



Mindtree

Welcome to possible

Enterprise Java Bean Session Beans

Objectives

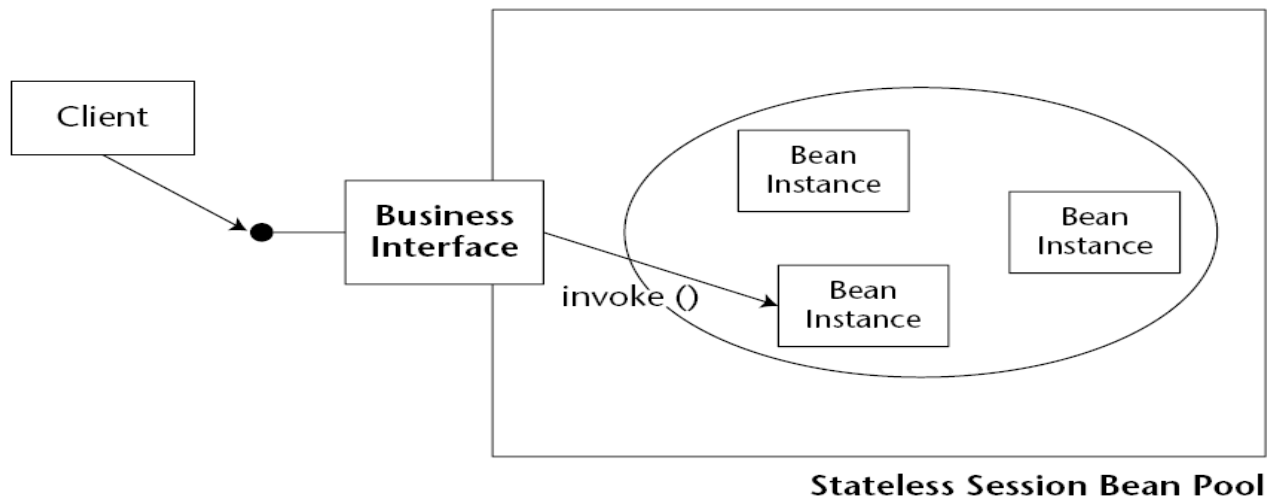
- Understand stateless and stateful session beans
- Define local and remote invocations and Component life cycle

Session Beans

- Session Beans are not persistent
- Used by client to perform application tasks like user validation, credit card verifier, etc.
- Each Session Bean is accessed by a single client at any point of time.
- Can be “Stateless” or “Stateful”

Stateless Session Bean

- All instances of Stateless Session Bean of a given type are identical.
- They can be easily pooled
- Provides better scalability for large number of clients



Stateless Session Bean creation steps

- Write a business interface
- Write Session Bean class
- Compile the Java Code
- Provide the Deployment Descriptor
- Create a ejb-jar file, containing the generated classes and deployment descriptor
- Deploy the ejb-jar file
- Write the client to check the functionality of the bean

Stateless Session Bean

- The business interface

```
package com.mindtree.ejb;

/**
 * @author Banu Prakash
 *
 */
public interface ComputeBeanRemote {
    public double add(double first, double second);
}
```

Stateless Session Bean

- The Bean class

```
package com.mindtree.ejb;

import javax.ejb.Remote;
import javax.ejb.Stateless;

/**
 * @author Banu Prakash
 *
 */
@Stateless
@Remote(ComputeBeanRemote.class)
public class ComputeBean implements ComputeBeanRemote{

    public double add(double first, double second) {
        return first + second;
    }
}
```

@Stateless indicates it is a stateless session bean

@Remote allows remote client view
Parameters passed between your EJB and the client should be passed "by value" instead of "by reference."

