockType.*WRITE*)

Singleton session beans

```
@Singleton
@LocalBean
@Startup
public class InventoryBean
{
    @Lock(LockType.READ)
     public int[] getInventory()  {…}

     @Lock(LockType.WRITE)
      public void setInventory()  {…}
}
```

Timers, Duration Based

```
@Stateless
public class Mybean{

@Resource
 private Session sess;

@Resource
private TimerService ts;

public void doSomeTimerWork(String message, Long duration)
{
    ts.createTimer(duration, message);
}

@Timeout
private void doSomeTimeOutWork(timer timer)

  .. Stuff
}
```

Automatic timer creation

- Created automatically.
- Can be created using annotation or xml
- Created/started when app first started.
- Removed when application is uninstalled.

Automatic timer creation

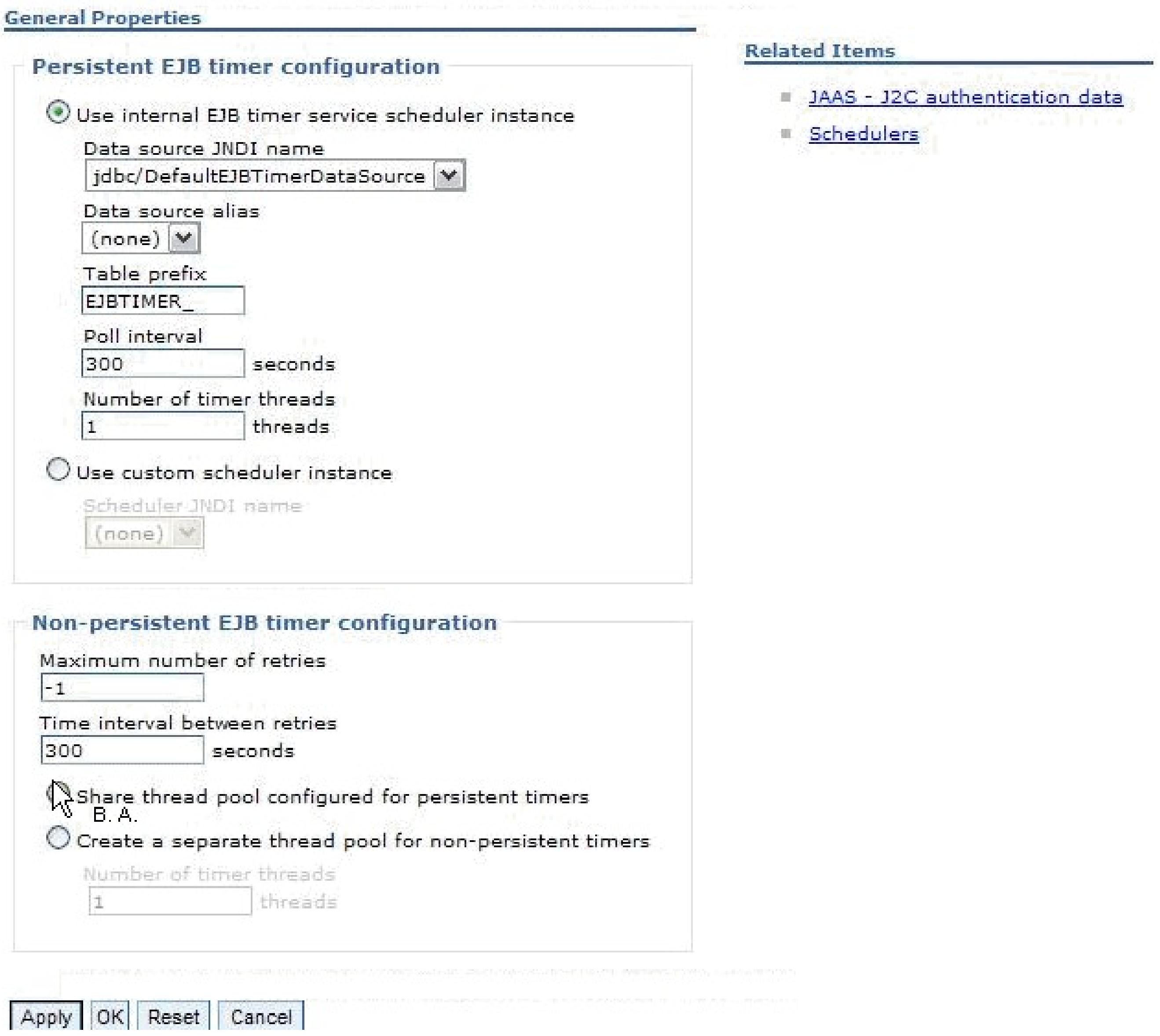
```
// Generate account statements at 1 a.m. on the 1st of every
month
@Schedule (hour=”1”, dayOfMonth=”1”,
info=”AccountStatementTimer”)
public void generateMonthlyAccountStatements(Timer t) {
String timerInfo = t.getInfo();
```

