System FileSystem
Everything is a Stream

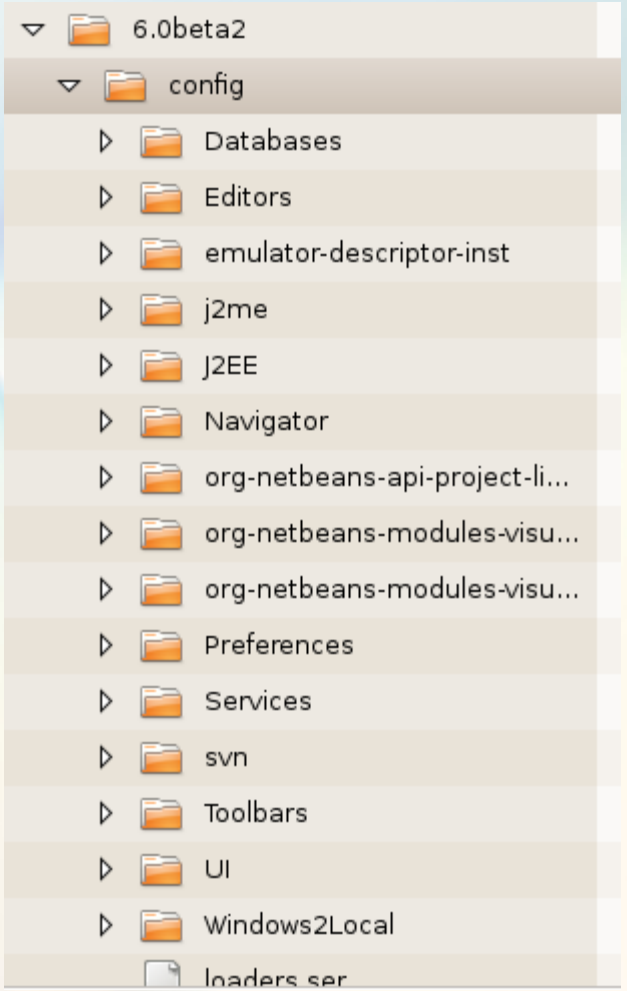
NetBeans

What is it?

- General registry of configuration data

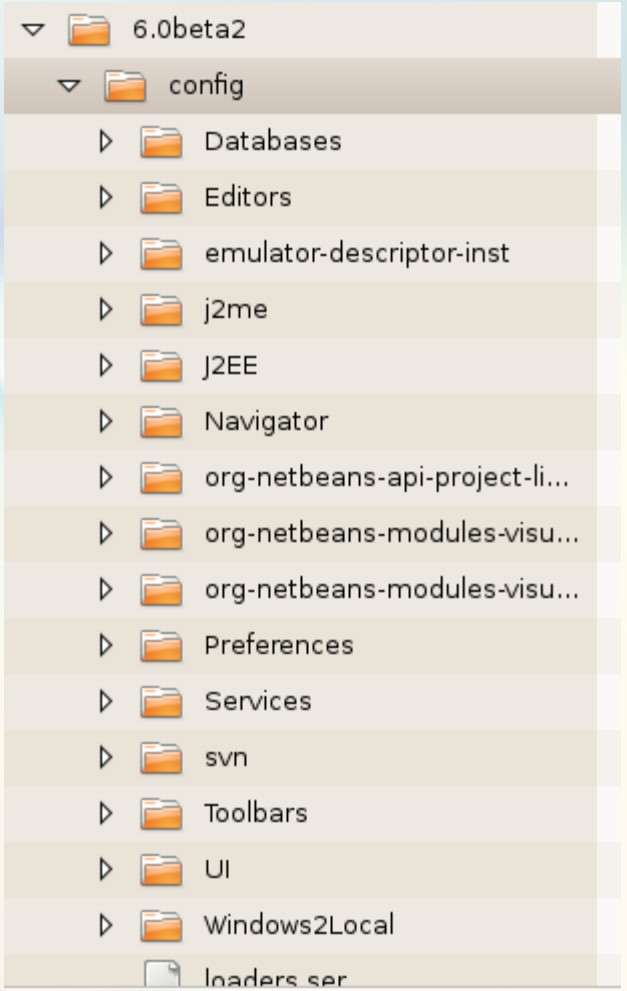
What is it?