@Local allows local client view
With local client view, you can do pass-by-reference

EJB client

- jboss-ejb-client.properties file
 - Create jboss-ejb-client.properties file in clients classpath

```
endpoint.name=client-endpoint
remote.connectionprovider.create.options.org.xnio.Options.SSL_ENABLED=false
remote.connections=default

remote.connection.default.host=localhost
remote.connection.default.port = 4447
remote.connection.default.connect.options.org.xnio.Options.SASL_POLICY_NOANONYMOUS=false
```


EJB client

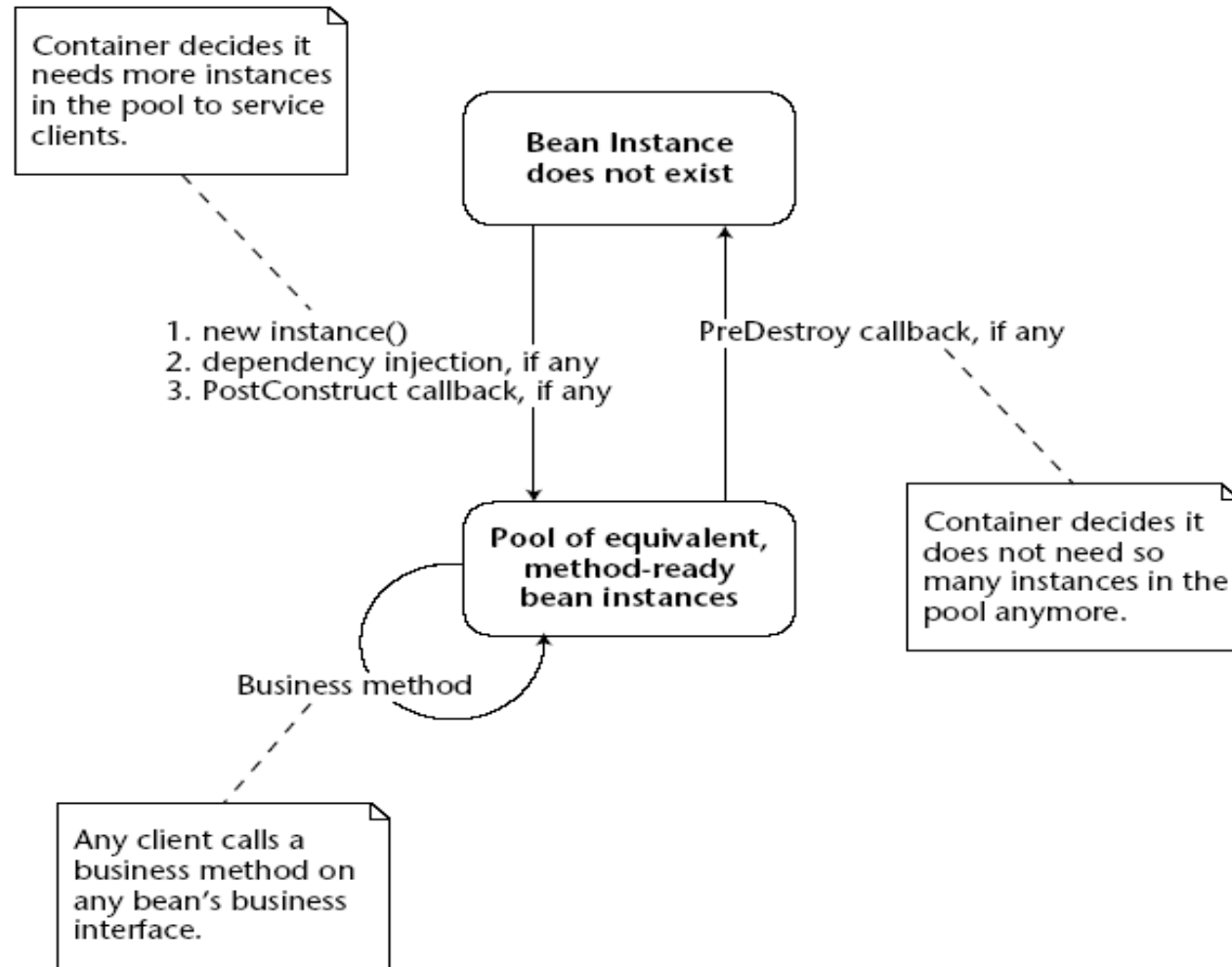
- Standalone client

```
@SuppressWarnings("unchecked")
private static ComputeBeanRemote lookupRemoteEJB() throws NamingException {
    @SuppressWarnings("rawtypes")
    final Hashtable jndiProperties = new Hashtable();
    jndiProperties.put(Context.URL_PKG_PREFIXES,
        "org.jboss.ejb.client.naming");
    final Context context = new InitialContext(jndiProperties);
    return (ComputeBeanRemote) context
        .lookup("ejb:/HelloWorld/ComputeBean!com.mindtree.ejb.ComputeBeanRemote");
}
```