Non-persistent Timers

- Ability to declare non persistent timers.
- Applies to automatically and programmatically created timers

```
@Singleton
public class CacheBean {
Cache cache;
// Setup an automatic timer to refresh
// the Singleton instance cache every 10 minutes
@Schedule(minute=”*/10”, hour=”*”, persistent=false)
public void refresh()
{
   // ... Code to refresh the cache.
}
}
```

Timers configuration



General Properties

Persistent EJB timer configuration

[✓ Use internal EJB timer service scheduler instance]

Data source JNDI name: jdbc/DefaultEJBTimerDataSource

Data source alias: (none)

Table prefix: EJBTIMER_

Poll interval: 300 seconds

Number of timer threads: 1 threads

Use custom scheduler instance

Scheduler JNDI name: (none)

Non-persistent EJB timer configuration

Maximum number of retries: -1

Time interval between retries: 300 seconds

Share thread pool configured for persistent timers

B. A.

Create a separate thread pool for non-persistent timers

Number of timer threads: 1 threads

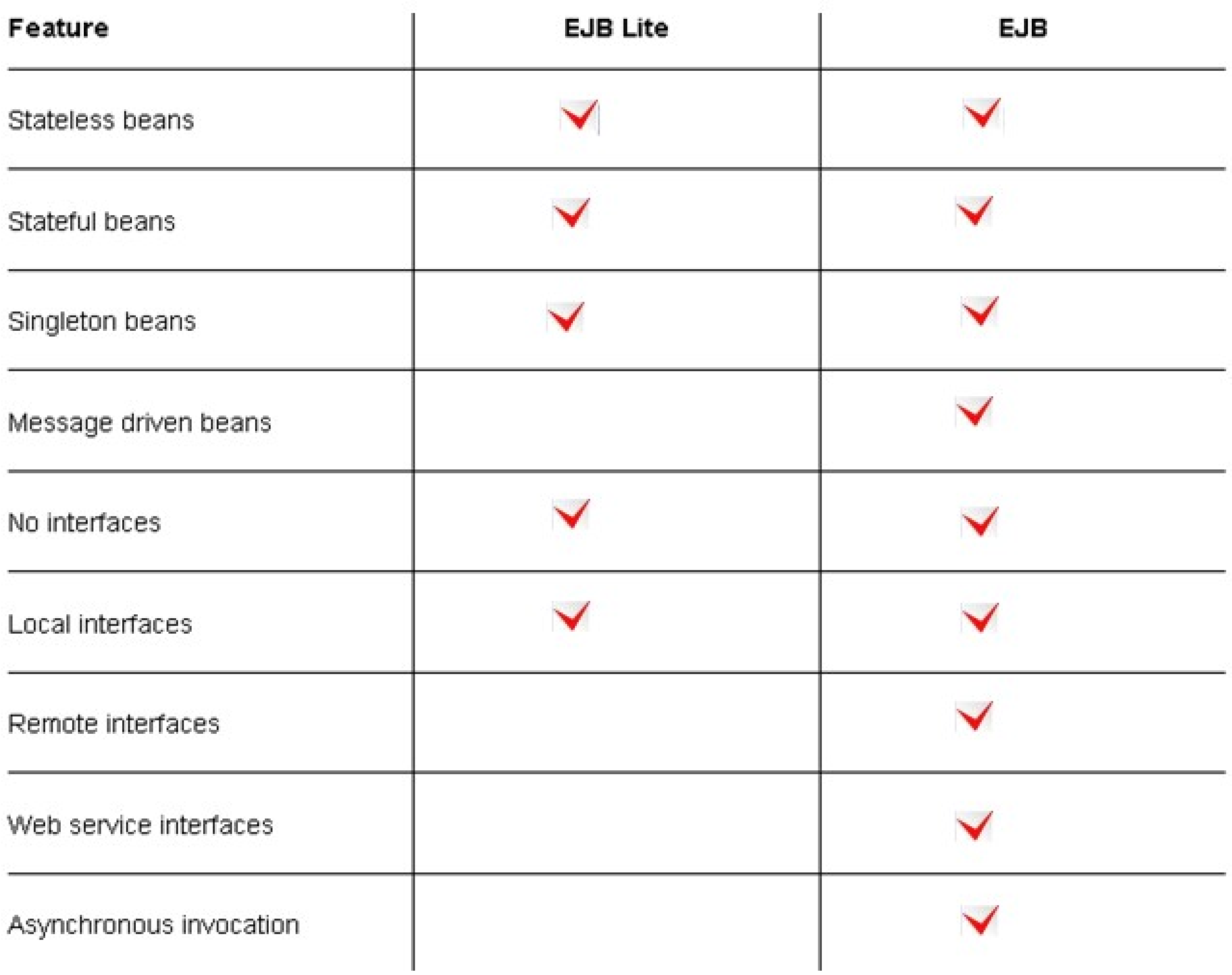
Related Items

- JAAS - J2C authentication data
- Schedulers

Embeddable EJBContainer

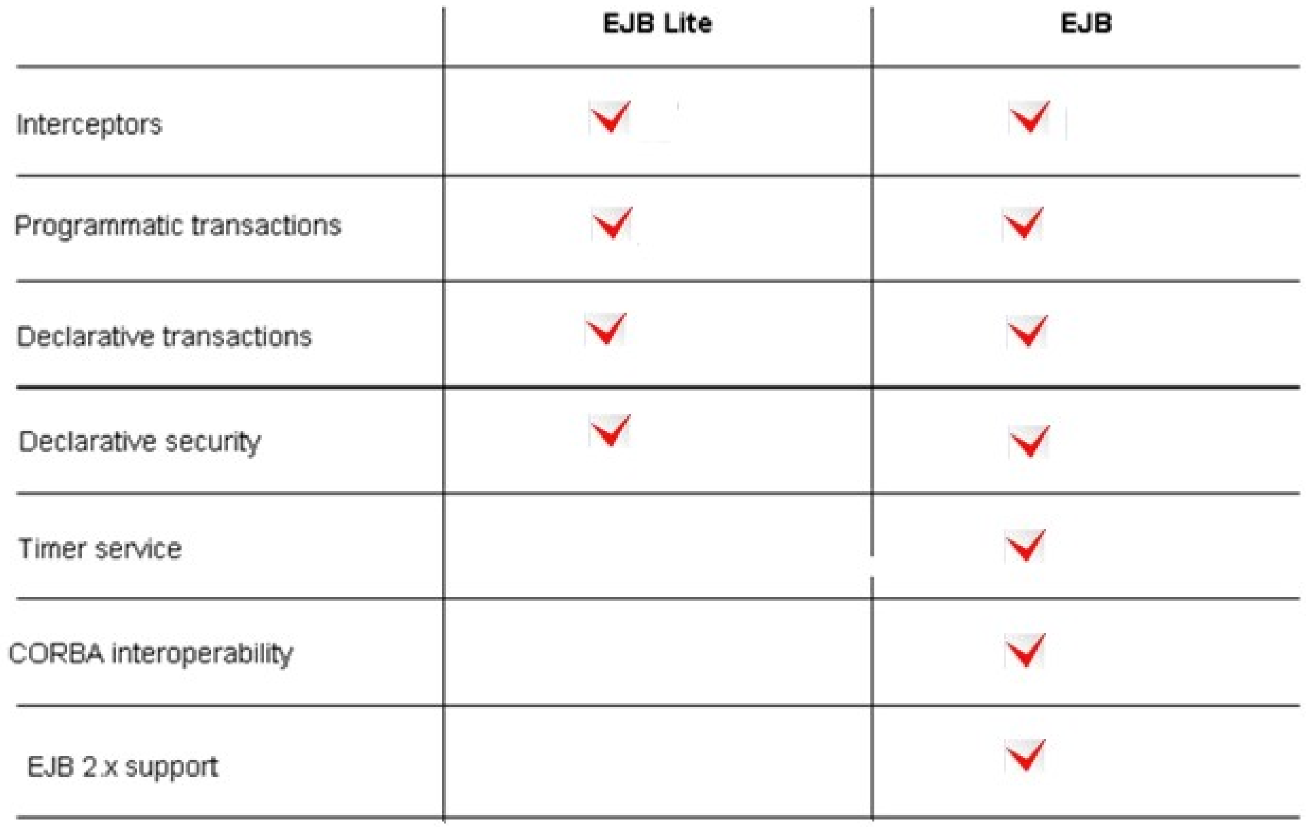
- Targeted for developers.
- Allow for easy way to unit test EJB business logic.
- Only need J2SE.

