Enhancing Robolectric for Comprehensive Dynamic Feature Module and AAB Testing Support

Google Summer of Code Program 2024 Project Proposal

Shashank Kumar
Github

Project Abstract

This project aims to extend Robolectric's testing capabilities to fully support dynamic feature modules and the AAB publishing format, addressing the current limitations in activity launching, resource loading, and environment setup for testing apps that utilize dynamic feature delivery and on-demand feature modules.

Project Description

With Google Play's adoption of the AAB format as the standard publishing format, optimizing app delivery has become more efficient, allowing apps to only download the code and resources needed for a user's specific device configuration. Dynamic feature modules further enhance this by enabling on-demand feature downloads. However, this advanced modularization introduces complexities in local testing environments, particularly with tools like Robolectric that are critical for unit testing Android applications.

Objectives

- Activity Resolution Enhancement: Modify Robolectric's internal mechanisms to recognize and correctly launch activities contained within dynamic feature modules.
- Resource Loading Improvement: Extend Robolectric's resource loading capabilities to seamlessly access and utilize resources from dynamic feature modules, addressing `Resources$NotFoundException` errors.
- Documentation and Examples: Create detailed documentation and example projects demonstrating the setup and usage of Robolectric in testing apps with dynamic feature modules and AABs.

Approach-

1. Activity Resolution Enhancement

Objective: Modify Robolectric to correctly identify and launch activities that reside within dynamic feature modules

Technical Solution:

- Implement a mechanism within Robolectric that can parse and understand the metadata associated with dynamic feature modules. This includes recognizing activities declared in the manifest of dynamic feature modules.
- Adjust Robolectric's `ActivityController` to resolve activities based on both the base app and its dynamic feature module

```java
public class EnhancedActivityController {
  public static Activity launchActivityWithDynamicFeatureSupport(
      Context baseContext, Intent intent) {
    // Logic to resolve activities from dynamic feature modules
    String targetActivityName = intent.getComponent().getClassName();
    Class<?> activityClass = resolveActivityClass(targetActivityName,
baseContext);
    return Robolectric.buildActivity(activityClass).create().get();
  }

  private static Class<?> resolveActivityClass(String activityName, Context
context) {
    // Implement logic to dynamically find and load the activity class

    return Class.forName(activityName);
  }
}
```

2. Resource Loading Improvement

Objective: Enhance resource loading mechanisms to seamlessly access resources from dynamic feature modules.

Technical Solution:

- Extend the resource resolution system within Robolectric to include resources bundled within dynamic feature modules. This may involve augmenting the `ResourceTable` and `ResourceMerger` classes to recognize and merge resources from these modules.
- Ensure that dynamic feature resources are correctly identified and loaded, both when accessed directly in tests and when referenced by activities or other components from dynamic feature modules.

Java
```java
public void mergeDynamicFeatureResources(Context context, ResourceTable
baseResourceTable) {
  List<ResourceTable> dynamicFeatureResourceTables =
getDynamicFeatureResourceTables(context);
  for (ResourceTable featureTable : dynamicFeatureResourceTables) {
    baseResourceTable.merge(featureTable);
  }
}

private List<ResourceTable> getDynamicFeatureResourceTables(Context context) {
  // Logic to load and parse resource tables from dynamic feature APKs
      return new ArrayList<>();
}
```

3. Testing Environment Configuration

Objective: Develop a setup process for simulating a comprehensive AAB environment within Robolectric tests, including dynamic feature loading and unloading.

Technical Solution:

- Introduce a configuration mechanism in Robolectric to specify dynamic feature modules relevant to a test suite. This can include specifying which feature modules are "installed" or "uninstalled" for a given test scenario.
- Implement support for simulating the installation, activation, and removal of dynamic feature modules during tests. This will likely involve mocking the behaviors of Google Play Services and the dynamic feature management APIs.

```java
@Before
public void setup() {
  DynamicFeatureConfigurator.configureFeatureModules(
    "com.example.feature1", "com.example.feature2" // Feature modules to include
  );
}

@Test
public void testFeatureModuleActivity() {
  Intent intent = new Intent(context, Feature1MainActivity.class);
  Activity activity =
EnhancedActivityController.launchActivityWithDynamicFeatureSupport(context,
intent);
  assertNotNull(activity);
}
```

4. Documentation and Examples

To ensure widespread adoption and effective use of these enhancements, comprehensive documentation and examples will be critical. This documentation will cover:

- Setup and configuration instructions for using Robolectric with dynamic feature modules and AABs.
- Detailed examples demonstrating the testing of activities, services, and resources across the base and dynamic feature modules.
- Best practices for structuring tests to accommodate modular app architectures.

