OSGi

1. Import (with copy)
2. Set target platoform, import demo/lib/derby jar if needed)
3. Set port: SzolgintDemo.launch: 8081
4. Run
5. Chrome Rest Client-ben:
   - GET http://localhost:8081/bookstore **(**üzenet OK)
   - GET http://localhost:8081/bookstore/books
6. Add Book
   - PUT http://localhost:8081/bookstore/books/1984/2500/1
7. Add user
   - PUT http://localhost:8081/bookstore/users/geza

.NET

8. DB indít (config managerből)
9. Projektet beállít a 8082-es portra
10. Teszt: http://localhost:8082/ServicesREST/BookstoreREST.svc/books/price/1984
    - Itt álljunk meg és nézzük meg a JSON-ra formázó taget

jBPM

11.jBPM mappába be, cmd:

```
ant start.demo
```

WF

12.New jBPM project

- name: BookOrder
- next: empty project és csak Add sample Java class for loading…
- next: jBPM runtime (C:\software\jbpm-installer\runtime\lib)

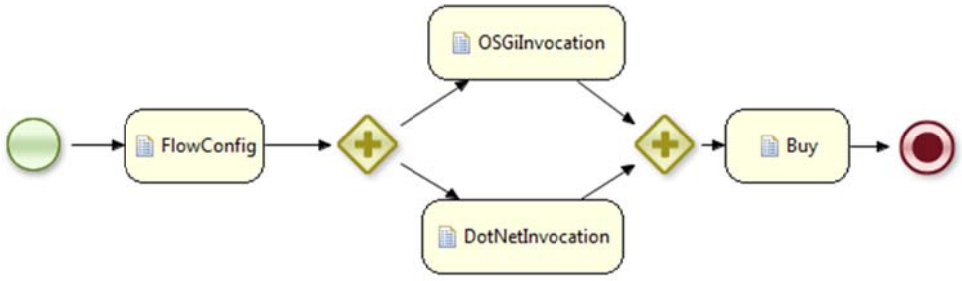
13.Properties>Build Path, jarok hozzáadása (C:\software\jbpm-installer\runtime)

WF összeállítása

14.src/main/resources-ba: new BMPN2 process (filename:bookorder)

- Id: hu.bme.mit.wf.bookorder

15.Rakjuk össze a workflowt script node-okkal:



16.állítsuk be a script node-okra, hogy logoljanak:

```
System.out.println("...");
```

17.Le kéne futtatni – ehhez kell egy main.

18.src/main/java: New package:

- hu.bme.mit.bookorder
- hu.bme.mit.bookorder.handlers

19.bookorder packagebe main class: ProcessMain névvel (add main method)

```
public class ProcessMain {
	public static void main(String[] args) throws Exception {
		KnowledgeBase kbase = readKnowledgeBase();
		StatefulKnowledgeSession ksession =
			kbase.newStatefulKnowledgeSession();

		ksession.startProcess("hu.mit.bme.wf.bookorder");
	}

	private static KnowledgeBase readKnowledgeBase() throws Exception {
		KnowledgeBuilder kbuilder =
			KnowledgeBuilderFactory.newKnowledgeBuilder();
		kbuilder.add(ResourceFactory.newClassPathResource
			("bookorder.bpmn"), ResourceType.BPMN2);
		return kbuilder.newKnowledgeBase();
	}
}
```

20.Run. Elvileg jól logol.

WorkItemHandlerek megírása

21.src/main/resources új mappa: META-INF

22.itt new File: MyWorkDefinitions.**wid**

```
import org.drools.process.core.datatype.impl.type.StringDataType;

[
  [
    "name" : "CheckPriceInvocation",
    "parameters" : [
      "URL" : new StringDataType(),
      "Message" : new StringDataType(),
      "Method" : new StringDataType()
    ],
    "displayName" : "CheckPriceInvocation",
    "icon" : "icons/webservice.gif"
  ]
]
```

- (ehhez kell: resources-ra új mappa (META-INF mellé): icons és abba tegyük be: webservice.gif)

23.META-INF-be: új File: drools.rulebase.conf

```
drools.workDefinitions = WorkDefinitions.conf MyWorkDefinitions.wid
```