EJB3.1 (JSR 318)
Embeddable EJBContainer



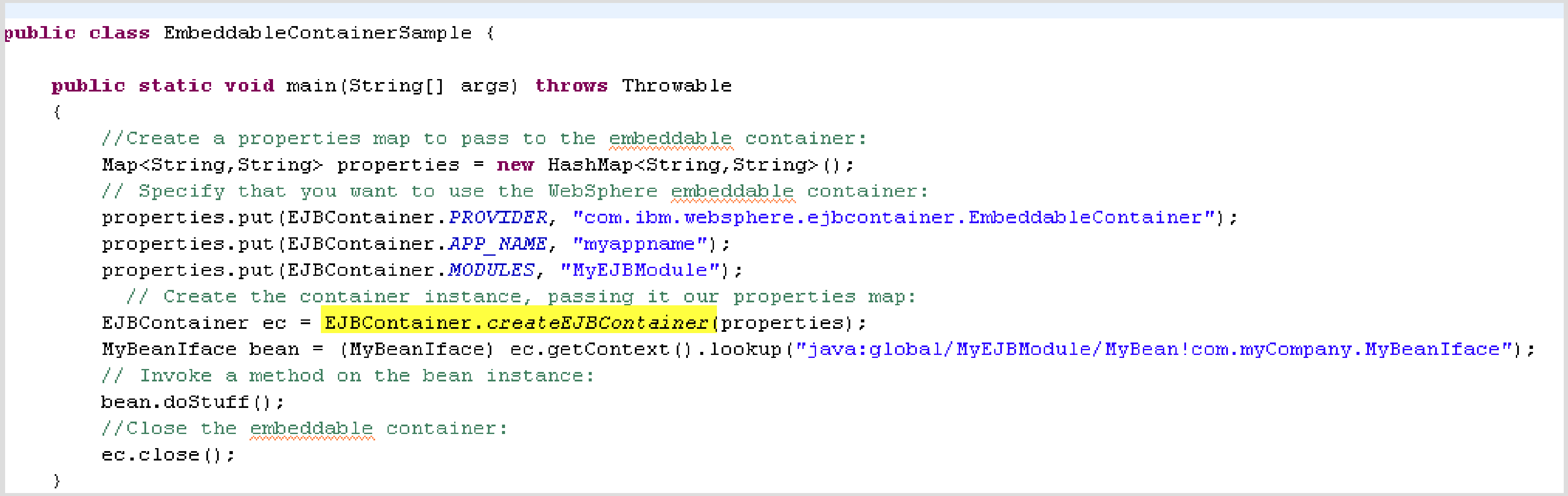
Feature	EJB Lite	EJB
Stateless beans	✓	✓
Stateful beans	✓	✓
Singleton beans	✓	✓
Message driven beans		✓
No interfaces	✓	✓
Local interfaces	✓	✓
Remote interfaces		✓
Web service interfaces		✓
Asynchronous invocation		✓

EJB3.1 (JSR 318)
Embeddable EJBContainer



	EJB Lite	EJB
Interceptors	✓	✓
Programmatic transactions	✓	✓
Declarative transactions	✓	✓
Declarative security	✓	✓
Timer service		✓
CORBA interoperability		✓
EJB 2.x support		✓

Embeddable EJBContainer



```
public class EmbeddableContainerSample {

    public static void main(String[] args) throws Throwable
    {
        //Create a properties map to pass to the embeddable container:
        Map<String,String> properties = new HashMap<String,String>();
        // Specify that you want to use the WebSphere embeddable container:
        properties.put(EJBContainer.PROVIDER, "com.ibm.websphere.ejbcontainer.EmbeddableContainer");
        properties.put(EJBContainer.APP_NAME, "myappname");
        properties.put(EJBContainer.MODULES, "MyEJBModule");
          // Create the container instance, passing it our properties map:
        EJBContainer ec = EJBContainer.createEJBContainer(properties);
        MyBeanIface bean = (MyBeanIface) ec.getContext().lookup("java:global/MyEJBModule/MyBean!com.myCompany.MyBeanIface");
        // Invoke a method on the bean instance:
        bean.doStuff();
        //Close the embeddable container:
        ec.close();
    }
```

Embeddable EJBContainer
Sample properties file

```
DataSource.ds1.name=env/jdbc/ds1
DataSource.ds1.className=org.apache.derby.jdbc.EmbeddedConnectionPoolDataSource
DataSource.ds1.transactional=true
DataSource.ds1.createDatabase=create
DataSource.ds1.databaseName=jtest1
DataSource.ds1.user=dbuser1
DataSource.ds1.password=dbpwd1
DataSource.ds1.maxPoolSize=5

DataSource.ds2.name=env/jdbc/ds2
DataSource.ds2.className=org.apache.derby.jdbc.EmbeddedXADataSource
DataSource.ds2.connectionSharing=MatchOriginalRequest
DataSource.ds2.createDatabase=create
DataSource.ds2.databaseName=jtest2
DataSource.ds2.user=dbuser2
DataSource.ds2.password=dbpwd2
DataSource.ds2.maxPoolSize=10
DataSource.ds2.minPoolSize=1
```

Recap : EJB 3.1 content in WAS V8 Beta

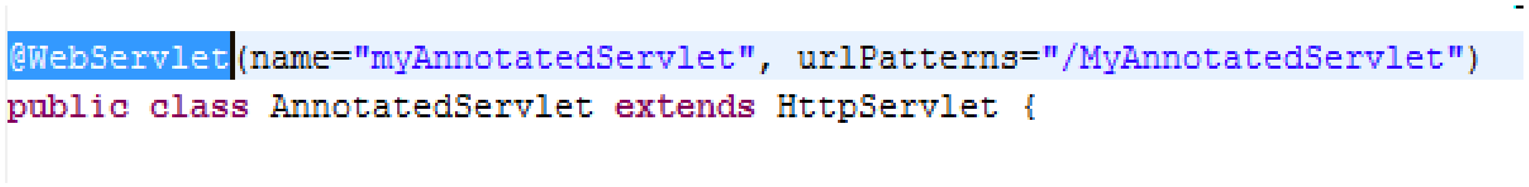
- Local non-interface view
- Asynch Method invocation
- Singleton Session Beans
- Calendar based timer expression
- Non-persistent EJB timers
- Automatically created EJB timers
- Embeddable Container