- Syntax for Session EJB JNDI Naming:
 - <app-name>/<module-name>/<distinct-name>/<bean-name>!<fully-qualified-
classname-of-the-remote-interface>

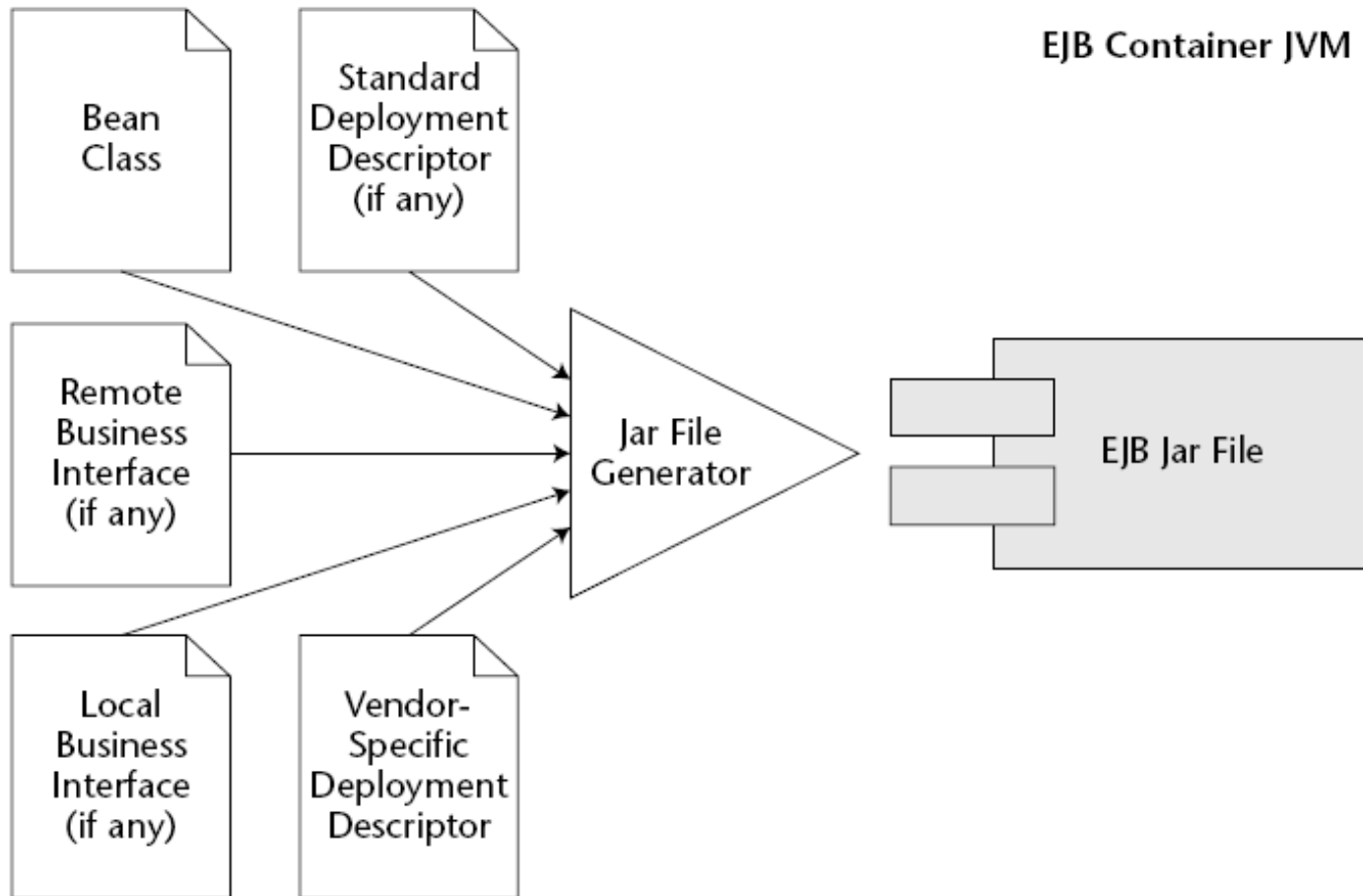
```
public static void main(String[] args) throws NamingException {
    ComputeBeanRemote bean = lookupRemoteEJB();
    System.out.println(bean);
    System.out.println(bean.add(45.22, 23.33));
}
```

Stateless session bean lifecycle



The life cycle of a stateless session bean.

