Bound Service – שירות כרוך ובקיצור bs

הקדמה

Bound service הוא צד השרת מצד ממשק המשתמש.
bs מאפשר לרכיבים להיקשר ל service, לשלוח בקשות, לקבל תגובות ואף לבצע תקשורת בין תהליכים.
ה- bs חי כל עוד הוא משרת רכיבי אפליקציה אחרים ורץ ברקע עם הגבלת זמן.

הבסיס

ה- bs הוא יישום של מחלקת service שמאפשר לאפליקציות אחרות להיקשר אליו ולבצע עימו אינטרקציות. בכדי לספק קשירה ל service, חייב ליישם את מתודת ההתקשרות (callBack) onBind(). מתודה זו מחזירה אובייקט הנקרא IBinder שמזהה את תכנות הממשק שהרכיב האפליקציוני יכול להשתמש בו כי לבצע אינטרקציה עם ה service.
למעשה אנו יוצרים ממשק ומבצעים bind כדי שנוכל לשלוט בפונקציות השונות שבו.
הרכיב האפליקציוני יכול להיקשר ל service על ידי קריאה ל- bindService(). ולאחר שזה נעשה, חייב לספק יישום ל- ServiceConnection אשר מציג את התקשורת עם ה- service. מתודת ה- bindService() מחזירה באופן מידי ללא ערך, אך כאשר מערכת האנדרואיד יוצרת תקשורת בין הרכיב האפליקציוני וה- service, היא קוראת ל- onServiceConnected() כדי לשלוח את ה- IBinder שהרכיב יכול להשתמש בו כדי לתקשר עם ה- service.
ייתכן מצב שבו מספר רכיבים יקשרו ל- service בבת אחת. עם זאת, המערכת קוראת ל- onBind() כדי לשלוף את אובייקט ה- IBinder רק כאשר הרכיב הראשון נקשר. אז המערכת שולחת את אותו IBinder לכל רכיב נוסף שנקשר, מבלי לקרוא שוב ל- onBind().
כשהרכיב האחרון מתנתק (unbinds) מה- service, המערכת משמידה את ה- service (אלא אם כן ה- service התחיל **גם** על ידי startService()).
כשמיישמים את ה- bs, הדבר החשוב ביותר הוא לזהות את הממשק שמתודת ה- onBind() מחזירה. קיימות מספר דרכים לזהות את ממשק ה- IBinder ובהמשך נדון במספר טכניקות קיימות לזיהוייו.

יצירת Bound service

כשיוצרים service שמספק קשירה, חייב לספק לו IBinder המספק את תכנות הממשק שהרכיב יכול להשתמש בו כדי לתקשר עם ה- service. יש שלוש דרכים איתן ניתן לזהות את הממשק:

1. **הורשת מחלקת binder**-
אם ה- service הוא לשימוש האפליקציה שלך בלבד ורץ באותו תהליך כמו הרכיב שמפעיל אותו, אז עליך ליצור את הממשק שלך על ידי ירושה ממחלקת Binder ולהחזיר ממנו דוגמא מ- onBind(). רכיב שמקבל את ה- Binder ויכול להשתמש בו לקבלת גישה ישירה למתודות ציבוריות (public) הזמינות ביישום ה- Binder או אפילו ב- Service.
זוהי הטכניקה המועדפת כאשר ה- service שלך עובד ברקע של האפליקציה שלך בלבד.
הסיבה היחידה שבגללה לא תיצור ממשק בדרך זו היא כאשר ה- service משמש אפליקציות נוספות או כאשר הוא רץ על תהליכים (process) נפרדים.
השלבים:
1. ב service ניצור מופע של Binder שהוא אחד מאלה:
   - מכיל מתודה ציבורית שהרכיב יכול לקרוא לה.
   - מחזיר את מופע ה service הנוכחי, שיש לו מתודה ציבורית שהרכיב יכול לקרוא לה.
   - או, מחזיר מופע של מחלקה אחרת "המתארחת" על ידי ה service בעלת מתודה ציבורית שהרכיב יכול לקרוא לה.
2. להחזיר את המופע הזה של Binder מהמתודה החוזרת onBind()
3. ברכיב עצמו, לקבל את ה Binder מהמתודה החוזרת onServiceConnected() ולבצע קריאה ל- bound service תוך שימוש במתודה המסופקת.

לשים ♥: הסיבה שה service והרכיב חייבים להיות באותה אפליקציה היא שהרכיב יוכל להטיל את האובייקט החוזר ולקרוא ל- APIs שלו כהלכה. ה service והרכיב חייבים להיות גם באותו הליך, בגלל שטכניקה זו אינה מבצעת שום עריכה בין תהליכים.