Servlet 3.0 features in the V8 beta

- Servlet 3.0 configuration options
- Annotations
- Web fragments
- Programmatic configuration
- Asynchronous processing support
- File upload/multipart form support

Servlet 3.0 Annotations

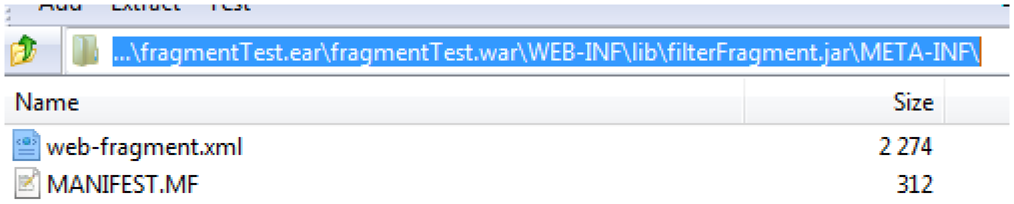
- @WebServlet, @WebFilter, @WebListener annotations can replace web.xml configuration
- Promotes developer productivity
- Example:



```
@WebServlet(name="myAnnotatedServlet", urlPatterns="/MyAnnotatedServlet")
public class AnnotatedServlet extends HttpServlet {
```

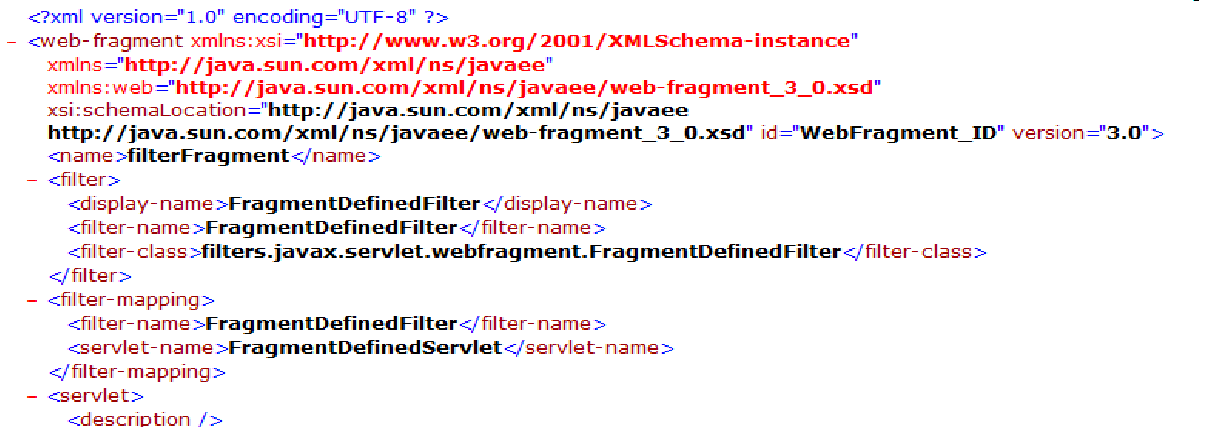
Web Fragments

- Configuration information can be embedded in WEB-INF/lib jars using Web fragments
- absolute-ordering allows you to order and exclude jars from scanning for fragments and annotations which can speed up startup time
- Example:



...\fragmentTest.ear\fragmentTest.war\WEB-INF\lib\filterFragment.jar\META-INF\

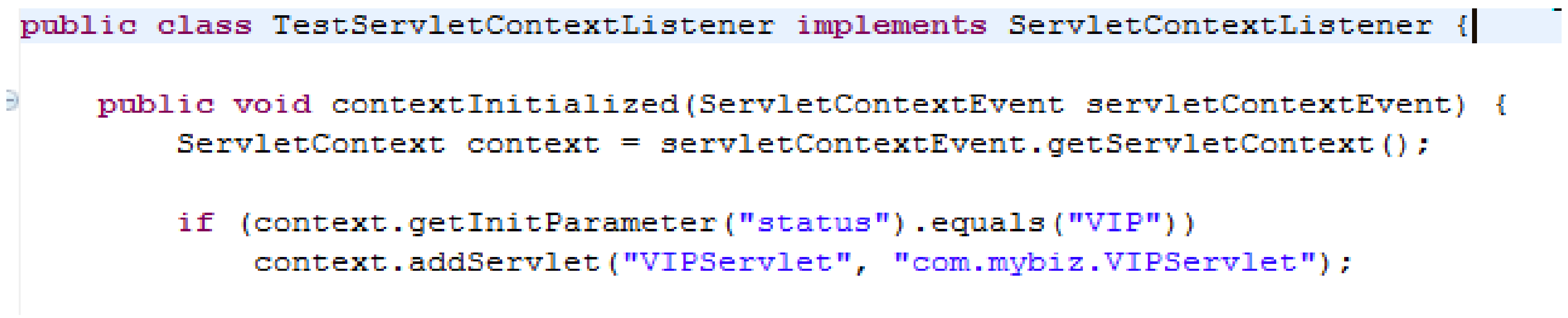
Name	Size
web-fragment.xml	2 274
MANIFEST.MF	312