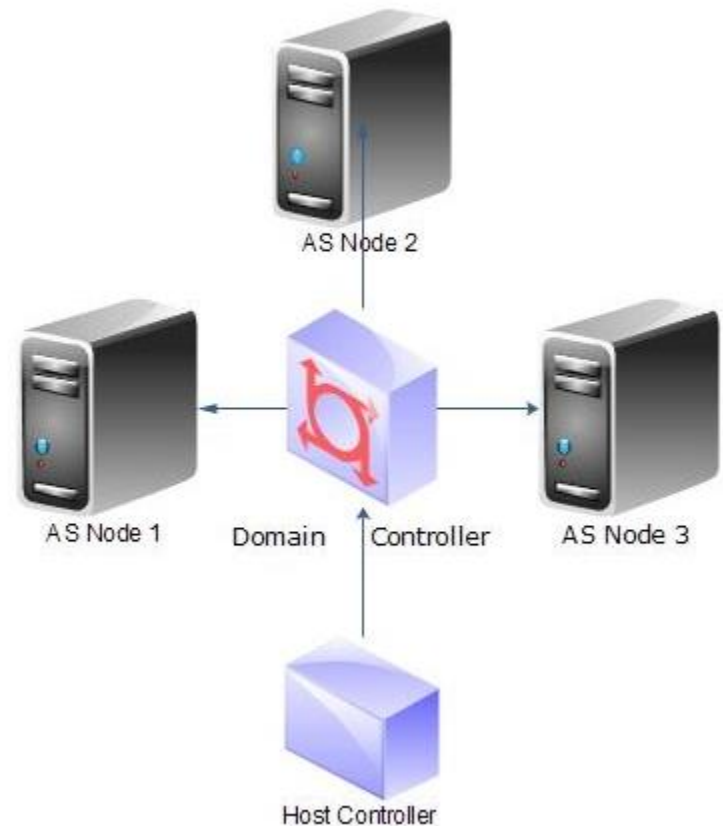
EJB jar file



Creating an Ejb-jar file.