2. **שימוש ב- Messenger-**
אם יש צורך שהממשק יחצה תהליכים נפרדים, טכניקה זו מאפשרת לבצע תקשורת בין תהליכים מבלי להשתמש ב AIDL.באמצעות שיטה זו ניתן ליצור ממשק עבור ה- service באמצעות Messenger. בדרך זו, ה- service מזהה Handler שמגיב לסוגים שונים של אובייקטי Message. ה- Handler הזה הוא הבסיס עבור Messenger שבאמצעותו יכול לחלוק את ה- IBinder עם הרכיב, ולאפשר לרכיב לשלוח פקודות ל- service תוך שימוש באובייקט ה- Message. בנוסף, הרכיב יכול לזהות Messenger של עצמו כך שה- service יכול לשלוח הודעה בחזרה.
זוהי הדרך הפשוטה ביותר לבצע תקשורת בין תהליכים, מאחר וה- Messenger מסדר את הבקשות בתור ל thread בודד כך שאיננו צריכים לדאוג לאבטחת ה- thread של ה service.
השלבים:
1. ה service מיישם Handler שמקבל callback עבור כל קריאה מהרכיב.
2. ה Handler משמש ליצירת אובייקט Messenger (שהוא מצביע ל- Handler).
3. ה- Messenger יוצר IBinder שה service מחזיר לרכיב מהמתודה onBind().
4. הרכיב משתמש ב IBinder ליצירת מופע של ה Messenger (שהוא מצביע את ה Handler של ה service), שהרכיב משתמש בו לשליחת אובייקט ה Message אל ה service.

5. ה service מקבל כל הMessage ב- Handler הספציפי שלו במתודה handleMessage().

בדרך הזאת, אין לרכיב מתודות שיתקשרו עם ה service, במקום זה, הרכיב שולח הודעות (אובייקט ה- Message) שה service מקבל ב Handler שלו.

3. שימוש ב- AIDL-

Android Interface Definition Language -AIDL
מבצע את כל העבודה לפרק אובייקטים לפרימיטיביים כך שמערכת ההפעלה תוכל להבין ולסדר אותם בין תהליכים כדי לבצע תקשורת בין תהליכים. בטכניקה הקודמת, השימוש ב- Messenger, היא למעשה מבוססת על המבנה הבסיסי של AIDL כמוזכר למעלה, ה- Messenger יוצר תור של כל בקשות הרכיבים ב thread בודד, כך שה- service מקבל בקשה אחת בכל פעם. כך שאם אנו רוצים שה service יתמודד עם בקשות מרובות בו- זמנית אז ניתן להשתמש ישירות ב- AIDL. במקרה זה, ה- service חייב להיות מסוגל להחזיק tread מרובים ומסוגל לבנות thread מאובטח.
בכדי להשתמש ב- AIDL ישירות, חייב ליצור תיקיית aidl. שמזהה את תכנות הממשק. SDK tools של אנדרואיד משתמש בתיקייה זו כדי ליצור מחלקה מופשטת שמיישמת את הממשק ומתמודדת על יצירת תקשורת בין תהליכים, שאז ניתן לרשת אל תוך ה service.
לשים ♥: מרבית האפליקציות לא אמורות להשתמש ב- AIDL כדי ליצור bound service, בגלל שזה עלול לדרוש יכולת ל טריידים מרובים ויכול לתת תוצאות ביישומים מסובכים יותר. אם כך, AIDL לא מתאים לרוב האפליקציות ולכן אנו לא עוסקים כאן באיך להשתמש בו ב- service. לכך יש פרק נפרד העוסק ב- AIDL.

כבילה ל- service (Binding)

רכיבי אפליקציה יכולים להיקשר ל service על ידי קריאה ל bindService(). אז מערכת האנדרואיד קוראת למתודת ה service onBind(), אשר מחזירה את IBinder היוצר אינטרקציה מול ה service.
הכבילה (=binding) הוא א-סינכרוני. bindService() מחזיר באופן מיידי ולא מחזיר את ה IBinder לרכיב. על מנת לקבל את ה IBinder, הרכיב חייב ליצור מופע של ServiceConnection ולהעביר אותו אל bindService(). ה- ServiceConnection כולל מתודת callback שהמערכת קוראת לה כדי לשלוח את ה IBinder.
לשים ♥: רק אקטיביטיס, סרוויס וספקי תוכן יכולים להיכבל ל- service. לא ניתן להיכבל ל service מתוך broadcast receiver.
כך, שעל מנת להיכבל ל service מתוך הרכיב מוכרחים ל:

1. ליישם את ServiceConnection, היישום מחייב לייבא שני מתודות callback-
   onServiceConnected()- המערכת קוראת לה כדי לשלוח את ה IBinder שחזר ממתודת ה service onBind().
   onServiceDisconnected()- מערכת האנדרואיד קוראת למתודה זו כאשר התקשורת ל service נקטע בפתאומיות, למשל כאשר ה service קרס או הושמד. המתודה **איננה** נקראת כאשר הרכיב ביצע unbinds.
2. קריאה ל bindService(), העברת היישום של ServiceConnection.
3. כשהמערכת קוראת למתודה onServiceConnected(), אפשר להתחיל לבצע קריאת ל service, תוך שימוש במתודות המזוהות על ידי הממשק.
4. כדי להתנתק מה service נקרא ל unbindService().

כאשר הרכיב מושמד הוא ינתק (unbind) מה service, עם זאת, תמיד צריך לבצע unbind כשמסיימים את האינטרקציה עם ה service או כאשר האקטיביטי מושהה (pauses) כך שה service יכול להיסגר כשהוא לא בשימוש.

הערות נוספות

- תמיד נכניס לקוד DeadObjectException אשר נזרקת כאשר התקשורת נקטעת. זוהי החריגה היחידה מתוך המתודות "באוויר".
- אובייקטים הם מצביעים הנספרים מעבר לתהליכים.
- בדרך כלל נשים בזוגות את bindind ו unbinding במהלך השלבים המקבילים במחזור החיים. למשל: onCreate==onDestroy onStart==onStop
- כדי שמישהו יוכל לעשות פעולת bind הוא חייב לקבל IBinder.
- ה IBinder נותן גישה לדבר עם הממשק.
- אם הוא מחזיר null הוא לא יוכל לדבר איתו.
- כדי שהסרוויס יהיה מסוגל לעשות bnd צריך לדעת לממש IBinder ורק אז אקטיביטי יוכל לזרוק חבל ולתקשר.
-

לשים ♥: בדרך כלל לא אמורים לעשות bind ו- unbind בשלבים onResume ו- onPause בגלל שה callBacks האלו מתרחשים בכל מעבר של מחזור חיים ועלינו לשמור את התהליך שהתרחש במינימום. כמו כן, אם מספר אקטיביטיס באפליקציה נקשרו לאותו service ויש שם החלפה בין שניים מתוך האקטיביטי האלו, ה service עלול להיות מושמד ולהיווצר מחדש כאשר האקטיביטי הנוכחי יתנתק (unbind) **לפני** שהבא בתור יכבל.

מחזור החיים של bound service

כאשר service אינו כבול לאף רכיב, מערכת האנדרואיד משמידה אותו (אלא אם כן ה service אותחל **גם** באמצעות onStartCommand()) אם כך, אין צורך לנהל את מחזור החיים של ה service אם הוא bound service טהור- אז מערכת האנדרואיד מנהלת את זה עבורנו על בסיס כך שה service כבול לאיזה שהוא רכיב.
בכל אופן, אם נבחר ליישם את מתודת ה callback onStartCommand(), אז חייבים להיות ספציפיים בעצירת ה service, בגלל שה service עכשיו מחכה להיות מאותחל.
במקרה כזה, ה service רץ עד שה service עוצר את עצמו באמצעות stopSelf() או שרכיב אחר קורא ל stopService(), בלי קשר לאם הוא כבול לרכיב או לא.
בנוסף, אם ה service מאותחל ומקבל binding, אז כאשר המערכת קוראת ל onUnbind() אז יש אופציה להחזיר true במקרה שנרצה להחזיר קריאה ל onRebind()

בפעם הבאה שרכיב יכבל אל ה service (במקום להחזיר קריאה ל ()onBind) המתודה ()onRebind מחזירה void, אך הרכיב עדיין מקבל את IBinder ב callback של ()onServiceConnected.

אסטרטגיית קולבק

מאחר ואנחנו עובדים עם return, אפשר לעבוד עם listener.
צריך שב onStart הוא יעביר callback שיאזין לסיום הפעולה- שמים אותו בסוגריים OnCompletionListener
נשמור את הlistener לפני שהוא יתחיל לנגן, וכשהוא יסיים לנגן הוא ייכנס ל ()onComplition.הנכנס מטרייד אחר שהוא זר, אינו ה Main Tread.
צריך למצוא את הטרייד שהוא בא ממנו
יצרנו טרייד חדש ושם כתבנו כל מיני פעולות,
שמנו לו מאזין לסיום כל הפעולות ועכשיו אנחנו רוצים לחזור למיין טרייד יחד עם המידע שרץ שם- אנחנו צריכים כלי שיצור צינור בין הMain ובין ה thread החיצוני.
נעשה זאת באמצעות handler והפעולה תתבצע בתוך הhandler וכך נוכל להעביר ב post. וזה מבטיח שכל פעולה הוא יבצע בMain.
ה handler שעובר ב listener נוצר בMain ויושב עליו וכלדבר שנכניס לו הוא יריץ בMain אקטיביטי.
שיטה נוספת לתקשר בין שני טריידים שונים באמצעות Handel Messeges.