24. Ha most újranyitjuk a process view-t, akkor van új elem a toolboxon. De kell hozzá handler!

25. Handlers package: new class: CheckPriceInvocationWorkItemHandler

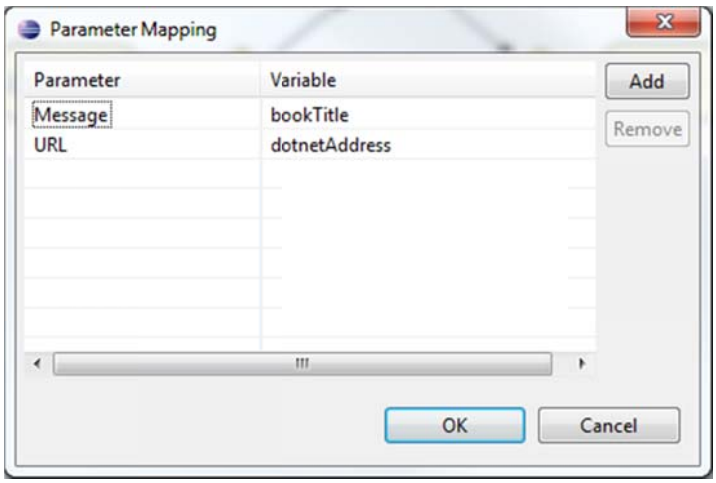
- implements WorkItemHandler, addoljuk a hiányzó metódusokat

```
public class CheckPriceInvocationWorkItemHandler implements WorkItemHandler {
	@Override
	public void abortWorkItem(WorkItem arg0, WorkItemManager arg1) {
		// TODO Auto-generated method stub

	}

	@Override
	public void executeWorkItem(WorkItem wi, WorkItemManager wim) {
		String address = (String) wi.getParameter("URL");
		String message = (String) wi.getParameter("Message");
		String method = (String) wi.getParameter("Method");

		System.out.println("Calling WS @" + address + message + " –
				method: " + method);
	}

}
```

26. Cseréljük le a két megfelelő script node-ot, paraméterezzük a workflowt és futtassuk!
27. Ehhez kellenek flow változók (process properties: Variables) és mind String
    - osgiAddress, dotnetAddress, buyAddress
    - bookTitle
    - osgiPrice, dotnetPrice
    - finalResult
28. FlowConfig Action:

```
kcontext.setVariable("osgiAddress", "http://localhost:8081/bookstore/books/price/");
kcontext.setVariable("dotnetAddress", "http://localhost:8082/ServicesREST/BookstoreREST.svc/books/price/");
kcontext.setVariable("bookTitle", "1984");
```

29. Parameter Mapping @ WorkItems



30. Nem parameter mappelve, hanem kézzel beállítva: Method = GET
31. Run-elhasal, mert be kell jegyezni a WF-be. Ehhez a mainben:

```
ksession.getWorkItemManager().registerWorkItemHandler("CheckPriceInvocation
", new CheckPriceInvocationWorkItemHandler());
```

32. Run

33. Para: nem fut le a Buy – mert nincs completelve a workitemhandler. Ezért az execute utolsó sora ez lesz:

```
wim.completeWorkItem(wi.getId(), null);
```

34. Run. Most zsírul lefut.

RESTFUL WS-ek hívása

35. Import Jersey JARs.
36. execute-ba:

```
try {
	Client client = Client.create();
	WebResource webResource = client.resource(address + message);
	String ret = "";

	if(method.equalsIgnoreCase("GET")){
		System.out.println("Executing via GET...");
		ret = webResource.get(String.class);
	}
	else if(method.equalsIgnoreCase("PUT")){
		System.out.println("Executing via PUT...");
		ret = webResource.put(String.class);
	}
	else if(method.equalsIgnoreCase("POST")){
		System.out.println("Executing via POST...");
		ret = webResource.post(String.class);
	}

} catch (Exception e) {
	e.printStackTrace();
}

int price = Integer.parseInt(ret);

System.out.println("Price is: " + price);
```

37. Run!
38. Logol, cool. De hogy adjuk vissza az eredményt a WF-nek?

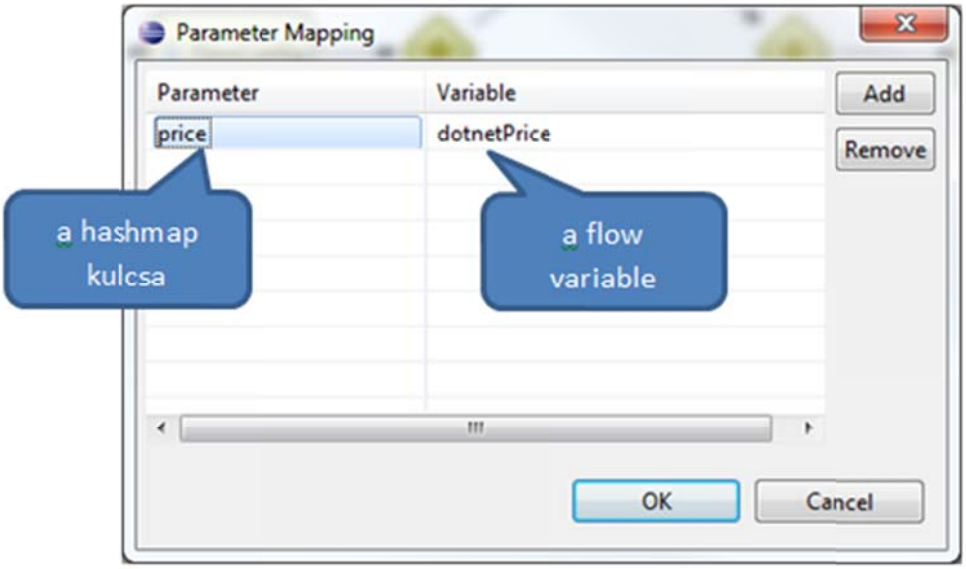
Eredmények visszaadása

39. Az előző sysout helyett:

```
Map<String, Object> results = new HashMap<String, Object>();
results.put("price", price);

wim.completeWorkItem(wi.getId(), results);
```

- (a nullos complete pedig a catch-be kerüljön.) így már tudunk rá hivatkozni a WF szintjén.

