

Nashpati Framework



www.bottlesoftware.com

Manish Kumar
www.bottlesoftware.com



What is Nashpati

*It is Simple, Easy to learn, MVC Framework
for Android application Development*



How to Download

- *Nashpati-1.1.0.jar*

<http://www.bottlesoftware.com/download.php?file=nashpati-1.1.0.jar>

- *API DOCs*

<http://www.bottlesoftware.com/nashpati/api-doc/index.html>



Components of Nashpati

- Nashpati Activity: Nashpati Activity is Nashpati counterpart of Android Activity. Nashpati Activity can be created by extending `com.bottlesoftware.nashpati.activity.BaseActivity` and having same lifecycle as Android Activity.
- Nashpati BroadcastReceiver: This is the Broadcast Receiver used in Nashpati Project . A Nashpati BroadcastReceiver can be created by extending `com.bottlesoftware.nashpati.broadcastreceiver.BaseBroadcastReceiver` and having same lifecycle as Android BroadcastReceiver.



Components of Nashpati

- Command : Commands are helper classes which encapsulates the a part of processing logic of request. Command chain is sequence of one or more commands. Any Command in Nashpati must extend `com.bottlesoftware.nashpati.commandprocessor.AbstractCommand`
- Command Request: This is Model part of Framework. Encapsulates input command and model data.
- Exception View: This is Nashpati Activity shown in case of Exception occur while the execution of command chain.
- Commands.xml : Configuration for Framework Container

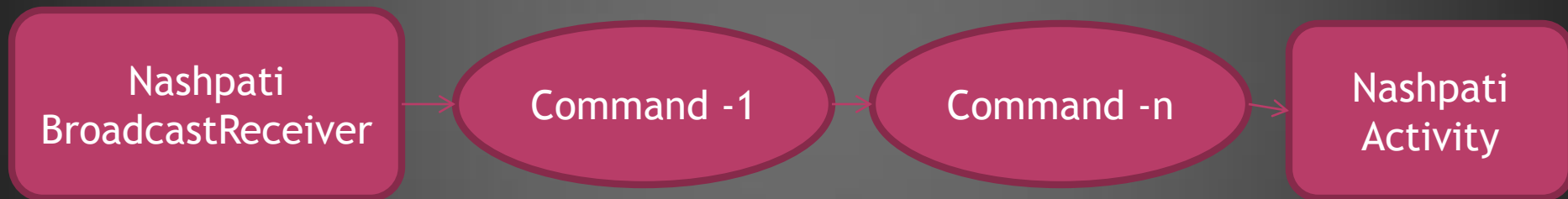


Architecture of Nashpati

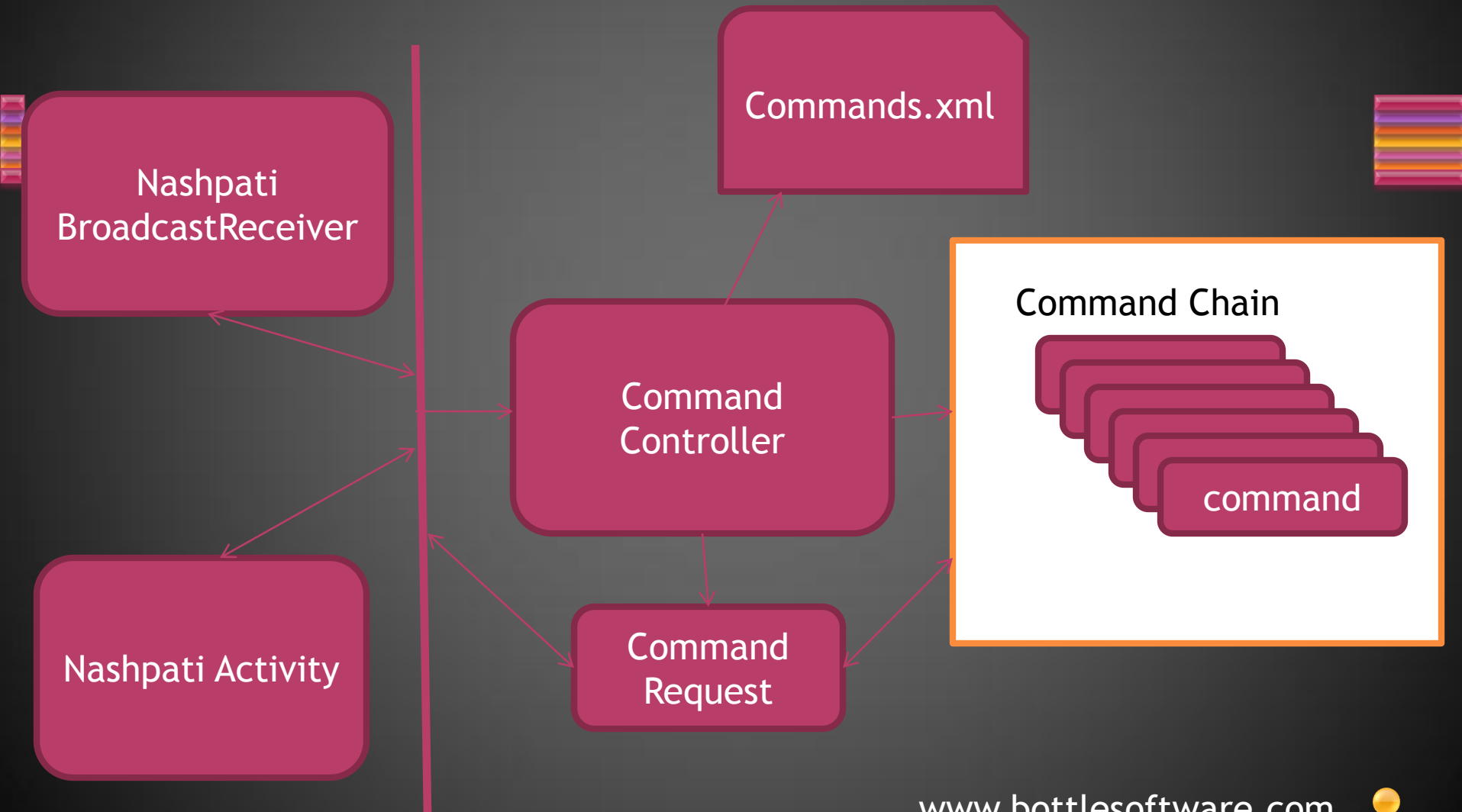
- Following three are Request Processing Flow in Nashpati Framework