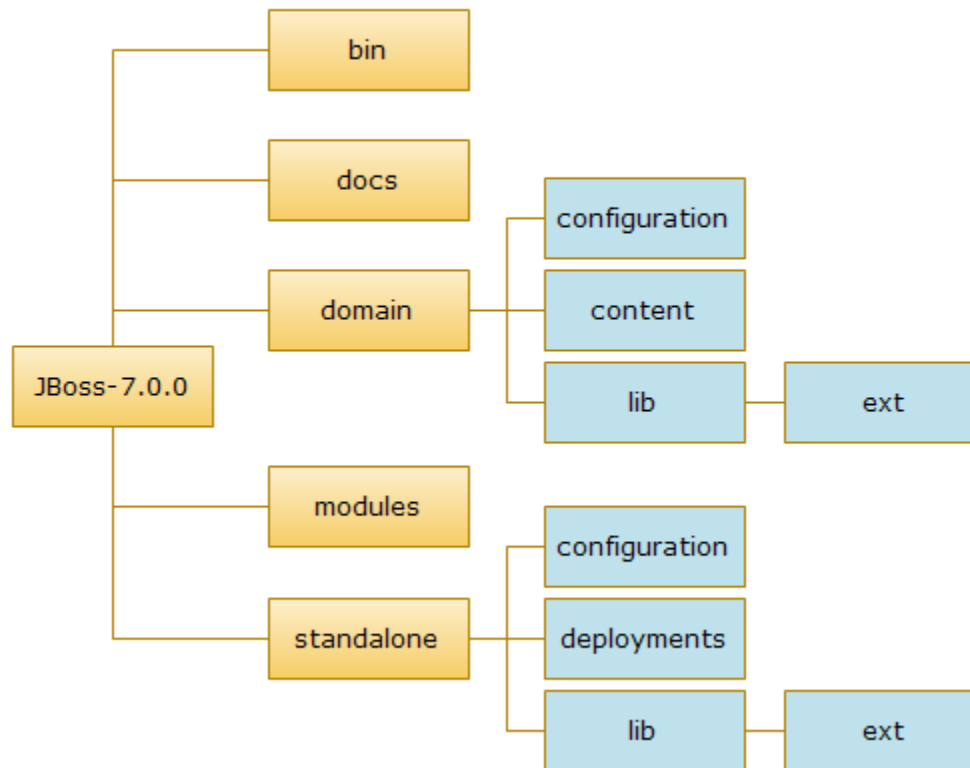
JBoss server

- A JBoss domain is used to manage and coordinate a set of application server instances. JBoss AS 7 in **domain mode** spawns multiple JVMs which build up the domain. Besides the AS instances, two more processes are created: the **Domain Controller** which acts as management control point of the domain and the **Host Controller** which interacts with the domain Controller to control the lifecycle of the AS instances



JBoss 7 folder structure

- Standalone **servers** and **domain** servers



JBoss deployment