Timeline

Phase 1: Preliminary Research and Planning

May 1 - May 14, 2024

Objective: Conduct an in-depth analysis of the current state of Robolectric in relation to dynamic feature module support and plan the development strategy.

Actions:

- Review Current Capabilities: Assess Robolectric’s current mechanisms for activity resolution and resource loading.
- Identify Gaps: Pinpoint specific gaps in support for dynamic feature modules and AABs.
- Strategic Planning: Develop a detailed plan for addressing identified gaps, including technical solutions and a prototype design.
- Learn :
    - 1. How AGP build and package dynamic features with multiple splitted apks.
    - 2. How an app identify dynamic feature parts, and load them.
    - 3. How dynamic features apks are installed in Android device.
    - 4. How AOSP load specific apk based on app requested when running the main apk?
    - 5. How does Robolectric load package and resources.
- Finally, Publish an Article to Introduce these details.

Phase 2: Documentation, Examples, and Community Feedback

May 14 - June 6, 2024

Objective: Create comprehensive documentation and examples for the new features and gather feedback from the community.

Actions:

- Documentation: Write detailed documentation on configuring and using the new Robolectric features for dynamic feature modules.
- Example Projects: Develop example projects that demonstrate testing with dynamic feature modules in Robolectric.

- Community Engagement: Share the developments with the community for feedback and contributions.

Phase 3: Development of Activity Resolution and Resource Loading Enhancements

June 6 - July 20, 2024

Objective: Implement enhancements in activity resolution and resource loading to support dynamic feature modules.

Actions:

- Implement Activity Resolution Enhancement: Develop the enhanced activity launching mechanism to support dynamic feature modules.
- Enhance Resource Loading: Modify Robolectric's resource resolution system to include dynamic feature module resources.
- Unit Testing: Create unit tests for new functionalities ensuring they work as expected.

Mid Evaluation

- Submit Investigation Articles
- Write Basic Setup Code and Package Loading

Phase 4: Testing Environment Setup and Configuration

July 20 - August 7, 2024

Objective: Develop and implement a comprehensive testing environment setup for simulating dynamic feature modules within Robolectric tests.

Actions:

- Configuration Mechanism: Create a configuration system for specifying dynamic feature modules in test scenarios.

- Simulation Support: Implement the ability to simulate the installation, activation, and removal of dynamic feature modules.
- Integration Testing: Conduct integration tests to validate the functionality of the testing environment with dynamic feature modules.

Phase 5: Finalization, Optimization, and Comprehensive Testing

August 7 - September 3, 2024

Objective: Finalize the implementation, optimize performance, and conduct comprehensive testing of the new features.

Actions:

- Performance Optimization: Analyze and optimize the performance of the new activity resolution and resource loading enhancements.
- Comprehensive Testing: Perform thorough testing, including stress and edge-case scenarios, to ensure robustness and reliability.
- Final Adjustments: Make any necessary adjustments based on testing outcomes and community feedback.

Final Submission and Evaluation

September 4 - September 11, 2024

Objective: Prepare and submit the final package of enhancements, documentation, and examples for Robolectric's support of dynamic feature modules and AABs.

Actions:

- Final Review: Conduct a final review of all code, documentation, and example projects.
- Submission Package Preparation: Compile the final submission package, including all developed code, comprehensive documentation, and detailed project report.
- Evaluation: Submit the package for final evaluation, including benchmarks and a summary of contributions to the Robolectric project.

Background about me

I am a seasoned Android app developer with a strong track record of creating successful applications. My notable achievements include:

- Developing and publishing Android applications that have collectively amassed over **40,000 downloads**. One of my applications, the Android-Java Programming Tutorial App, serves as an educational tool for learning Java programming on Android devices.
- Founding the startup "**Opined**," where I developed a social media application using Java, Kotlin, and Jetpack Compose. This app, Opined, achieved over 10,000 installs on the Play Store and was selected for **Google's Appscale Academy 2022 Cohort**, a testament to its quality and potential. More about the program.
- Earning certifications as a **Google Certified Associate Android Developer** and an **Oracle Certified Java Programmer**, which validate my expertise in Android development and Java programming. View my certification.
- Contributing to the Wikimedia Commons Android App (Which uses Robolectric), where I have submitted several pull requests that have been successfully merged, showcasing my ability to collaborate on open-source projects.

Internet presence

- LinkedIn
- Github

References

Following are sources of the research that I did for this Project Idea:

Roboelectric Official Documentation

https://developer.android.com/guide/app-bundle

https://developer.android.com/guide/playcore/feature-delivery

https://github.com/robolectric/robolectric/issues/5597