Architecture of Nashpati



Architecture of Nashpati



Commands.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://bottlesoftware.com/xml/ns/
  nashpati/commands.xsd">
  <commands>
    .....
    .....
  </commands>
  <commandviews>
    -----
    -----
  </commandviews>
  <exceptionview>-----</exceptionview>
  <commandchains>
    -----
    -----
  </commandchains>
</config/>
```



Commands.xml

```
<commands>
  <command>
    <view-name>command-1</view-name>
    <view-class></view-class>
  </command>
  -----
  -----
  <command>
    <view-name>command-n</view-name>
    <view-class></view-class>
  </command>
</commands>
```



Commands.xml

```
<commandviews>
  <command-view>
    <view-name>view-1</view-name>
    <view-class></view-class>
  </command-view>
```

```
<command-view>
  <view-name>view-n</view-name>
  <view-class></view-class>
</command-view>
```

```
</commandviews>
```



Commands.xml

<exceptionview>com.bottlesoftware.test.nashpati.ExceptionActivity</exceptionview>





Commands.xml

```
<commandchains>
  <commandchain name="request-1">
    <command-name>command-1</command-name>
    -----
    -----
    <command-name>command-n</command-name>
    <command-view>final-view</command-view>
  </commandchain>
```



Commands.xml



```
</commandchains>  
  <commandchain name="request-1">  
    -----  
    -----  
  </commandchain>  
  <commandchain name="request-n">  
    -----  
    -----  
  </commandchain>  
</commandchains>
```



Calling Command chain from Nashpati Activity and Receiving Result if any

- *Receiving Result in Same Activity:*

```
CommandRequest request = new CommandRequest();  
request.setCommandName("test");  
request.setAttribute("text", "This is test Text");  
executeCommand(request);  
Object result = request.getAttribute("key");
```



Calling Command chain from Nashpati Activity and Receiving Result if any

- *Receiving Result in Next view Activity:*

```
CommandRequest request =  
(CommandRequest) getIntent().getSerializableExtra("request");  
Object value= request.getAttribute("key");
```



Calling Command chain from Nashpati BroadcastReceiver and Receiving Result if any

```
public void onReceive(Context context, Intent intent) {  
    .....  
    // Parse Intent  
    .....  
    .....  
    CommandRequest request = new CommandRequest();  
    request.setCommandName("test");  
    //Set Intent data to request  
    request.setAttribute("text", "This is test Text");  
    executeCommand(context ,request);  
}
```



Creating Command

```
public class TestCommand extends AbstractCommand {  
    public boolean execute()throws Exception {  
        request.setAttribute("result", "Test Command Executed");  
        Toast.makeText(context, "Test Command Executed",  
            Toast.LENGTH_LONG).show();  
        executeNext();  
        return true;  
    }  
}
```



Creating Command

- *executeCommand() method will call next command in sequence or render next Activity.*
- *In case of Call back methods call executeCommand() from callback.*





Thank You