- The **deployments** folder is the location in which users can place their deployment content (for example, WAR, EAR, JAR, SAR files) to have it automatically deployed into the server runtime. Users, particularly those running production systems, are encouraged to use the JBoss AS management APIs to upload and deploy deployment content instead of relying on the deployment scanner subsystem that periodically scans this directory
- As soon as the deployer HD scanner detects the application, the module is moved to the work folder of the application, leaving a placeholder `yourProj.war.deployed` file in the deployments folder.

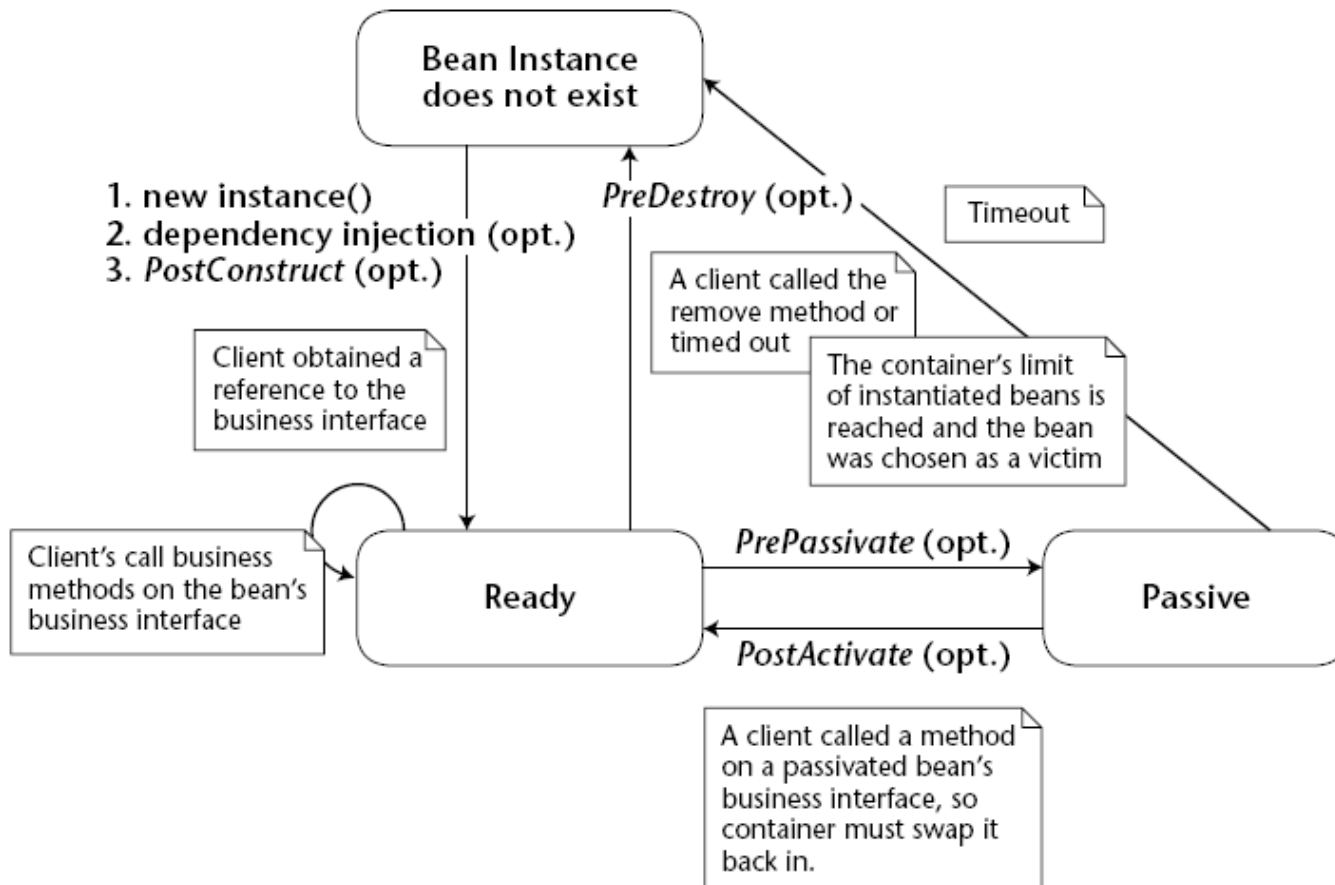
EJB classes can be now packaged in .WAR files!