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <web-fragment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:web="http://java.sun.com/xml/ns/javaee/web-fragment_3_0.xsd"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-fragment_3_0.xsd" id="WebFragment_ID" version="3.0">
    <name>filterFragment</name>
  - <filter>
      <display-name>FragmentDefinedFilter</display-name>
      <filter-name>FragmentDefinedFilter</filter-name>
      <filter-class>filters.javax.servlet.webfragment.FragmentDefinedFilter</filter-class>
    </filter>
  - <filter-mapping>
      <filter-name>FragmentDefinedFilter</filter-name>
      <servlet-name>FragmentDefinedServlet</servlet-name>
    </filter-mapping>
  - <servlet>
      <description />
```

Programmatic Configuration

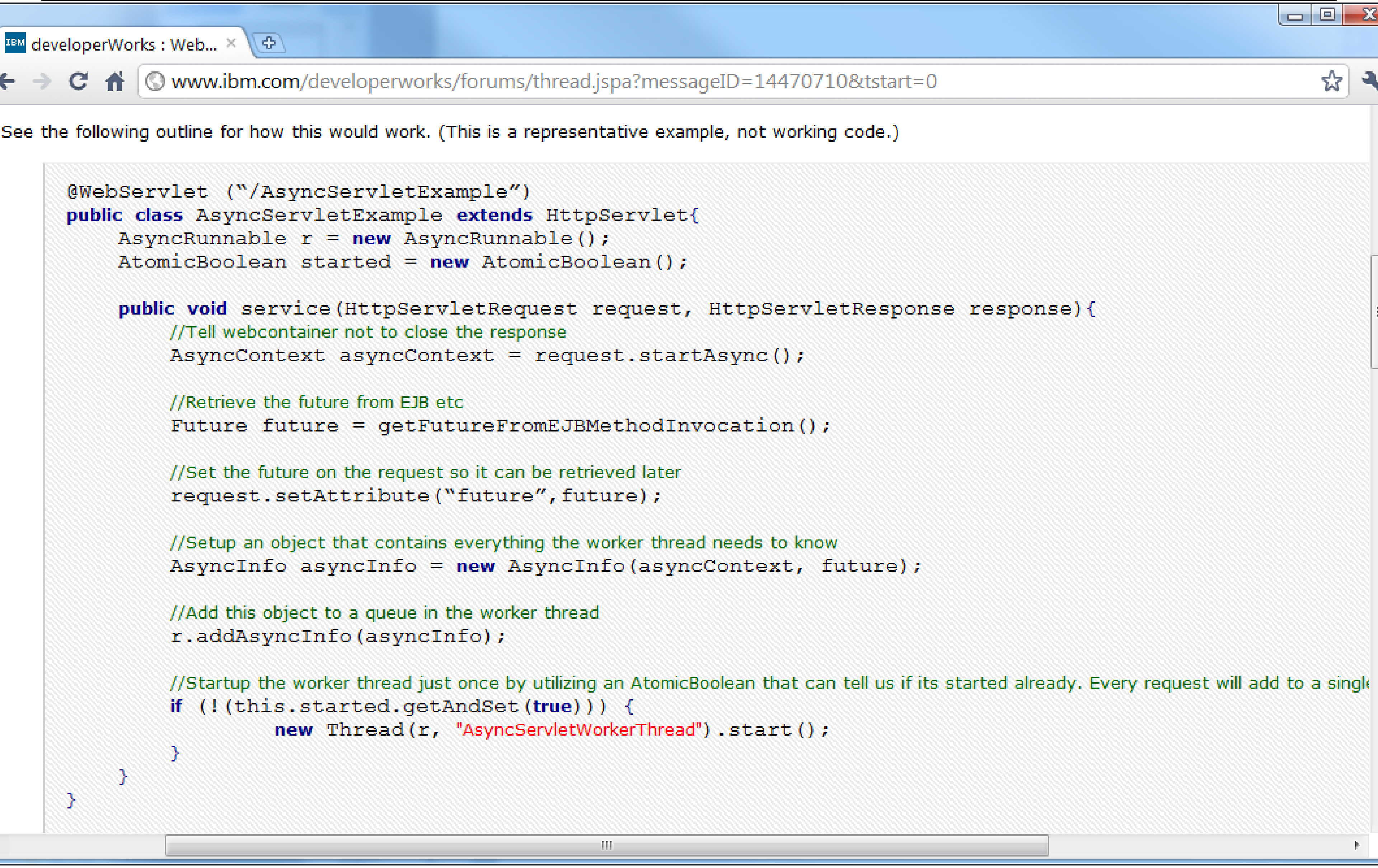
- Programmatic methods such as addServlet dynamically configure at Web app initialization
- Allows one to customize the application. Good for frameworks such as JSF.
- Example:



```
public class TestServletContextListener implements ServletContextListener {

    public void contextInitialized(ServletContextEvent servletContextEvent) {
        ServletContext context = servletContextEvent.getServletContext();

        if (context.getInitParameter("status").equals("VIP"))
            context.addServlet("VIPServlet", "com.mybiz.VIPServlet");
```

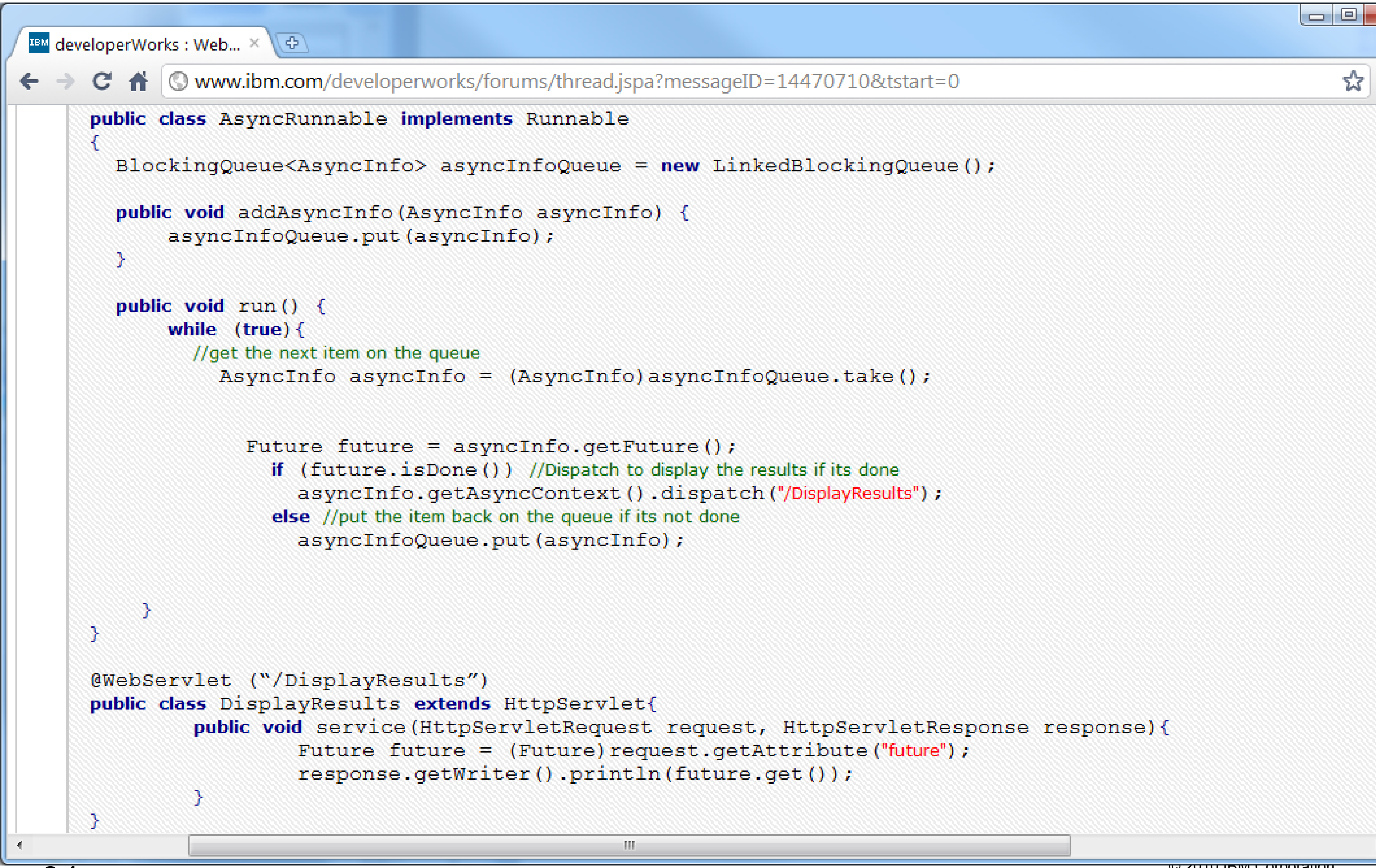
Asynchronous servlets

- Supports a suspend/resume paradigm that allows you to detach a request/response from normal thread lifecycle
- Good for server push operations
- Improves scalability
- Uses and applications:
  - Asynchronous EJB method invocation
  - Accessing RESTful Web services
  - Chat
  - Quality of Service
- Example on DeveloperWorks forum


developerWorks : Web...
www.ibm.com/developerworks/forums/thread.jspa?messageID=14470710&tstart=0

See the following outline for how this would work. (This is a representative example, not working code.)

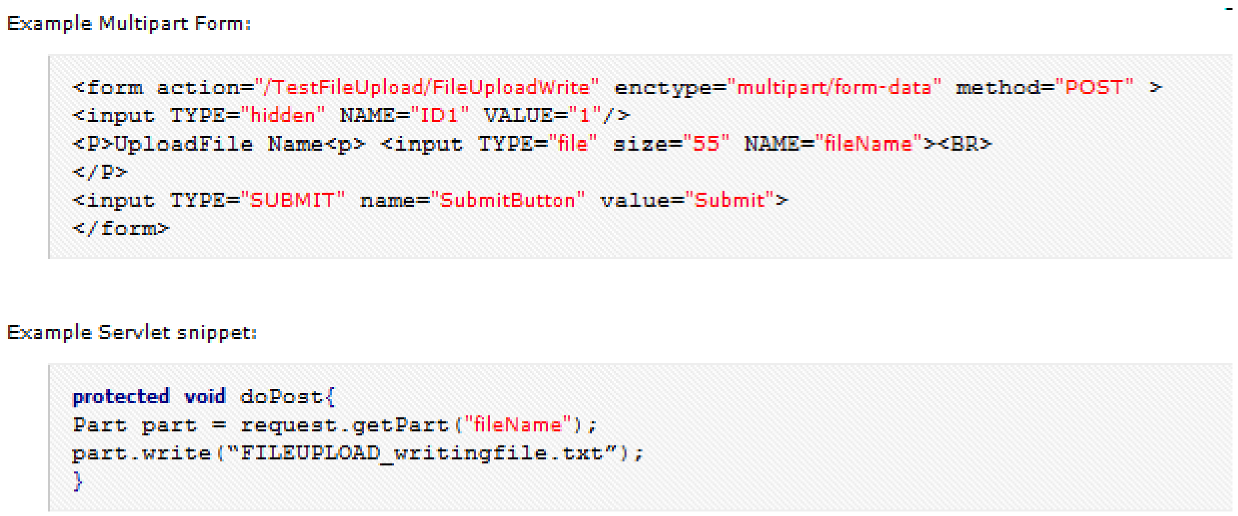
```
@WebServlet ("/AsyncServletExample")
public class AsyncServletExample extends HttpServlet{
    AsyncRunnable r = new AsyncRunnable();
    AtomicBoolean started = new AtomicBoolean();