- General registry of configuration data
- tree structure
- folders/directories
- files and streams



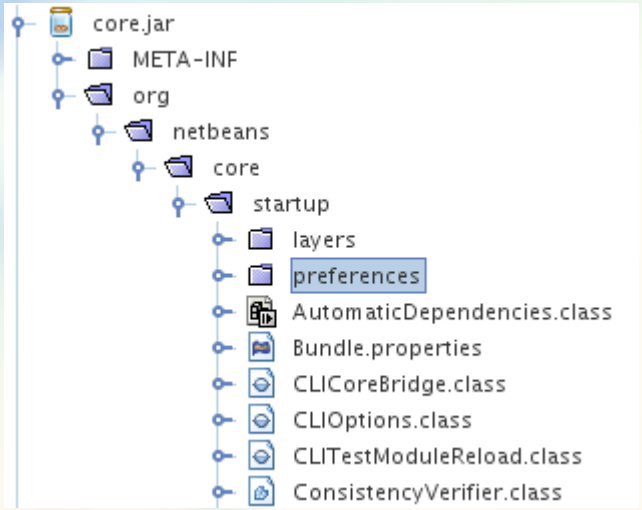
What is it?

- General registry of configuration data
  - > installation directory vs. user directory



What is a “filesystem”?

- NetBeans specific:
  - In NetBeans, we are dealing with *virtual* filesystems.
  - FileObjects, not java.io.File



FileObjects vs. java.io.File

- *Get* them from the filesystem, you do not create them.
- FileObjects have MIME types.
- FileObjects have attributes.
- FileObjects have input and output stream.
- You can listen for changes.