- The EJB specification states that enterprise beans needs to be packaged in an enterprise module called an xxx.jar.
- If a Web application needed to use the EJB classes (without replicating the interfaces into the Web Archive) , you had to package the EJB and WEB applications in an xxx.ear archive.
- The EJB 3.1 specification removed the restriction that enterprise bean classes must be packaged in an ejb-jar file. You now have the option of placing EJB classes directly in the .war file, using the same packaging guidelines that apply to web application classes.
- This means that you can place EJB classes under the WEB-INF/classes directory or in a .jar file within the WEB-INF/lib directory.
- The EJB deployment descriptor is also optional. If it's needed, you can package it as a WEB-INF/ejb-jar.xml file.

Stateful Session Bean

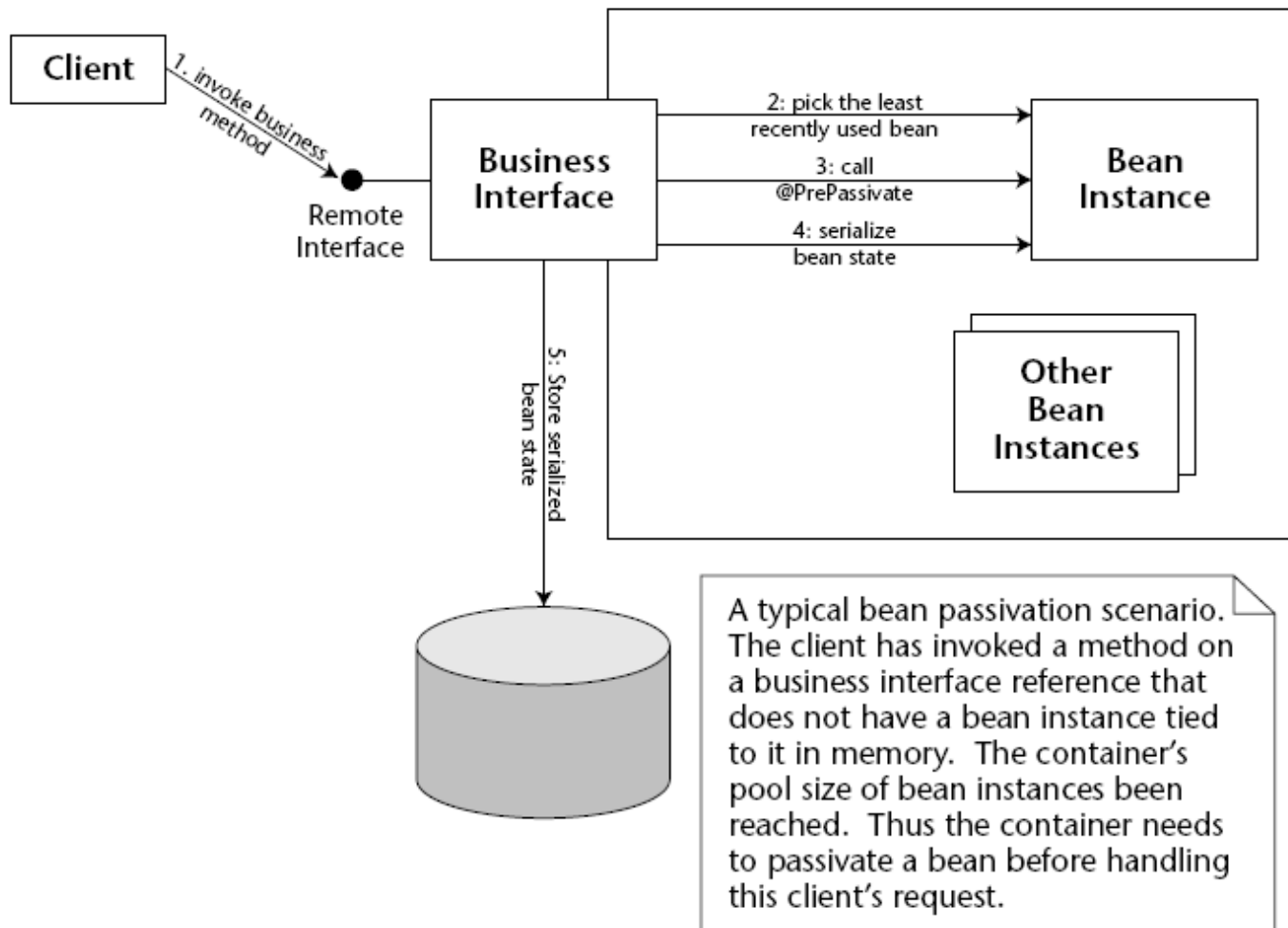
- Stateful session beans retain state on behalf of an individual client.
- Some business process are naturally drawn-out conversations over several requests.
- A Stateful session Bean is a bean that is designed to service business process that span multiple method requests or transactions.