40. WF nézet, properties, Result mapping



41.módosítsuk a Buy actiont:

```
System.out.println("OSGi price: " + kcontext.getVariable("osgiPrice"));
System.out.println(".NET price: " + kcontext.getVariable("dotnetPrice"));

if (kcontext.getVariable("osgiPrice") < kcontext.getVariable("dotnetPrice")){
  System.out.println("Buying book from the OSGi store");
}
if (kcontext.getVariable("osgiPrice") > kcontext.getVariable("dotnetPrice")){
  System.out.println("Buying book from the .NET store");
}
```

42.Run. Fut – átadtuk az eredményeket. Most meg kéne venni a könyvet.

Könyv megvétele

43.A korábbi WIH-hez hasonlóan egy újat készítünk és paraméterezzük.

44.BuyBook.wid:

```
import org.drools.process.core.datatype.impl.type.StringDataType;
[
  [
    "name" : "BuyBook",
    "parameters" : [
      "URL" : new StringDataType(),
      "BookTitle" : new StringDataType(),
      "Method" : new StringDataType()
    ],
    "displayName" : "BuyBook",
    "icon" : "icons/webservice.gif"
  ]
]
```

45.rulebase updatelése:

```
drools.workDefinitions = WorkDefinitions.conf MyWorkDefinitions.wid
BuyBook.wid
```

46.Handler: egy az egyben copy paste az előzőből, de:

- message helyett booktitle
- nem kell a parse a price-ra (az a sor törörlhető)

```
Map<String, Object> results = new HashMap<String, Object>();
results.put("ret", ret);
```

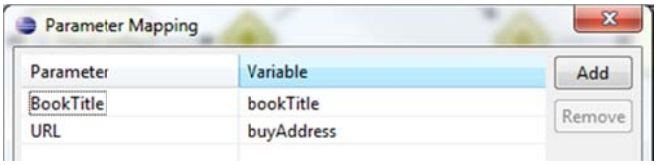
47.REGISZTRÁLNI KELL A MAINBEN!

```
ksession.getWorkItemManager().registerWorkItemHandler("BuyBook", new
                                                BuyBookWorkItemHandler());
```

48.A Buy node után bedobjuk az új WorkItemet, a Buy-ban kofigolni fogjuk az inputját:

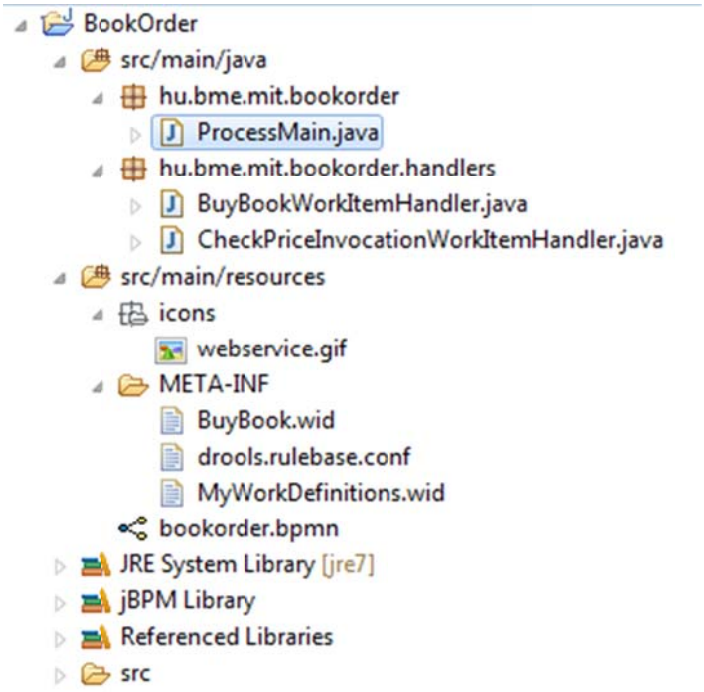
```
System.out.println("OSGi price: " + kcontext.getVariable("osgiPrice"));
System.out.println(".NET price: " + kcontext.getVariable("dotnetPrice"));

if (kcontext.getVariable("osgiPrice") < kcontext.getVariable("dotnetPrice")){
  System.out.println("Buying book from the OSGi store");
  kcontext.setVariable("buyAddress", "http://localhost:8081/bookstore/buyBook/geza/");
}
if (kcontext.getVariable("osgiPrice") > kcontext.getVariable("dotnetPrice")){
  System.out.println("Buying book from the .NET store");
  kcontext.setVariable("buyAddress", "http://localhost:8082/ServicesREST/BookstoreREST.svc/buyBook/geza/");
}
```

49.Parameter mapping és method = POST



50.Mappeljük ki a finalResult-ba a ret-et és írassuk ki egy script node-dal.

VÉGLEGES PROJECT LAYOUT:



VÉGLEGES FLOW: