Runtime Java EE

Spring MVC – Jak na front-end

Petr Musil

Spring MVC – front-end

- ✓Mapování requestu
- ✓Freemarker view: resolver a dekorátor
- ✓GET Template
- ✓Form formatters
- ✓POST Template
- ✓Generování menu
- ✓Obsluha jQuery requestu
- ✓DTO mít či nemít?

Controller: request mapping

✓Bindování requestu do controlleru

```
@Controller
@MenuItem(key = PnrController.KEY, showAfter = SpecialOfferController.KEY)
@RequestMapping(value = "/" + PnrController.KEY + ".do")
public class PnrController extends ControllerBase<BookingLight> {

        @RequestMapping(method = RequestMethod.GET)
        public ModelAndView processGet() {
                …
        }

        @RequestMapping(method = RequestMethod.POST, params = PNR_FILTER_KEY)
        public ModelAndView processSetPnrFilter(PnrFilter filter, BindingResult errors) {
                …
        }
}
```

Controller: view relover

- ✓Common controller – nalezení template pro view
- ✓package **….**web.controller obsahuje základní kontroller
- ✓Další controllery se nachází v podbalíčcích

```
protected final String getViewName(String view) {
            StringBuilder pkg = new StringBuilder();
            pkg.append(getClass().getPackage().getName().toLowerCase());
            if (pkg.toString().startsWith(ControllerBase.PKG_NAME)) {
                        pkg.delete(0, ControllerBase.PKG_NAME.length());
                        if (pkg.length() > 0 && pkg.charAt(0) == '.') {
                                    pkg.delete(0, 1);
                        }
            }
            return pkg.append('/').append(view).toString().replaceAll("\\.", "/");
}
```

View „***address***“ pro controller v balíčku ….web.controller.***customer*** se bude hledat v ***<fm_template_base>/customer/address.ftl***

Pozor na cílové prostředí pro běh aplikace!

GET Template

- ✓Vynucení implementace požadovaných metod
- ✓Sjednocené obsluhování požadavku

```
public abstract class SimpleGetTemplate extends RequestTemplate {

            public final ModelAndView processGet() {
                        …
            }

            protected void checkPrecondition(PageFlowErrors errors) {
                        //no default precondition
            }

            protected Class<? extends ControllerBase<?>> getPreconditionFailureRedirect() {
                        return null;
            }

            protected abstract void callBackend(Map<String, Object> backendResult) throws
WebservicesLocalizedException;

            protected abstract void updateCacheState(Map<String, Object> backendResult);

}
```

GET Template: process


```
public final ModelAndView processGet() {
            PageFlowErrors errors = new PageFlowErrors();
            //lze obsloužit tento požadavek tímto controllerem?
            checkPrecondition(errors);

            if (errors.hasError()) {
                        String errorsFormatted = formatErrors(errors);
                        Class<? extends ControllerBase<?>> clazz = getPreconditionFailureRedirect();
                        Map<String, Object> nextPage = new HashMap<String, Object>();
                        nextPage.put(ControllerBase.BACKEND_ERROR, errorsFormatted);
                        return redirectTo(clazz, nextPage);
            }

            Map<String, Object> backendResult = new HashMap<String, Object>();
            try {
                        callBackend(backendResult);
            } catch (Throwable t) {
                        processTemplateException(this, t, backendResult, true);
            }

            //naplnit společnými daty modelu a controller specific
            updateCacheState(backendResult);

            //vytvořit model a předat view vrstvě
            return createModel(backendResult);
}
```

View: Freemarker šablony

- ✓Umístění freemarker šablon a jejich přetěžování
- ✓Base projekt s obecnou funkcionalitou a specifické projekty
  - •WebApp projekt: WEB-INF/<fm_base>/templatePath
  - •Base projekt - jar: /<fm_base>/templatePath