Stateful Session Bean life cycle



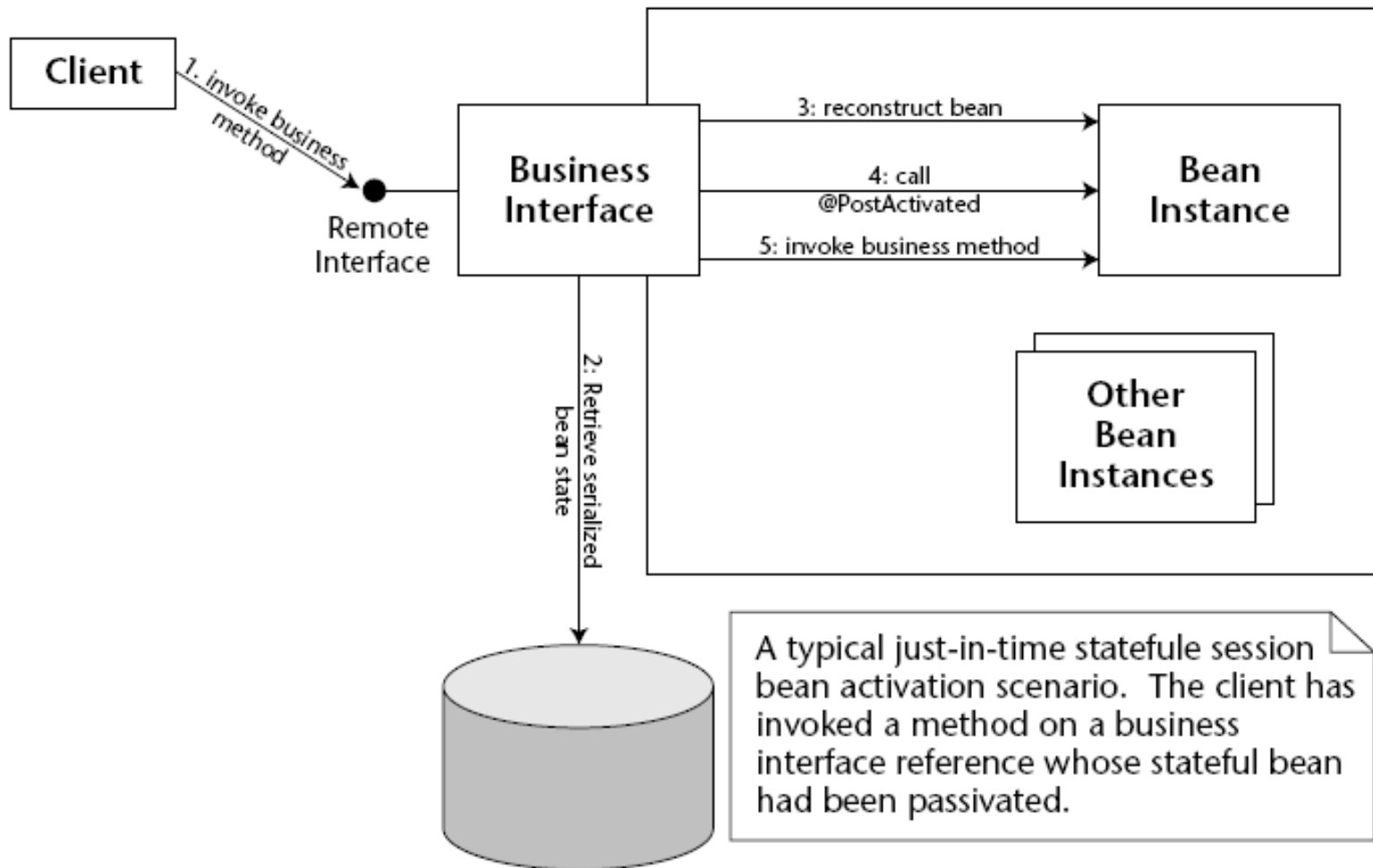
Life cycle of a stateful session bean.

Stateful bean passivation

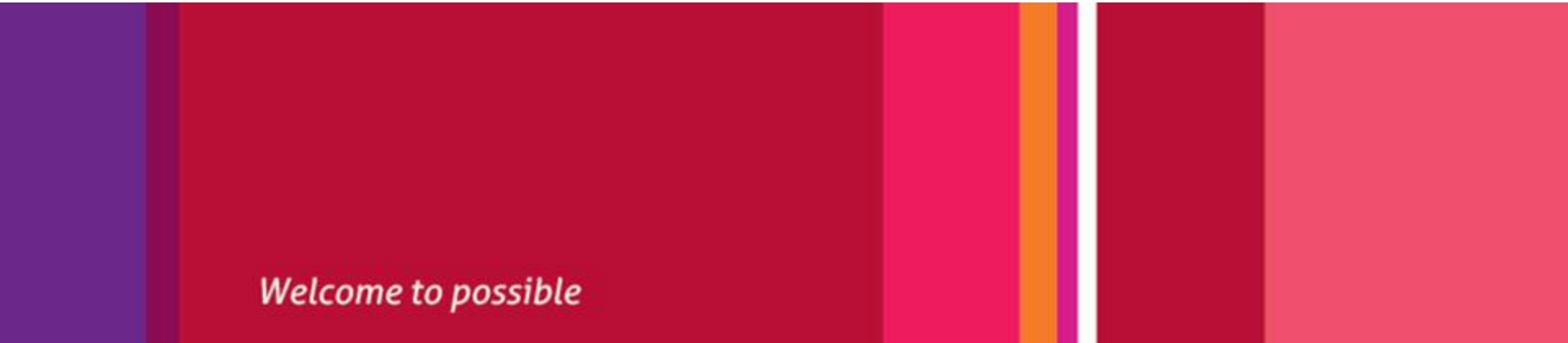


Passivation of a stateful bean.

Stateful bean activation



Activation of a stateful bean.



Welcome to possible

Name

Email

www.mindtree.com/social