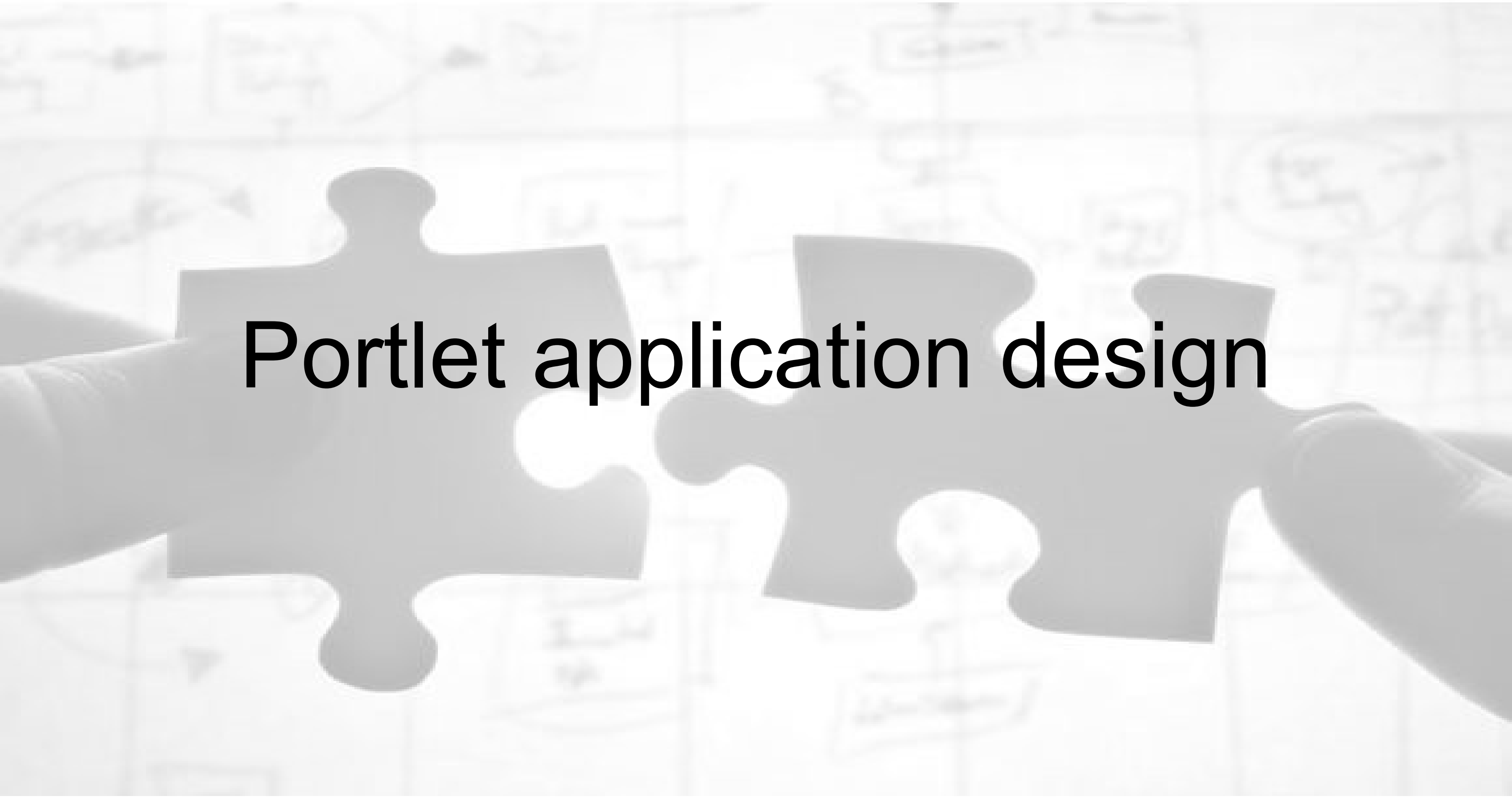
Best practices for portlet development

PV230 Podnikové portály
Petr Adámek, IBA CZ, s.r.o.
petr.adamek@ibacz.eu

Agenda

Best practices for portlet development

- Portlet application design
- UI design
- Coding guidelines
- Common mistakes
- Common problems and pitfalls with portlets



Portlet application design

Integration into portal

Application should be integrated into portal seamlessly

- Focus on key functionality
  - Provide easy and immediate access to the most useful information and services that users need 95% of the time
- Do not reproduce the look and feel of a visually complex application
  - See UI Design later

Integration into portal – IFRAMEs

Use IFRAMEs with caution

- The IFRAME fills its content based on a URL which must be addressable by the browser
- Not all browser levels support IFRAMEs
- If the content is larger than the IFRAME region, then horizontal and vertical scrolling should be enabled

Keep portlets simple

Portlets should be as simple as possible

- One use-case – one portlet
  - Beware of different content for different roles
  - Caching and performance problems
- Consider different portlets for different roles
- Beware of überportlet
  - Complicated maintenance
  - Performance problems
  - Reduced usebility

Portlet application design

Portlet decomposition rules I.