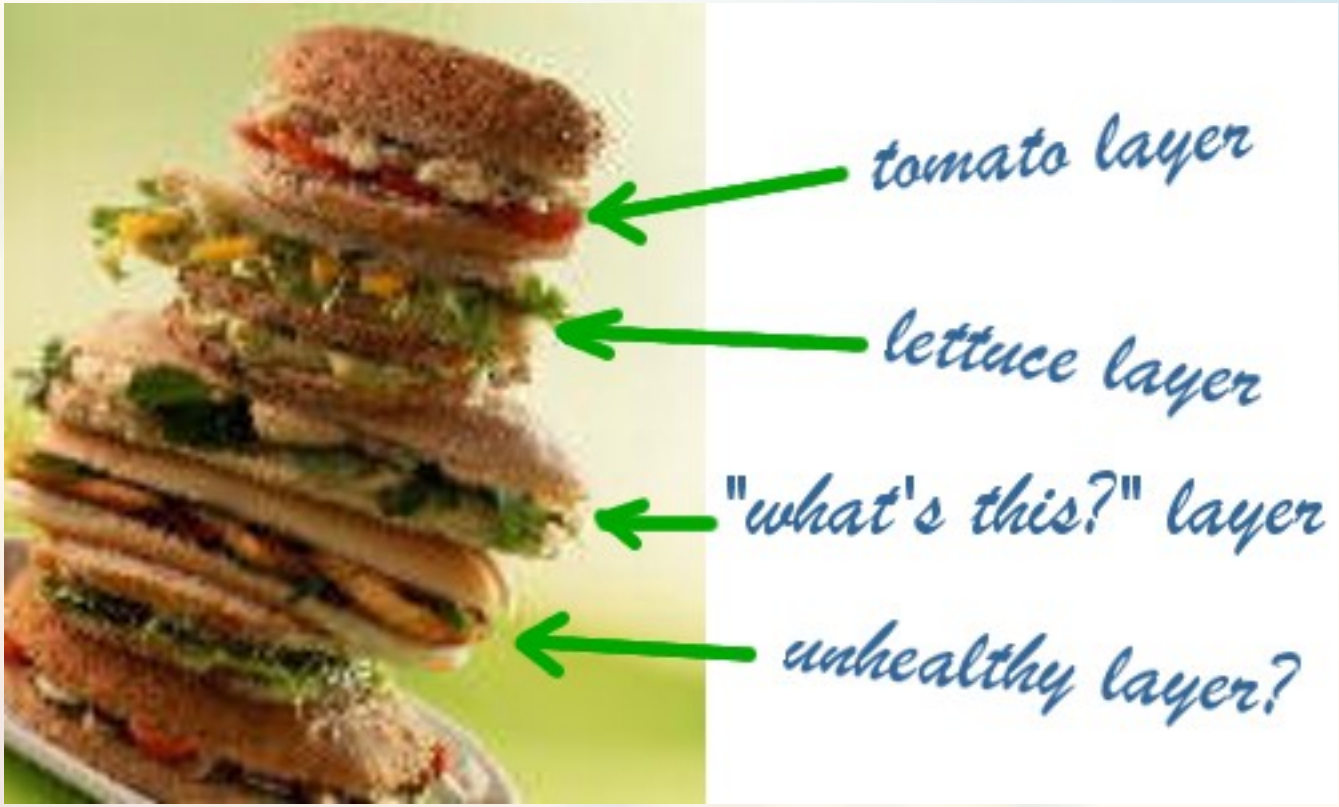
Lookup is a registry

- Simple registration of objects
  - put file in META-INF/services
  - default Lookup creates declared objects
- But:
  - want to associate additional attributes?
  - show objects in UI without instantiating?
- Examples:
  - Show icon in list, without creating object.

Filesystem is also a registry

- Filesystem
  - hierarchy of files and folders
    - e.g., local filesystem, works with real files
    - e.g., JARFilesystem
  - different folders for different purposes
  - declarative registration

System FileSystem



Giant sandwich...

- Every module provides a layer.
- A layer = an XML file.
- Each layer contains declarations for:
  - > menu items, toolbar buttons
  - > editor configurations
  - > Options window settings
  - > window persistence
  - > + much more

All of them together

= System FileSystem

Demo

Let's look at a layer file...

How do layers work?

1. NetBeans Platform **starts up**.
2. NetBeans Platform **finds all layers**.
3. XML Layers are **merged,** with one writable filesystem, into actual filesystem on disk.
4. NetBeans Platform **opens**.
5. **Results** of merge = application.

Influencing Content

- add to a folder

```
<filesystem>
    <folder name = “menu”>
        <file name="MyAppServices"/>
    </folder>
</filesystem>
```

- delete from a folder

```
<filesystem>
    <folder name = “menu”>
        <file name="Help_hidden"/>
    </folder>
</filesystem>
```

- install/uninstall module
  - automatically adds/removes its files

Some Details About Layers

- ordering
- .instance vs. .shadow
- .settings files
- instanceCreate
- methodValue
- localization
- icons

How to provide a layer?

1. Create a layer.xml file.
2. Add entries to folders to register your own folders and files.
3. Provide pointer to layer.xml, in manifest.

= use a wizard

How to register in the layer?

- Type manually in the layer.xml file.
- Use wizards to get started.
  - Actions
  - Windows
  - Options panels
  - ...
- Visual editing node
  - Important files
  - Layer/Layer in context

Demo

Let's work with the layer file...

How to access the layer?

```
FileObject dir =
  FileUtil.getConfigFile("Menu");
```

Pre-7.0 version:

```
FileObject root =
  Repository.getDefault().
  getDefaultFileSystem().
  getRoot();


FileObject dir =
  root.getFileObject("Menu");
```

Let's see what's in 'Menu'...

```
public void performAction() {
    FileObject dir = FileUtil.getConfigFile("Menu");

    FileObject[] kids = dir.getChildren();
    for (int i = 0; i < kids.length; i++) {
          FileObject fileObject = kids[i];
          String name = fileObject.getName();
          JOptionPane.showMessageDialog(null, name);
    }

}
```

Demo...

Let's access it from code...

Is not that Too Complex I?

- Why learn File system API?
- I like Lookup!
- There is Lookups.forPath(String ctx)
  - Like FileUtil.getConfigFile(ctx)
  - Typed access
  - Understands .instance, .settings files

Is not that Too Complex II?

- Why edit XML File system?
- Use annotations!
  - like @ConvertAsProperties
  - completion
  - type checked
  - find usages
  - define your own.

```
@ConvertAsProperties(
    dtd="-//org.nb.linz.demo//ViewButton//EN",

)
pu  autostore = true    boolean
    ignoreChanges = {} String[]  pComponent exter
```

Demo...

Use of annotations...
What they generate...

Summary

- General Registry
- Raw Data
- Basic folder based API
  - Menu/, Toolbars/, OptionsDialog/, Loaders/
- Access via
  - FileUtil.getConfigFile(ctx)
  - Important Files Node
  - Lookup.forPath(ctx)
  - @Annotations

Questions & Answers