    public void service(HttpServletRequest request, HttpServletResponse response){
        //Tell webcontainer not to close the response
        AsyncContext asyncContext = request.startAsync();

        //Retrieve the future from EJB etc
        Future future = getFutureFromEJBMethodInvocation();

        //Set the future on the request so it can be retrieved later
        request.setAttribute("future",future);

        //Setup an object that contains everything the worker thread needs to know
        AsyncInfo asyncInfo = new AsyncInfo(asyncContext, future);

        //Add this object to a queue in the worker thread
        r.addAsyncInfo(asyncInfo);

        //Startup the worker thread just once by utilizing an AtomicBoolean that can tell us if its started already. Every request will add to a singl
        if (!(this.started.getAndSet(true))) {
                new Thread(r, "AsyncServletWorkerThread").start();
        }
    }
}
```


developerWorks : Web...
www.ibm.com/developerworks/forums/thread.jspa?messageID=14470710&tstart=0

```
public class AsyncRunnable implements Runnable
{
  BlockingQueue<AsyncInfo> asyncInfoQueue = new LinkedBlockingQueue();

  public void addAsyncInfo(AsyncInfo asyncInfo) {
      asyncInfoQueue.put(asyncInfo);
  }

  public void run() {
      while (true){
        //get the next item on the queue
          AsyncInfo asyncInfo = (AsyncInfo)asyncInfoQueue.take();


            Future future = asyncInfo.getFuture();
              if (future.isDone()) //Dispatch to display the results if its done
                asyncInfo.getAsyncContext().dispatch("/DisplayResults");
              else //put the item back on the queue if its not done
                asyncInfoQueue.put(asyncInfo);



    }
}

@WebServlet ("/DisplayResults")
public class DisplayResults extends HttpServlet{
        public void service(HttpServletRequest request, HttpServletResponse response){
                Future future = (Future)request.getAttribute("future");
                response.getWriter().println(future.get());
        }
}
```

File upload/multipart support

- Supports retrieval of multipart/form-data
- Automatically writes files to disk based on file size threshold
- Write to disk on demand with Part.write()
- Example:



Example Multipart Form:

```
<form action="/TestFileUpload/FileUploadWrite" enctype="multipart/form-data" method="POST" >
<input TYPE="hidden" NAME="ID1" VALUE="1"/>
<P>UploadFile Name<p> <input TYPE="file" size="55" NAME="fileName"><BR>
</P>
<input TYPE="SUBMIT" name="SubmitButton" value="Submit">
</form>
```

Example Servlet snippet:

```
protected void doPost{
Part part = request.getPart("fileName");
part.write("FILEUPLOAD_writingfile.txt");
}
```

Summary

- There's lots of good new things in the WAS V8 beta. This talk covered only a few of them:
  - The WAS V8 Beta and WebSphere Customer Experience program
  - IBM Installation Manager
  - EJB 3.1
  - Servlet 3.0

Thank You

Questions?