- One use-case – one portlet
- Make common functions available externally to portlet
  - Reusability
  - Portablity
- Use provider beans to represent model in MVC
  - Use Portlet only as Controller
  - Do not include application logic or web services calls into portlet

Portlet application design

Portlet decomposition rules II.

- Do not rely on portlet sessions if the portlet is to allow anonymous access
  - Problem with old web containers
  - Timeout problem
  - Performance problems for large amount of users
- Define inter-portlet communication dependencies and interfaces
  - Defensive programming – good input validation
  - Well defined contract → well used interface

UI Design

Design

What is design?

- A profession and discipline
- Simplifies and clarifies
- Provide order
- Provoke emotional response
- Adds value and meaning
- Provide information at a glance
- Integral part of the development cycle

Design

What isn't design?

- Design is not applied after the fact
  - Design is not "putting lipstick" on the product
- Design is not art

UI Design

Simplicity

- Use simple, intuitive user interfaces
- Think small
  - Portlets should be as functional as possible in a minimum of space.
  - Avoid large logos and disclaimers.
  - Click through to the back-end application for advanced functionality.
- Do not reproduce the look and feel of a visually complex application
  - HTML over which you have no control
  - output large amounts of HTML
  - Portlets should appear in the style of the portal
  - Portlets should require minimal processing.

UI Design

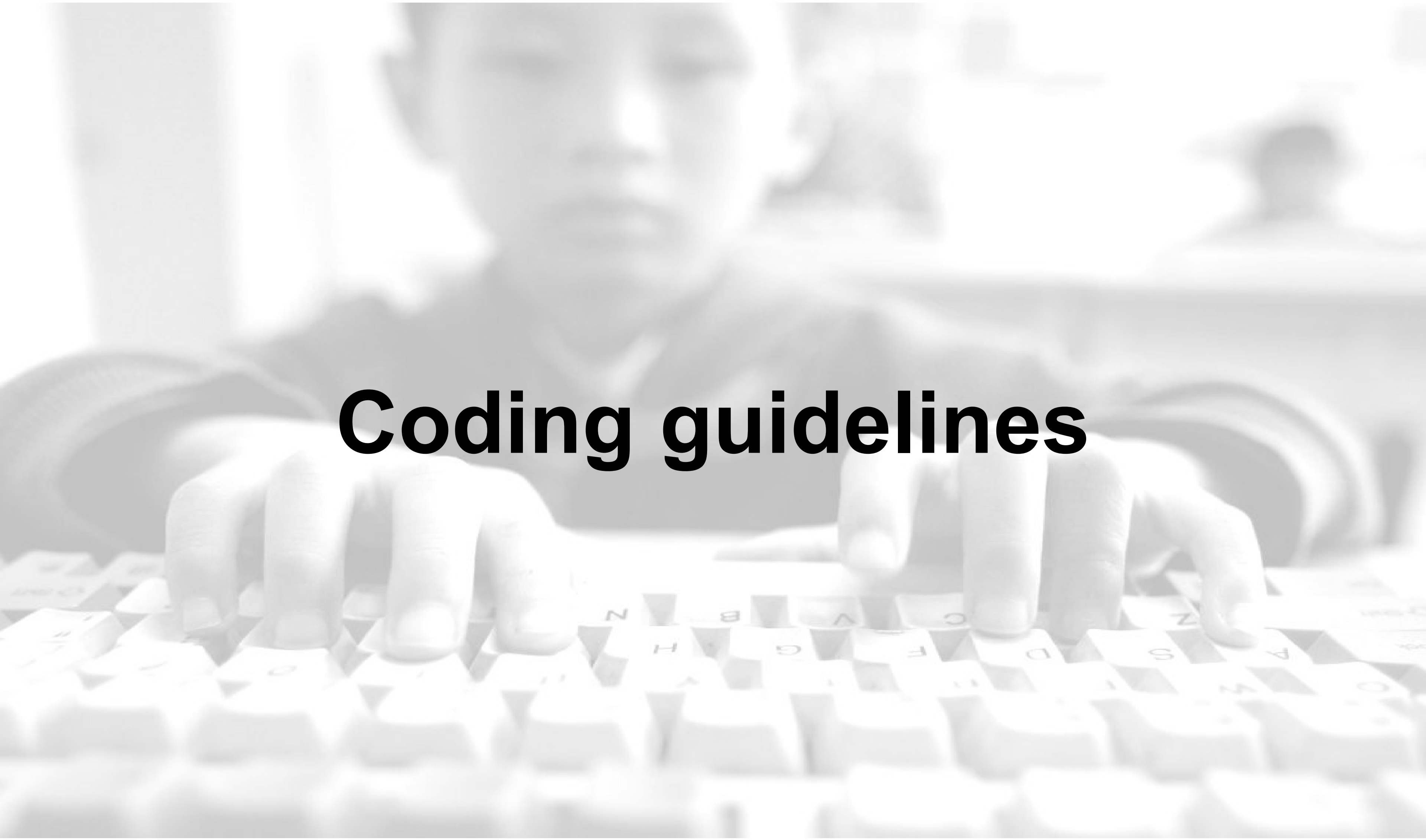
Look & Feel

- Respect look and feel of portal and its theme
  - Use standard portlet or portal CSS
- Design view to fit on a page with other portlets.
  - If you need more space
    - Use maximized portlet mode
    - Implement two versions of portlet
    - Set amount of information in portlet preferences
- Never impose an exact pixel size on a portlet
  - Users work in a variety of screen sizes and resolutions
  - Fixed size can destroy the inherent structure of the portal page

Accessibility

Make portlets as accessible as possible

- JSPs should be enabled for keyboard control and assistive technologies
  - Use ALT attribute with images
  - Use <LABEL> tags to associate labels with form input controls
  - Do not use color alone to denote state or information. For example, using red to emphasize text doesn't help those who are color blind.



Coding guidelines

Coding guidelines

Do not forget, that portlet is multithreaded

- Single instance is serving lots of conucurrent requests
- Methods must be thread safe
- Attributes are shared byl multiple threads
- Do not use attributes for storing data
- All resources stored in attributes must be thread-safe
- Beware of synchronization – one instance could serve many threads

Coding guidelines

Anotations

- Use anotated methods for precessing actions and events
  - `@ProcessAction(name=”destroyAllLightsAction”)`
  - `@ProcessEvent(qname=...)`
- This does not work when you override generic processAction(...)or processEvent(...) method

Preferences, attributes, parameters etc.

Use appropriate place for storing information

Portlet with multiple views

```
public static final String VIEW_PARAM = “view”;
public static final String VIEW_DETAIL = “detail”;

protected void doViewMain(RenderReq., RenderResp.).. {
    ...
}

protected void doViewDetail(RenderReq., RenderResp.).. {
    ...
}

public void doView(RenderReq., RenderResp.).. {
    String view = req.getParameter(VIEW_PARAM);
    if (view == null) {
        doViewMain(request, response);
    } else if (view.equals(VIEW_DETAIL)) {
        doViewDetail(request, response);
    } ...
}
```

Portlet with multiple views

Or use some MVC framework

- Page flow management
- Data validation
- Transforming form data into java objects

JSP

Rules for writing JSP I.

- Forms and functions must be uniquely named
  - Use <portlet:namespace/>
- Portlet Code pages should contain HTML fragments only
  - no <head>, <body>, etc.
- Use JSP style comments in JSPs instead of HTML style
  - <%-- This is a comment which will not appear in HTML code. --%>

JSP

Rules for writing JSP II.

- Use taglibs/common include files (jQuery...) whenever possible
  - Try to reuse libs from portal, do not introduce new frameworks
  - Potential problem with portability – solvable
- Identify all culturally dependent data
  - http://kore.fi.muni.cz:5080/wiki/index.php/I18n_-_Internacionalizace
- Do not use compound messages (concatenated strings) to create text

CSS

CSS rules

- Try to use portlet CSS classes whenever it is possible

Common mistakes

Common mistakes

- Libraries (commons-logging, log4j, JSTL, etc.)
  - Classloader hierarchy misunderstanding
  - Shared libraries must be loaded by global classloader
- Java EE versions
  - Descriptors or class format incompatibility
  - Develop on the same version of environment as on production system
- Portlet API libraries packed into portlet WAR
  - See the classloader problemodrážka

Common mistakes

Common mistakes

- method in forms with ActionURL in JSPs must be POST
  - Typical problem on Liferay
  - Very hard to investigate why form does not work
- Saving request specific data in portlet (portlet usualy works in dev. environment, but makes trouble in production with many users)
- Forgotten to set render params in processAction()
  - response.setRenderParametersMap(request.getParametersMap())

Common mistakes

Common mistakes

- Processing action using annotated methods vs. overriden generic processAction()
- Multi-part form data
- RequestDispatcher.include() vs. forward()
  - render / serveResource

Performance problems

Performance problems

- Wrong cache configuration
- Wrong desing
- Overloaded sessions
- Web services or other data sources latencies
- Consider caching at suitable level
- Data fetching in multiple threads
- Do not forget for performance testing

Common problems and pitfalls with portlets

MVC Frameworks

Integration with frameworks via bridge

- Struts, JSF, etc.
- Bridge is usually not as mature as the framework
- Performance problems (eg. JSF)
- Complexity of framwork and Portlet API combination (eg. JSF – lots of javascript, complicated JSF model + Portlet model, etc.)

Native Portlet frameworks

- Spring Portlet MVC

Missing portal specification

No portal specification

- Users
- Pages
- Portlets
- Administration

And More ☺

stop bad ideas before they can multiply

Questions

?