Attention : les exercices pratiques de cette semaine nécessitent la connaissance du langage Java (qui fait partie des pré-requis du MOOC). Si vous n'êtes pas familiers avec ce langage, vous pouvez simplement passer cette partie. N'oubliez pas cependant de répondre au quizz de fin de semaine.

PRACTICAL EXERCISE 3 (W3_PE3): DYNAMIC MBEAN

In this third exercise, you will extend the TimeServer project with a second MBean of type dynamic. The attributes and the operations exposed by the MBean are the same as the first exercise. In addition, you will use a global variable *searchCity* available in the class TimeServer.java that takes the boolean values true or false.

If the value of this variable is true, the dynamic MBean should expose a new operation *isPresent* that takes as argument, the name of a city and returns true if the city is available in the server, otherwise it returns false.

For validation purpose the Java class name of your MBean should be *TimeServerDynMO.java,* and the ObjectName of the MBean to be registered in the MBeanServer should be*:*

fun.mooc.management.jmx.mbeans:type=TimeServerDynMOMBean

A template of the implementation class of the dynamic Mbean (*TimeServerDynMO.java*) follows:

4: Adding a Notification
Practical_Exercise_Quiz

Evaluations
Week_Evaluation
Echéance le avril 10, 2022 at 22:00 UTC

Aidez-nous à améliorer ce MOOC

- Week 4: Next-Generation Management Protocols
- Votre avis nous intéresse

```
import java.util.Iterator;

import javax.management.Attribute;
import javax.management.AttributeList;
import javax.management.AttributeNotFoundException;
import javax.management.DynamicMBean;
import javax.management.InvalidAttributeValueException;
import javax.management.MBeanAttributeInfo;
import javax.management.MBeanConstructorInfo;
import javax.management.MBeanException;
import javax.management.MBeanInfo;
import javax.management.MBeanOperationInfo;
import javax.management.MBeanParameterInfo;
import javax.management.ReflectionException;

import fun.mooc.management.jmx.timeserver.ThreadPoolServer;
import fun.mooc.management.jmx.timeserver.TimeServer;

public class TimeServerDynMO implements DynamicMBean {
	private Hashtable<String,Object> attributs = new
Hashtable<String,Object>();
	private ThreadPoolServer server;
	public TimeServerDynMO(ThreadPoolServer server) {
		this.server = server;
		attributs.put("NumberOfCities",0);
		attributs.put("SizeOfThreadsPool",0);
		/* To be completed for two more attributes*/
	}
	@Override
	public Object getAttribute(String attribute)
			throws AttributeNotFoundException, MBeanException,
			ReflectionException {
		Object result=null;
		if (attributs.containsKey(attribute)) {
			if (attribute.equals("NumberOfCities"))
				return server.getNumberOfCities();
	   /*To be completed for the other attributes: SizeOfThreadsPool,
	    NumberOfRequests,NumberOfUnknownCities*/

		}else{
			throw new AttributeNotFoundException(attribute);
		}
		return result;
	}
```

```
throws AttributeNotFoundException,
InvalidAttributeValueException,
        MBeanException, ReflectionException {
    String name = attribute.getName();
    Object value = attribute.getValue();
    if (name.equals(/* To be completed with the name of attribute
*/)) {
        server.setSizeOfThreadsPool((int)value);
    }else{
        throw new AttributeNotFoundException(name);
    }
}

@Override
public AttributeList getAttributes(String[] attributes) {
    AttributeList resultat = new AttributeList();
    for (String cle : attributes) {
        if (attributs.containsKey(cle)) {
            Object value;
            try {
                value = getAttribute(cle);
                resultat.add(new Attribute(cle, value));
            } catch (AttributeNotFoundException e) {
                e.printStackTrace();
            } catch (MBeanException e) {
                e.printStackTrace();
            } catch (ReflectionException e) {
                e.printStackTrace();
            }
        }
    }
    return resultat;
}

@Override
public AttributeList setAttributes(AttributeList attributes) {
    for (Iterator i = attributes.iterator(); i.hasNext();) {
        Attribute attr = (Attribute) i.next();
        try {
            setAttribute(attr);
        } catch (AttributeNotFoundException e) {
            e.printStackTrace();
        } catch (InvalidAttributeValueException e) {
            e.printStackTrace();
        } catch (MBeanException e) {
```

```
                e.printStackTrace();
            }
        }
        return attributes;
    }

    @Override
    public Object invoke(String actionName, Object[] params, String[]
signature)
            throws MBeanException, ReflectionException {
        try {
            if (actionName.equals(/*To be completed with the name
of the operation*/)) {
                server.stop();
            }
            if (actionName.equals(/*To be completed with the name
of the operation*/)) {
                return server.isPresent((String)params[0]);
            }
            return null;
        } catch (Exception x) {
            throw new MBeanException(x);
        }
    }

    @Override
    public MBeanInfo getMBeanInfo() {
        MBeanParameterInfo[] sansParamInfo = new
MBeanParameterInfo[0];
        MBeanParameterInfo[] params = new
MBeanParameterInfo[1];
        int i = 0;
        MBeanOperationInfo[] operations=null;
        MBeanAttributeInfo attribs[] = new
MBeanAttributeInfo[attributs.size()];
        for (String cle : attributs.keySet()) {
            if (cle.equals("SizeOfThreadsPool")){
                attribs[i] = new MBeanAttributeInfo(cle,
attributs.get(cle).getClass()
                        .getName(), "Description de l'attribut " + cle,
true, true, false);
            }else {
                attribs[i] = new MBeanAttributeInfo(cle,
attributs.get(cle).getClass()
                    .getName(), "Description de l'attribut " + cle, true,
```

```
            ...,
        }
        params[0] = new MBeanParameterInfo("server",
"ThreadPoolServer", "Instance of the treaded server");
        MBeanConstructorInfo[] constructeurs = new
MBeanConstructorInfo[1];
        constructeurs[0] = new
MBeanConstructorInfo("TimeServerDynMO",
                "Constructeur de la classe", params);

        if (TimeServer.searchCity) {
            operations = new MBeanOperationInfo[2];
            operations[0] = new MBeanOperationInfo("stopServer",
                    "Stop the Time Server", sansParamInfo,
void.class.getName(),
                    MBeanOperationInfo.ACTION);
            MBeanParameterInfo[] paramInfo = new
MBeanParameterInfo[1];
            paramInfo[0] = new MBeanParameterInfo("city",
String.class.getName(), "Name of the city");
            operations[1] = new MBeanOperationInfo("isPresent",
                    "Check if a city is in the DB", paramInfo,
boolean.class.getName(),
                    MBeanOperationInfo.ACTION);
        }else{
            operations = new MBeanOperationInfo[1];
            operations[0] = new MBeanOperationInfo("stopServer",
                    "Stop the Time Server", sansParamInfo,
void.class.getName(),
                    MBeanOperationInfo.ACTION);

        }
        return new MBeanInfo(getClass().getName(),
"TimeServerDynMO",
                attribs, constructeurs, operations, null);

    }
}
```