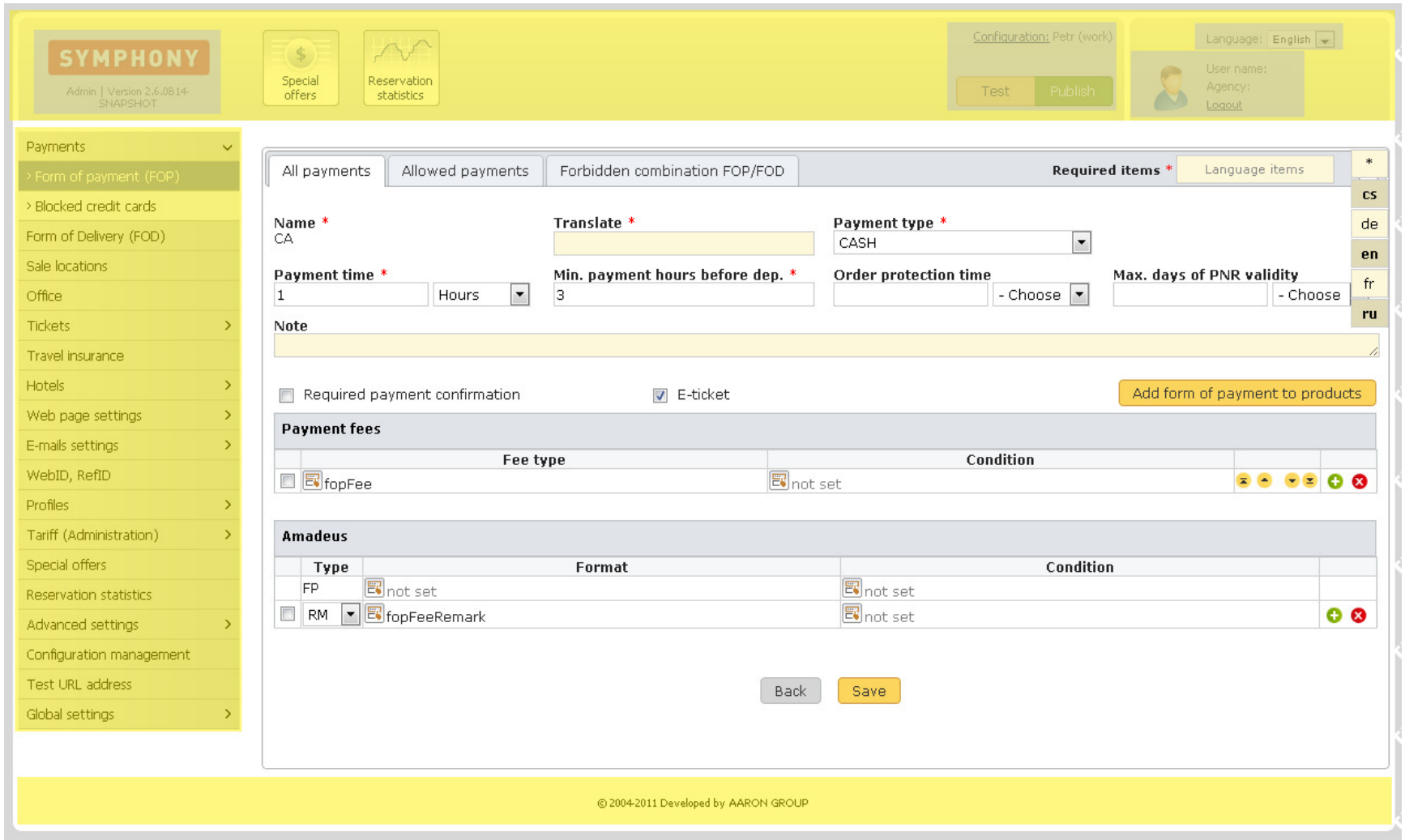
Spring-servlet-applicationContext.xml

```
<bean id="freemarkerConfig" class="cz.aag.webadmin.web.freemarker.FreemarkerConfigurer">
            <property name="classLoadPath" value="/fm/" />
            <property name="freemarkerSettings">
                        <props>
                                    <prop key="number_format">###0.##</prop>
                        </props>
            </property>
            …
            <property name="devLoadPathKey" value="s2" />
</bean>
```

View: Freemarker loader


```
Public class FreemarkerConfigurer extends FreeMarkerConfigurer {

public Configuration createConfiguration() throws IOException, TemplateException {
          Configuration config = super.createConfiguration();
          List<TemplateLoader> loaders = new ArrayList<TemplateLoader>();
          loaders.add(new WebappTemplateLoader(servletContext, "/WEB-INF" + getClassLoadPath()));
          loaders.add(new ClassTemplateLoader(FreeMarkerConfigurer.class, getClassLoadPath()));
          FreemarkerMultiTemplateLoader multiTemplateLoader = new
FreemarkerMultiTemplateLoader(loaders.toArray(new TemplateLoader[0]));
          config.setTemplateLoader(multiTemplateLoader);

          return config;
}

...
}
```

View: Freemarker decorator



View: Freemarker decorator

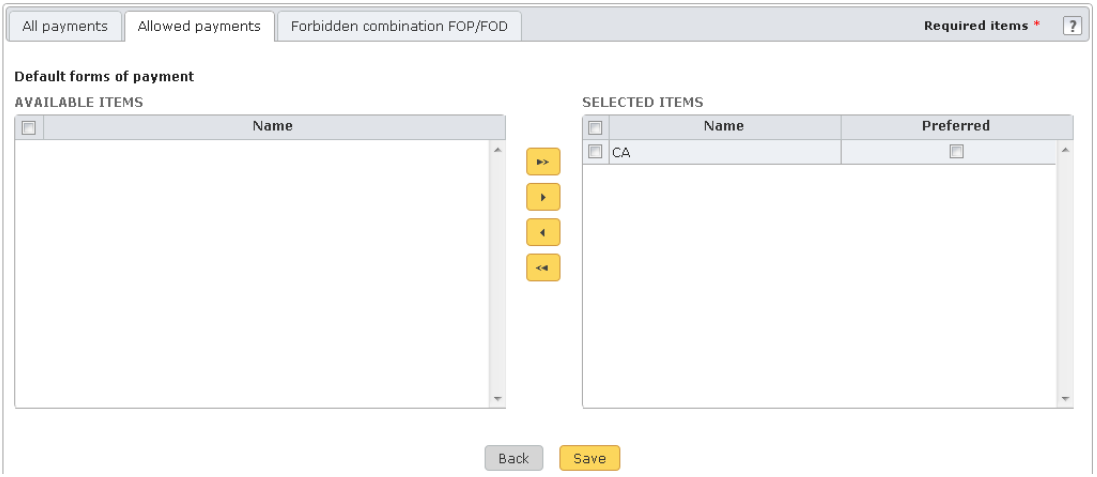

```
public class FreemarkerView extends FreeMarkerView {

private static final Map<String, String> REPLACE_KEY_TEMPLATE = new HashMap<String, String>();

static {
          final String prefix = "include/";
          final String suffix = ".ftl";
          FreemarkerView.REPLACE_KEY_TEMPLATE.put("blockLogin", prefix + "loginInfo" + suffix);
          ...
}

protected void processTemplate(Template template, SimpleHash model, HttpServletResponse response)
throws IOException, TemplateException {
          response.setCharacterEncoding("UTF-8");
          model.put("nowDate", new Date());

          final Locale locale = template.getLocale();
          for (final Map.Entry<String, String> keyTemplate : REPLACE_KEY_TEMPLATE.entrySet()) {
                    final StringWriter write = new StringWriter();
                    Template template = getTemplate(keyTemplate.getValue(), locale)
                    template.process(model, new PrintWriter(write, true));
                    model.put(keyTemplate.getKey(), write.toString());
          }

          final StringWriter write = new StringWriter();
          template.process(model, new PrintWriter(write, true));
          model.put("blockContainer", write.toString());

          this.getTemplate("layout.ftl", locale).process(model, response.getWriter());
}
```

View: Freemarker decorator


```
<#include "./include/htmlHead.ftl" />
<body>
      <#include "./developer.ftl" />
      <div id="page" class="big-corners">
            <div id="head" class="big-top-corners">
                  <#include "./include/logo.ftl" />
                  <#include "./include/htmlHeadInclude.ftl" />
            </div>
            <#if currentPublishConfiguration??>
                  <div id="login-info" class="big-top-corners">
                        <div class="inlay">
                              ${blockLogin!""}
                        </div>
                  </div>
                  <div class="clear"></div>
            </#if>
            ${blockMenu!""}
            <#if backendError??>
                  <span class="error backendError">${backendError!}</span>
            </#if>
            <#include "./cascadeDelete.ftl" />
            <div id="main">${blockContainer!""}</div>
            <div class="clear"></div>
            <div id="footer">${blockCopy!""}</div>
      </div>
      <div id="jQueryDialogBox" style="hidden"></div>
      <div id="jQueryDialogBoxContainer" style="hidden"></div>
</body>
</html>
```

Formulář

- ✓GET vkládá nový formulářový objekt do modelu
- ✓Zabezpečení formuláře:
  - ✓Generování hodnoty instanceId pro každý formulář a jeho následná kontrola při POSTU
- ✓Uchování stavu formuláře
  - ✓Form action
  - ✓Form submit action



AARONGROUP

POST Template

- ✓Vynucení implementace požadovaných metod
- ✓Sjednocené obsluhování požadavku

```
public abstract class SimplePostTemplate<F extends FormBase> extends SimpleTemplate<F> {

        protected F preValidationProcess(F form) {
                return form;
        }

        protected final void assertFormInstance(F form, Errors errors) {
                …
        }
        protected ModelAndView processValidationErrors(F form, Errors errors) {
                …
        }

        public final ModelAndView processPost(F form, BindingResult errors) {
                …
        }

        protected abstract String getFormCacheKey();

        protected abstract Validator getValidator();

        protected abstract void callBackend(F form, Map<String, Object> backendResult) throws
WebservicesLocalizedException;

}
```

POST Template: process


```
public final ModelAndView processPost(F form, BindingResult errors) {

            assertFormInstance(form, errors);
            Validator validator = getValidator();
            if (validator == null) {
                        errors.rejectValue("formState", "form.error.validator.null");
            } else {
                        try {
                                    invokeValidator(validator, preValidationProcess(form),errors);
                        } catch (Throwable t) {
                                    errors.rejectValue("formState", "form.error.validator.fail");
                        }
            }

            if (errors.hasErrors()) {
                        return processValidationErrors(form, errors);
            }

            Map<String, Object> backendResult = new HashMap<String, Object>();
            try {
                        callBackend(form, backendResult);
            } catch (Throwable t) {
                        processTemplateException(this, t, backendResult, true);
            }

            putPageCache(getFormCacheKey(), form);
            return redirectToGet();
}
```

Form instance check


```
protected final void assertFormInstance(F form, Errors errors) {
            FormBase cachedFormBase = findFormInCache();
            if (cachedFormBase == null) {
                        errors.rejectValue("formState", "form.error.instanceId.notfoundInCache");
                        return;
            }

            if (!StringUtil.hasLength(cachedFormBase.getInstanceId())) {
                        errors.rejectValue("formState", "form.error.instanceId.noInstaceId");
                        return;
            }

            if (!cachedFormBase.getInstanceId().equals(form.getInstanceId())) {
                        errors.rejectValue("formState", "form.error.instanceId.wrongInstanceId");
                        return;
            }
}
```

Form validation errors

AARONGROUP

www.aag.cz

```
protected ModelAndView processValidationErrors(F form, Errors errors) {

      form.setEdited(true);

      putPageCache(getFormCacheKey(), form);

      Map<String, Object> map = new HashMap<String, Object>();
      map.put(getFormCacheKey(), form);
      map.put(ControllerBase.FORM_ERRORS, errors.getAllErrors());
      updateCacheStateOnValidationError(map, errors);

      putPageCache(ControllerBase.FORM_ERRORS, errors.getAllErrors());

      return createModel(map);
}
```

Form formatter

```
public class DateFormatter implements Formatter<Date> {

      public static final DateFormat FORMAT = new SimpleDateFormat("yyyy/MM/dd HH:mm");

      public String print(Date date, Locale locale) {
            if (date == null) {
                  return "";
            }
            try {
                  return DateFormatter.FORMAT.format(date);
            } catch (Exception e) {
                  return "";
            }
      }

      public Date parse(String dateString, Locale locale) throws ParseException {
            if (!StringUtil.hasLength(dateString)) {
                  return null;
            }

            try {
                  return DateFormatter.FORMAT.parse(dateString);
            } catch (ParseException e) {
                  return null;
            }
      }
}
```

Form formatter

```
public class Localization {
          private String code;
          private String name;
          private String comment;
          ...
}

Public class ProductDetail {
          private Localization defaultLocalization;
          private List<Localization> localizations;
          ...
}

JSON data {"code":"DEFAULT","comment":"lorem ipsum","name":"box"}

public Localization parse(String text, Locale locale) throws ParseException {
     if (!StringUtil.hasLength(text)) {
          return null;
     }
     try {
          Localization loc = new Localization();
          JSONObject json = StringUtil.parseJSON(text);
          loc.setCode(StringUtil.getJsonString(json, "code"));
          loc.setName(StringUtil.getJsonString(json, "name"));
          loc.setComment(StringUtil.getJsonString(json, "comment"));
          return loc;
     } catch (JSONException e) {
          throw new ParseException(e.getMessage(), 0);
     }
}
```

Front-end: menu


```
@Target( { ElementType.TYPE })
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface MenuItem {

      public static final String ROOT = "ROOT";
      public static final String NONE = "NONE";

      String key();

      String showAfter() default MenuItem.NONE;

      String parent() default MenuItem.ROOT;

      boolean ready() default true;

      String cssPath() default "";

      String userRole() default "";
}

@Controller
@MenuItem(key = PnrController.KEY, showAfter = SpecialOfferController.KEY)
@RequestMapping(value = "/" + PnrController.KEY + ".do")
public class PnrController extends ControllerBase<BookingLight> {
```

Front-end: menu

```
private void findMenuItems() {
	BeanDefinitionRegistry beanDefRegistry = (BeanDefinitionRegistry)
		((ConfigurableApplicationContext) this.applicationContext).getBeanFactory();
	Set<BeanDefinition> beanDefs = new ClassPathBeanDefinitionScanner(beanDefRegistry).
		findCandidateComponents("cz.aag.webadmin");
	Map<String, MenuElement> items = new HashMap<String, MenuElement>();
	for (BeanDefinition beanDef : beanDefs) {
		AnnotationMetadata metadata = ((AnnotatedBeanDefinition) beanDef).getMetadata();
		Map<String, Object> attrs = metadata.getAnnotationAttributes(MenuItem.class.getName());
		Boolean ready = (Boolean) attrs.get("ready");
		String key = (String) attrs.get("key");
		String userRole = (String) attrs.get("userRole");
		String parent = (String) attrs.get("parent");
		String showAfter = (String) attrs.get("showAfter");
		String link = getRequestMappingLink(metadata);
		if (StringUtil.hasLength(key, link)) {
			items.put(key, new MenuElement(key, parent, showAfter, link, ready, userRole));
			String css = (String) attrs.get("cssPath");
			if (StringUtil.hasLength(css)) {
				this.cssStyles.put(key, css);
			}
		}
	}

	MenuElement root = new MenuElement(MenuItem.ROOT, null, null, "index.jsp", true, null);
	sortMenu(items, root);
}
```

jQuery: request mapping

- ✓Obdoba POST requestu
- ✓Validace nemusí být povinná
- ✓Můžeme použít napřímo HttpServletRequest nebo POJO objekt jako u postu formuláře (použít i BindingResult)
- ✓Odpověd je automaticky v json formátu (spring-jsonview)

```
@RequestMapping(method = RequestMethod.POST, params = { "cascadeDeleteAccepted" })
public ModelAndView processCascadeDeleteAccepted(HttpServletRequest request) {
      return new SimpleJsonPostTemplate() {

            @Override
            protected void updateCacheState(ModelAndView mav) {
                  //osetreni ajax volani a ulozeni dat pro json reply do modelu
            }

            @Override
            protected void callBackend(Map<String, Object> backendResult) throws … {
                  //osetreni ajax volani a ulozeni dat pro json reply do backendResultu
            }
      }.processPost(request);
}
```

jQuery: AJAX request


```
var responseText = jQ.ajax({
      type: 'POST',
      url: baseUrl,
      async: false,
      data: urlParams,
      error: function(request, textStatus, error){
            if (textStatus == "parsererror" && request.status == 200) {
                  jQ(location).attr('href',"login.do");
            } else {
                  warnDialog(jsMsg.warn.ajaxCallFail);
            }
      }
}).responseText;

try {
            return parseJSON(responseText);
} catch(e) {
            return undefined;
}
```

DTO: mít či nemít?

AARONGROUP

www.aag.cz

- ✓Klonování objektů pro formuláře
- ✓Hlídání změn v API, které je mimo náš projekt

```
cz.aag.webadmin.web.controller.SimpleControllerBase
Interfaces:
          cdbfc73045f5c12b9ea848d1bf58e8460e03ee39
Annotations:
          cdbfc73045f5c12b9ea848d1bf58e8460e03ee39
Constructors:
          public cz.aag.webadmin.web.controller.SimpleControllerBase()
Methods:
     public static java.lang.String
     cz.aag.webadmin.web.controller.SimpleControllerBase.getControllerUrlMapping(java.lang.Class<?
     extends cz.aag.webadmin.web.controller.SimpleControllerBase>,boolean)
     protected final org.springframework.web.servlet.ModelAndView public boolean
     cz.aag.webadmin.web.controller.SimpleControllerBase.removeGlobalCache(java.lang.String)
     ...
Fields:
     26 org.apache.log4j.Logger LOG
     26 java.lang.String DEVELOPMENT_MODE
     2 cz.aag.webadmin.conf.EnvironmentManager environmentManager
     2 org.springframework.context.ApplicationContext applicationContext
     ...
Parent: 8b4c489b8196dc87ee6d1cdf4a9affdb7a61dd7f
```

Výsledek je MD5 třídy: acef4a73923e527b3e7195d75ff10977d793352d

Maven: check dependency


```
public class CheckDependencyPlugin extends AbstractMojo {
      public void execute() throws MojoExecutionException, MojoFailureException {
            getLog().info("********************* Artifact path. **********************");
            Set<?> artifacts = project.getDependencyArtifacts();
            List<URL> clUrls = new ArrayList<URL>();

            Artifact checkArtifact = null;
            for (Iterator<?> artifactIterator = artifacts.iterator(); artifactIterator.hasNext();) {
                  Artifact artifact = (Artifact) artifactIterator.next();
                  try {
                        if (GROUP_ID.equals(artifact.getGroupId()) &&
                                    ARTIFACT_ID.equals(artifact.getArtifactId())) {
                                    checkArtifact = artifact;
                        }
                        if (artifact.getFile() != null) {
                                    clUrls.add(artifact.getFile().toURI().toURL());
                        }
                  } catch (MalformedURLException e) {
                        throw new MojoFailureException("Wrong configuration: " + e.getMessage());
                  }
            }

            if (checkArtifact == null) {
                        throw new MojoFailureException("Artifact not found!");
            }
            ClassLoader cl = new URLClassLoader(clUrls.toArray(new URL[0]),
            getClass().getClassLoader());
            checkDependencies(checkArtifact, cl);
      }
      ...
}
```

Děkuji za pozornost

Otázky?