- You need to update the file TimeServer.java to register your Dynamic Mbean in the same way as the Standard Mbean, using an ObjectName and the method registerMBean.
- To validate your work, you have to execute the validation script, available in the folder */home/user/jmx/validation*, as following:

If the script returns the code 200, so you succeeded the second exercise.

Please note, that you have to execute the command: *source /usr/local/bin/set-jmx-lab-env.sh* in each new terminal to set correctly the lab environment variables.

QUESTION W3.PE3.1 (1/1 point)

If your implementation is valid after running the script, you should obtain a 8 digits validation token, that you need to copy and paste in the answer box. What is the value of the validation token that you obtained?

72377361 ✔

72377361

Vous avez utilisé 1 essais sur 3

SOLUTION: TIMESERVERDYNMO.JAVA

```
package fun.mooc.management.jmx.mbeans;

import java.util.Hashtable;

import java.util.Iterator;


import javax.management.Attribute;

import javax.management.AttributeList;

import javax.management.AttributeNotFoundException;

import javax.management.DynamicMBean;

import javax.management.InvalidAttributeValueException;

import javax.management.MBeanAttributeInfo;
```

```
import javax.management.MBeanException;

import javax.management.MBeanInfo;

import javax.management.MBeanOperationInfo;

import javax.management.MBeanParameterInfo;

import javax.management.ReflectionException;


import fun.mooc.management.jmx.timeserver.ThreadPoolServer;

import fun.mooc.management.jmx.timeserver.TimeServer;


public class TimeServerDynMO implements DynamicMBean {

 private Hashtable<String,Object> attributs = new Hashtable<String,Object>();

 private ThreadPoolServer server;

 public TimeServerDynMO(ThreadPoolServer server) {

  this.server = server;

  attributs.put("NumberOfCities",0);

  attributs.put("SizeOfThreadsPool",0);

  attributs.put("NumberOfRequests",0);

  attributs.put("NumberOfUnknownCities",0);

 }

 @Override

 public Object getAttribute(String attribute)

   throws AttributeNotFoundException, MBeanException,

   ReflectionException {
```

```
 if (attributs.containsKey(attribute)) {
  if (attribute.equals("NumberOfCities"))
   return server.getNumberOfCities();
  if (attribute.equals("SizeOfThreadsPool"))
   return server.getPoolSize();
  if (attribute.equals("NumberOfRequests"))
   return server.getNumberOfrequests();
  if (attribute.equals("NumberOfUnknownCities"))
   return server.getNumberOfUnkownCities();
 }else{
  throw new AttributeNotFoundException(attribute);
 }
 return result;
}

@Override
public void setAttribute(Attribute attribute)
  throws AttributeNotFoundException, InvalidAttributeValueException,
  MBeanException, ReflectionException {
 String name = attribute.getName();
 Object value = attribute.getValue();
 if (name.equals("SizeOfThreadsPool")) {
```

```
}else{
 throw new AttributeNotFoundException(name);
}

}

@Override
public AttributeList getAttributes(String[] attributes) {
 AttributeList resultat = new AttributeList();
 System.out.println(attributes);
 for (String cle : attributes) {
  if (attributs.containsKey(cle)) {
   Object value;
   try {
    value = getAttribute(cle);
    resultat.add(new Attribute(cle, value));
   } catch (AttributeNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
   } catch (MBeanException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
   } catch (ReflectionException e) {
```

```
  e.printStackTrace();
  }
 }
}
return resultat;
}

@Override
public AttributeList setAttributes(AttributeList attributes) {
 for (Iterator i = attributes.iterator(); i.hasNext();) {
  Attribute attr = (Attribute) i.next();
  try {
   setAttribute(attr);
  } catch (AttributeNotFoundException e) {
   e.printStackTrace();
  } catch (InvalidAttributeValueException e) {
   e.printStackTrace();
  } catch (MBeanException e) {
   e.printStackTrace();
  } catch (ReflectionException e) {
   e.printStackTrace();
  }
```

```
return attributes;
}

@Override
public Object invoke(String actionName, Object[] params, String[] signature)
  throws MBeanException, ReflectionException {
 try {
  if (actionName.equals("stopServer")) {
   server.stop();
  }
  if (actionName.equals("isPresent")) {
   System.out.println("isPresent "+params[0]);
   return server.isPresent((String)params[0]);
  }
  return null;
 } catch (Exception x) {
  throw new MBeanException(x);
 }
}

@Override
public MBeanInfo getMBeanInfo() {
 MBeanParameterInfo[] sansParamInfo = new MBeanParameterInfo[0];
```

```
int i = 0;

MBeanOperationInfo[] operations=null;

MBeanAttributeInfo attribs[] = new MBeanAttributeInfo[attributs.size()];

for (String cle : attributs.keySet()) {

 if (cle.equals("SizeOfThreadsPool")){

  attribs[i] = new MBeanAttributeInfo(cle, attributs.get(cle).getClass()

    .getName(), "Description de l'attribut " + cle, true, true, false);

 }else {

  attribs[i] = new MBeanAttributeInfo(cle, attributs.get(cle).getClass()

   .getName(), "Description de l'attribut " + cle, true, false, false);

 }

 i++;

}

 params[0] = new MBeanParameterInfo("server", "ThreadPoolServer", "Instance of the
treaded server");

 MBeanConstructorInfo[] constructeurs = new MBeanConstructorInfo[1];

 constructeurs[0] = new MBeanConstructorInfo("TimeServerDynMO",

  "Constructeur de la classe", params);


 if (TimeServer.searchCity) {

  operations = new MBeanOperationInfo[2];

  operations[0] = new MBeanOperationInfo("stopServer",

   "Stop the Time Server", sansParamInfo, void.class.getName(),
```

```
  MBeanParameterInfo[] paramInfo = new MBeanParameterInfo[1];

  paramInfo[0] = new MBeanParameterInfo("city", String.class.getName(), "Name of the city");

  operations[1] = new MBeanOperationInfo("isPresent",

    "Check if a city is in the DB", paramInfo, boolean.class.getName(),

    MBeanOperationInfo.ACTION);

 }else{

  operations = new MBeanOperationInfo[1];

  operations[0] = new MBeanOperationInfo("stopServer",

    "Stop the Time Server", sansParamInfo, void.class.getName(),

    MBeanOperationInfo.ACTION);

 }

 return new MBeanInfo(getClass().getName(), "TimeServerDynMO",

   attribs, constructeurs, operations, null);

}

}
```

SOLUTION: TIMESERVER.JAVA

```
package fun.mooc.management.jmx.timeserver;

import java.lang.management.ManagementFactory;

import javax.management.InstanceAlreadyExistsException;

import javax.management.MBeanRegistrationException;

import javax.management.MBeanServer;
```

```
import javax.management.NotCompliantMBeanException;

import javax.management.ObjectName;

import fun.mooc.management.jmx.mbeans.TimeServerBaseMO;

import fun.mooc.management.jmx.mbeans.TimeServerDynMO;


public class TimeServer {

 public static boolean searchCity = true;

 public static void main(String[] args) throws InterruptedException{

  CityTimeZone helper = new CityTimeZone("/cities1000.txt");

  ThreadPoolServer server = new ThreadPoolServer(9000,helper);

  // Start the MBean server

  MBeanServer mbs = ManagementFactory.getPlatformMBeanServer();

  ObjectName stdName,dynName = null;

  try {

   stdName = new
ObjectName("fun.mooc.management.jmx.mbeans:type=TimeServerBaseMOMBean");

   TimeServerBaseMO sMbean= new TimeServerBaseMO(server);

   mbs.registerMBean(sMbean, stdName);

   dynName = new
ObjectName("fun.mooc.management.jmx.mbeans:type=TimeServerDynMOMBean");

   TimeServerDynMO dMbean= new TimeServerDynMO(server);

   mbs.registerMBean(dMbean, dynName);

  } catch (InstanceAlreadyExistsException | MBeanRegistrationException
```

```
e.printStackTrace();
}
// Start the time server
Thread thServer = new Thread(server);
thServer.start();
thServer.join();
}
}
```